K.R. MANGALAM UNIVERSITY

THE COMPLETE WORLD OF EDUCATION

SCHOOL OF ENGINEERING AND TECHNOLOGY

Programme Handbook

(Programme Structure and Evaluation Scheme)

B. Tech (Computer Science & Engineering)

Programme Code: 01

FOUR YEAR UNDERGRADUATE PROGRAMME

(with effect from 2026-27 session)

Session: 2026-30

Approved in the 40th Meeting of Academic Council

K.R. MANGALAM UNIVERSITY

Table of Contents

S.N.	Content	Page No.
1	Preamble	03
2	Categories of Courses	05
3	University Vision and Mission	07
4	About the School	08
5	School Vision & Mission	10
6	About the Programme	11
7	Programme Educational Objectives (PEO)	13
8	Programme Outcomes (PO)	14
9	Programme Specific Outcomes (PSO)	15
10	Career Avenues	16
11	Duration	19
12	Eligibility Criteria for Award of Degree	19
13	Student's Structured Learning Experience from Entry to Exit in the Programme	20
14	Evaluation Scheme	33
15	Scheme of Studies	34
16	Syllabus	43

1. Preamble

Welcome to the School of Engineering & Technology (SOET) at K.R. Mangalam University -a vibrant ecosystem of innovation, technology, creativity and experiential learning dedicated to shaping the next generation of engineers, innovators and technology leaders.

Established in 2013, SOET has emerged as a leading center for industry-focused education, project-based learning and skill development. The School is committed to delivering a transformative learning experience that combines strong academic foundations with practical implementation, innovation and professional excellence. With state-of-the-art infrastructure, advanced laboratories, modern computing facilities and distinguished faculty members, SOET provides an environment where students are empowered to learn, build, innovate and grow.

The School offers a wide range of undergraduate, postgraduate and doctoral programs including B.Tech, BCA, B.Sc, M.Tech and MCA across emerging technology domains. Specialized B.Tech programs in Artificial Intelligence & Machine Learning, Data Science, Cyber Security, Full Stack Development, UI/UX Development and Robotics & AI are carefully designed to align with evolving industry requirements and future technology trends.

At SOET, learning extends far beyond traditional classrooms. The curriculum emphasizes project-based learning, coding culture, real-world problem solving, interdisciplinary learning and hands-on implementation from the very first year. Students actively participate in coding competitions, hackathons, bootcamps, innovation challenges, technical clubs, workshops and industry projects that strengthen technical competency, creativity and professional confidence. Signature initiatives such as Hack-KRMU, MindBenders, Robo-Rush and participation in the Smart India Hackathon encourage students to develop innovative solutions for real-world challenges.

The School maintains strong industry integration through collaborations with globally recognized organizations such as IBM, Microsoft, NASSCOM, EC-Council, Xebia and ImaginXP. Industry experts contribute to curriculum design, bootcamp programs, certifications, workshops,

internships and mentoring activities, ensuring that students gain practical exposure to modern technologies, tools and industry workflows.

SOET also promotes a strong self-learning and certification culture aligned with NEP 2020 principles. Students are encouraged to pursue MOOCs, global certifications and industry learning pathways through platforms such as Google, AWS, Microsoft, Cisco, Coursera and NPTEL to strengthen lifelong learning and employability skills. Dedicated Centres of Excellence in AI & Machine Learning and Robotics & Automation further support advanced learning, research and innovation activities through cutting-edge infrastructure and practical experimentation facilities.
With its strong focus on employability, innovation, entrepreneurship and practical learning, SOET continues to nurture future-ready professionals capable of contributing effectively to the rapidly evolving global technology ecosystem. We welcome you to a community where excellence, innovation and continuous learning form the foundation of every student's journey.

School of Engineering & Technology
K.R Mangalam University

2. Categories of Courses

Major: The major would provide the opportunity for a student to pursue in-depth study of a particular subject or discipline.

Multidisciplinary (Open Elective): These courses are intended to broaden the intellectual experience and form part of liberal arts and science education. These introductory-level courses may be related to any of the broad disciplines given below:

- Natural and Physical Sciences
- Mathematics, Statistics, and Computer Applications
- Library, Information, and Media Sciences
- Commerce and Management
- Humanities and Social Sciences

A diverse array of Open Elective Courses, distributed across different semesters and aligned with the aforementioned categories, is offered to the students. These courses enable students to expand their perspectives and gain a holistic understanding of various disciplines. Students can choose courses based on their areas of interest.

Ability Enhancement Course (AEC): In alignment with the vision and recommendations of the National Education Policy (NEP) 2020, the Ability Enhancement Courses (AEC) are designed to develop essential communication, analytical, quantitative and professional competencies required for holistic student development, employability and lifelong learning. The courses emphasize practical skill development, critical thinking, problem-solving ability, communication proficiency and workplace readiness through experiential and application-oriented learning approaches.

The AEC curriculum comprising Foundations of Everyday Communication, Professional & Workplace Communication, Advanced Communication and Interview Mastery and Arithmetic & Reasoning focuses on strengthening students' verbal and written communication skills, interpersonal effectiveness, aptitude, logical reasoning, interview preparedness and professional confidence. These courses support NEP 2020 objectives by promoting multidisciplinary learning,

skill enhancement, experiential education and competency-based development necessary for academic progression, career readiness and active participation in the modern knowledge economy.

The courses are delivered through interactive learning methods including presentations, group discussions, role plays, workplace simulations, aptitude exercises, mock interviews, problem-solving activities and self-learning components to ensure practical exposure and continuous professional growth.

Skills Enhancement Courses (SEC): These courses are aimed at imparting practical skills, hands-on training, soft skills, etc., to enhance the employability of students.

Discipline Specific Electives (DSE): The purpose of offering discipline-specific electives is to provide students with the flexibility to specialize in emerging and high- demand domains such as Full Stack Development, Cloud Computing, AI & ML, and Cyber Security. These electives are designed to equip students with advanced knowledge and skills in their chosen fields, ensuring they are well-prepared for specialized roles and industry demands in these cutting-edge areas.

Industry project/Research Project: Students choosing a 4-Year Bachelor's degree are required to take up Industry/research projects. The purpose of our full- time, 6-month industry project for final-year students is to provide them with practical exposure by working on real-world industry projects.

Massive Open Online Courses (MOOCs): Massive Open Online Courses (MOOCs) are self-paced, online courses offered by globally recognized platforms such as Coursera, edX, NPTEL, Google, AWS, and Microsoft. Students are encouraged to enroll in MOOCs to supplement their academic learning, earn industry-recognized certifications, and develop skills aligned with current technology trends. MOOCs promote self-directed learning, lifelong education, and continuous professional development beyond the formal classroom curriculum.

3. University Vision and Mission

3.1 Vision

K.R. Mangalam University aspires to become an internationally recognized institution of higher learning through excellence in inter-disciplinary education, research, and innovation, preparing socially responsible life-long learners contributing to nation building.

3.2 Mission

- Foster employability and entrepreneurship through futuristic curriculum and progressive pedagogy with cutting-edge technology
- Instill notion of lifelong learning through stimulating research, Outcomes-based education, and innovative thinking
- Integrate global needs and expectations through collaborative programs with premier universities, research centres, industries, and professional bodies.
- Enhance leadership qualities among the youth having understanding of ethical values and environmental realities

4. About the School

School of Engineering and Technology:

Since its establishment in 2013, the School of Engineering and Technology at K.R. Mangalam University has rapidly developed into a hub of innovation, quality education, and skill development. Our focus is on delivering a transformative educational experience that equips students with advanced technical knowledge while fostering creativity and critical thinking. With state-of-the-art infrastructure, modern laboratories, and a distinguished faculty, we provide an environment that nurtures both academic and professional excellence.

Our school offers a comprehensive range of programs, including undergraduate (B.Tech, BCA, B.Sc), postgraduate (M.Tech, MCA), and doctoral studies across key engineering disciplines. We are proud to offer specialized B.Tech programs in high- demand fields such as Artificial Intelligence & Machine Learning, Data Science, Cyber Security, Full Stack Development, UI/UX Development and Robotics & AI. These programs are designed to meet the evolving needs of the industry, ensuring that students are equipped with the skills and knowledge required to succeed in the modern workforce.

Our curriculum is grounded in best practices from leading global institutions and incorporates insights from the Open-Source Society University. It emphasizes interdisciplinary learning, problem-solving, and innovative teaching methodologies. This approach not only enhances students' technical competencies but also develops their ability to think critically and work collaboratively across diverse domains.

Industry integration is a key component of our educational model. We collaborate with renowned organizations such as IBM, Samatrix, Xebia, EC Council, and ImaginXP to provide students with practical, real-world experience through internships, projects, and workshops. These partnerships ensure that our students are well- prepared to meet industry demands. Additionally, we offer elective courses in areas

such as AI, Cloud Computing, Cyber Security, and Full Stack Development, allowing students to tailor their learning experience to align with their career goals.

We are also committed to fostering innovation and entrepreneurship. Our Entrepreneurship and Incubation Center and initiatives like 'MindBenders,' 'Hack- KRMU,' and participation in the Smart India Hackathon inspire students to develop forward-thinking solutions and entrepreneurial ventures.

With cutting-edge computing facilities, advanced research opportunities, and a focus on practical application, the School of Engineering and Technology ensures that its graduates are well-prepared to excel in their careers. Our alumni have made significant contributions across various sectors, reflecting the high standards of education they receive.

5. School Vision and Mission

5.1 Vision

To excel in scientific and technical education through integrated teaching-learning, research, and innovation.

5.2 Mission

- **Creating** a unique and innovative learning experience to enhance quality in the domain of Engineering & Technology.
- **Promoting** Curricular, co-curricular and extracurricular activities that support overall personality development and lifelong learning, emphasizing character building and ethical behavior.
- **Focusing** on employability through research, innovation and entrepreneurial mindset development.
- **Enhancing** collaborations with National and International organizations and institutions to develop cross-cultural understanding to adapt and thrive in the 21st century.

6. About the Programme

The Bachelor of Technology (B.Tech) in Computer Science and Engineering (CSE) is a rigorous undergraduate program designed to equip students with a solid foundation in computer science, engineering principles, and modern technological applications. Spanning four years, the program offers a balanced mix of theoretical knowledge and practical experience. Students are introduced to core subjects such as programming languages, algorithms, data structures, database management, operating systems, computer networks, and software engineering. The curriculum is continuously updated to keep up with the rapid advancements in technology, ensuring that graduates are well-prepared for the evolving demands of the tech industry.

Beyond core computer science topics, the B.Tech CSE program integrates mathematics, electronics, and communication skills to provide a well-rounded education. Students participate in hands-on labs, workshops, and industry-oriented projects that foster problem-solving skills and real-world application of their knowledge. Internship opportunities and industry collaborations offer practical exposure, allowing students to gain insights into professional environments and current industry practices.

One of the key strengths of the B.Tech CSE program is its focus on emerging and future technologies, such as artificial intelligence, machine learning, blockchain, cloud computing, and cybersecurity. This ensures that students are not only proficient in current technologies but also capable of innovating and adapting to future challenges. The program emphasizes research and development, with opportunities for students to work on capstone projects that reflect the latest trends and innovations in computer science.

Overall, the B.Tech in CSE aims to cultivate highly skilled, versatile, and industry- ready professionals who can excel in diverse roles, from software development and system design to data science and AI-driven solutions.

- **Programme Educational Objectives (PEOs)**

  Programme Educational Objectives of a degree programme are the statements that describe the expected achievements of graduates in their career, and what the graduates are expected to perform and achieve during the first few years after graduation.

- **Programme Specific Outcomes (PSOs)**

  Programme Specific Outcomes are statements about the various levels of knowledge specific to the given program which the student would be acquiring during the program.

- **Programme Outcomes (POs)**

  Programme Outcomes are statements that describe what the students are expected to know and would be able to do upon the graduation. These relate to the skills, knowledge, and behaviour that students acquire through the programme.

- **Credit**

  Credit refers to a unit that measures the amount of academic work required for a course. It typically reflects the number of instructional hours per week. For theory courses, one credit is equivalent to 14-15 hours of classroom instruction over a semester. For practical courses, such as workshops, labs, or tutorials, one credit is equivalent to 28-30 hours of instructional or hands-on work over a semester.

7. Programme Educational Objectives (PEO)

PEO1: Successful professionals in industry, government, academia, research, entrepreneurial pursuits and consulting firms.

PEO2: Able to apply their knowledge of computer science & engineering principles to solve societal problems by exhibiting a strong foundation in both theoretical and practical aspects of the field.

PEO3: Dedicated to upholding professional ethics and social responsibilities, with a strong commitment to advancing sustainability goals.

PEO4: Demonstrating strong leadership skills and a proven ability to collaborate effectively in diverse, multidisciplinary teams to successfully achieve project objectives.

8. Programme Outcomes (PO)

Engineering Graduates will be able to:

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development.

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required.

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions.

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems.

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment.

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws.

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and

learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

9. Programme Specific Outcomes (PSO)

PSO1: Understanding the concepts, theories, tools, techniques, and methodologies of Computer Science & Engineering.

PSO2: Applying the concepts, theories, tools, techniques, and methodologies to solve real-world Computer Science & Engineering challenges.

PSO3: Analysing the methodologies context, problems, situations and issues related to Computer Science & Engineering.

PSO4: Evaluating the possible alternative solutions and making choices/decisions to solve problems in Computer Science & Engineering.

PSO5: Designing and **developing** innovative solutions to address complex problems in Computer Science & Engineering

10. Career Avenues

Graduates of the B.Tech Computer Science & Engineering program have a wide array of career opportunities, including:

1. **Software Developer**: Design, code, and maintain software across various domains such as web, mobile, game, and enterprise applications.
2. **Systems Analyst**: Evaluate and enhance organizational computer systems, design new solutions, and optimize existing processes for improved efficiency.
3. **Data Scientist**: Analyze large datasets, apply statistical methods and machine learning, and develop predictive models to drive data-informed decisions.
4. **AI Engineer**: Develop and implement AI algorithms, including natural language processing, computer vision, and other intelligent systems.
5. **Cybersecurity Analyst**: Protect systems and data by identifying vulnerabilities, implementing security measures, and responding to security incidents.
6. **Network Engineer**: Design, deploy, and maintain reliable and secure network infrastructures, and troubleshoot network issues to ensure optimal performance.
7. **IT Project Manager**: Oversee technology projects from planning to execution, manage teams, coordinate resources, and ensure timely and budget-compliant delivery.
8. **Database Administrator**: Maintain database systems, ensure data integrity and security, and optimize performance for efficient data management.
9. **Quality Assurance Engineer**: Test and ensure the reliability and functionality of software applications by developing test plans, identifying issues, and collaborating with development teams.
10. **Research and Development**: Engage in cutting-edge research, explore emerging technologies, and contribute to innovation in academia, industry labs, or R&D departments.
11. **Prospective Companies**:
    - Infosys
    - HCL Technologies
    - LTI Mindtree

- Tech Mahindra
- Wipro
- Tata Consultancy Services (TCS)
- Capgemini
- Cognizant Technology Solutions (CTS)
- Zoho Corporation
- Coforge (formerly NIIT Technologies)

11. Duration

4 Years (8 Semesters) - Full-Time Program

12. Eligibility Criteria for Award of Degree

Students must successfully complete the minimum required credits of 165 to be eligible for award of degree.

13. Student's Structured Learning Experience from Entry to Exit in the Programme

a) Education Philosophy and Purpose:

Learn to Earn a Living: At KRMU we believe in equipping students with the skills, knowledge, and qualifications necessary to succeed in the job market and achieve financial stability. All the programmes are tailored to meet industry demands, preparing students to enter specific careers and contributing to economic development.

Learn to Live: The university believes in the holistic development of learners, fostering sensitivity towards society, and promoting a social and emotional understanding of the world. Our aim is to nurture well-rounded individuals who can contribute meaningfully to society, lead fulfilling lives, and engage with the complexities of the human experience.

b) University Education Objective

Focus on Employability and Entrepreneurship through Holistic Education using Bloom's Taxonomy. By targeting all levels of Bloom's Taxonomy— remembering, understanding, applying, analysing, evaluating, and creating—students are equipped with the knowledge, skills, and attitudes necessary for the workforce and entrepreneurial success. At KRMU we emphasize on learners critical thinking, problem-solving, and innovation, ensuring application of theoretical knowledge in practical settings. This approach nurtures adaptability, creativity, and ethical decision-making, enabling graduates to excel in diverse professional environments and to innovate in entrepreneurial endeavours, contributing to economic growth and societal well-being.

c) Importance of Structured Learning Experiences:

A structured learning experience (SLE) is crucial for effective education as it provides a clear and organized framework for acquiring knowledge and skills. By following a well-defined curriculum, teaching-learning methods and assessment strategies, learners can build on prior knowledge systematically, ensuring that foundational concepts are understood before moving on to more complex topics. This approach not only enhances comprehension but also fosters critical thinking by allowing learners to connect ideas and apply them in various contexts. Moreover, a structured learning experience helps in setting clear goals and benchmarks, enabling both educators and students to track progress and make necessary adjustments. Ultimately, it creates a conducive environment for sustained intellectual growth, encouraging learners to achieve their full potential.

At K.R. Mangalam University SLE is designed as rigorous activities that are integrated into the curriculum and provide students with opportunities for learning in two parts:

Inside the Classroom:

Our educational approach within the classroom is designed to foster **cognitive development** and enhance **student-centric learning**. We prioritize active engagement and deep understanding by employing a variety of methods, tools, and techniques. These include **problem-based learning**, **case studies**, **interactive discussions**, and **technology- enhanced learning platforms**. Our faculty focuses on developing critical thinking, analytical reasoning, and problem-solving abilities, ensuring students achieve well-defined **cognitive outcomes**. Additionally, we integrate the use of **modern teaching tools**, such as Learning Management Systems (LMS), virtual labs, and multimedia resources, to enhance the learning experience and accommodate diverse learning styles. This comprehensive approach not only promotes academic excellence but also nurtures independent learning and lifelong intellectual curiosity.

Outside the Classroom:

Beyond the classroom, our focus shifts to developing students' **people skills** and

psychomotor skills through hands-on experiences in **industry, community, and laboratory settings**. We encourage participation in internships, industrial visits, community engagement projects, and research opportunities, which allow students to apply theoretical knowledge to real-world challenges. These activities build essential interpersonal skills such as **teamwork, leadership, communication**, and **professional networking**. Simultaneously, students engage in **lab-based learning** and technical workshops that refine their psychomotor abilities, including precision, technical expertise, and problem-solving under practical conditions. Through these outside-the-classroom experiences, students gain a holistic skill set that prepares them to excel in both professional and societal contexts, aligning their education with real-world expectations and industry needs.

d) Educational Planning and Execution

The B.Tech in Computer Science & Engineering (CSE) at K.R. Mangalam University is designed to foster a holistic educational experience, integrating both theoretical knowledge and practical skills. The program offers students a structured path from entry to exit, ensuring they develop technical expertise, problem-solving skills, and professional competencies.

Entry Phase: Upon entering the B.Tech CSE programme, students are introduced to the foundational concepts of engineering mathematics, physics/chemistry, and programming. This phase is designed to strengthen their understanding of core scientific and technical principles. Courses such as Engineering Calculus, Fundamentals of Computer Programming using Python, and Basics of Electrical & Electronics Engineering provide a strong foundation. Students also engage in hands-on laboratory sessions to

complement theoretical learning, which helps them connect classroom knowledge with real-world applications.

Orientation Program: The university conducts a **one-day orientation program** for first-year students to familiarize them with the university's environment and key aspects. During the program, students are introduced to the university's highlights, important procedures, key functionaries, and the code of conduct. This orientation serves to ensure that students are well-informed and prepared for a smooth transition into university life.

In the first year, students are exposed to critical problem-solving approaches, basic programming, and ethics in engineering, laying the groundwork for their technical and professional growth.

Induction program: The School organizes a **5-day induction program** for first-year students, aimed at providing them with a comprehensive understanding of the school's various aspects. During the program, students are introduced to learning resources, facilities, and opportunities available to them, along with the rules and regulations governing academic and campus life. The induction also includes faculty introductions, guidelines on academic conduct, and detailed information about examination and evaluation methods, ensuring students are well-prepared for their academic journey.

Core Learning: As students advance through the program, they delve deeper into core computer science subjects such as Data Structures, Algorithms, Object-Oriented Programming (Java), Operating Systems, and Database Management Systems. This phase emphasizes both theoretical concepts and their practical application through lab work. The learning is enhanced through exposure to industry-standard tools and techniques, including programming languages like Java and Python, and systems for data management and networking.

The structured academic schedule, with a well-distributed credit system over eight semesters, ensures students acquire deep technical knowledge and skills in software development, systems design, and computing technologies. The Summer Internship Programs and Minor Projects in the curriculum allow students to apply their learning in real-life projects, facilitating experiential learning.

Summer Internships: School offers 2-credit summer internships spanning 6 weeks, where students are encouraged to pursue internships in startups, industries, or premier institutions such as IITs, NITs, and IIITs. In addition, students have the opportunity to earn global certifications during this period. The School also organizes in-house summer schools in collaboration with industry partners, providing further avenues for students to gain hands-on experience and enhance their professional skills. These initiatives are designed to offer students practical exposure, helping them develop industry-relevant expertise.

Value Added Courses: The School offers a range of 2-credit Value Added Courses (VACs) designed to equip students with industry-relevant skills. These courses aim to bridge the gap between academic knowledge and practical application by providing hands-on training that aligns with current industry demands, ensuring that students are well-prepared for professional challenges.

Skill Development: Throughout the program, there is a significant emphasis on developing practical skills and ensuring students are industry- ready. Courses on Artificial Intelligence, Machine Learning, Cloud Computing, and Cybersecurity provide students with cutting-edge knowledge in emerging fields. Value-Added Courses (VAC) like AWS Cloud Fundamentals, Software Testing, Cyber Security, and Design Thinking & Innovation help bridge the gap between academic learning and industry demands.

Collaborative projects, internships, and industry-based certification courses (offered through partnerships with organizations like IBM and Samatrix) further develop students' practical and professional skills, preparing them to thrive in a dynamic workplace.

Capstone and Exit Phase: In the final semesters, students undertake discipline-specific electives and capstone projects. These projects integrate the knowledge and skills they have acquired over the course of their studies. Electives such as Natural Language Processing, Generative AI, and Blockchain Technologies offer students the flexibility to specialize in areas of their interest.

The final Industrial Project or R&D Project in the eighth semester is a full- time engagement where students work on live industry problems, research projects, or start-up ideas. This project phase, combined with career readiness boot camps and placement preparation activities, ensures that students are equipped to enter the workforce with both technical competence and professional acumen.

Co-Curricular and Extra-Curricular Activities

Students are encouraged to participate in various clubs, societies, and extra-curricular activities. Engagement in activities such as hackathons, coding competitions, and leadership roles in clubs fosters teamwork, leadership, and creativity. These activities complement academic learning, contributing to the students' holistic development.

Community Connect

Aligning with the NEP 2020's vision of social responsibility, the B.Tech CSE programme includes community engagement through activities like Extension Projects and social service initiatives. Students work on community projects and participate in programs aimed at addressing local

and national challenges, promoting civic responsibility, and developing empathy towards society.

Ethics and Professional Values

The program places a strong emphasis on ethics and professionalism. Students are taught to incorporate ethical considerations in technological development and decision-making processes. This prepares them to not only be skilled engineers but also responsible professionals who contribute positively to society.

Career Counseling and Entrepreneurship

The university offers comprehensive **career counseling services**, providing students with expert guidance on **job placements, internships**, and **skill development** to help them effectively navigate their career paths. In addition, the university's **incubation center** plays a pivotal role in nurturing **entrepreneurial and leadership skills**, empowering students to explore innovative ideas and launch their own ventures. These initiatives are designed to equip students with the tools and resources necessary for professional success and entrepreneurial growth.

Course Registration

Every student has to register at the beginning of each semester for the courses offered in the given semester. Major courses are registered centrally for the students. However, for other multidisciplinary courses (DSE, VAC, OE) the students have to register by themselves through ERP.

e) Student Support Services

Mentor-Mentee: At K.R. Mangalam University, the **Mentor-Mentee Program** plays a crucial role in fostering academic and personal growth. Each student is assigned a faculty mentor who serves as a guide throughout

their academic journey. This program ensures continuous interaction, where mentors assist students with academic planning, help in resolving personal issues, and provide career guidance. The mentor-mentee relationship transcends the classroom and often involves personal development, professional growth, and overall well-being. The program aims to nurture a supportive environment that enhances the learning experience and helps students reach their full potential.

Counselling and Wellness Services: The university places a strong emphasis on the mental and emotional well-being of its students through its Counselling and Wellness Services. A dedicated team of trained counselors provides personalized sessions, workshops, and wellness programs to address the mental health needs of the student community. These services focus on holistic well-being, including stress management, emotional resilience, and coping strategies. Regular wellness programs, meditation sessions, and mental health awareness campaigns are conducted to promote a balanced lifestyle and ensure that students can focus on their studies while maintaining their emotional health.

f) Evaluation of Learning:

At K.R. Mangalam University, assessment and evaluation are integral components of the teaching-learning process, designed to ensure continuous academic progress and holistic development of students. The university follows a Learning Outcome-Based Framework (LOCF), where assessments are aligned with the specific learning outcomes of each program. A variety of assessment methods, including assignments, presentations, quizzes, practical examinations, and project work, are used to gauge students' understanding. The examination system is 100% automated, ensuring timely and transparent evaluation processes. Results are processed efficiently, typically within 13 days, and complaints related to evaluation are minimal, reflecting the university's commitment to

maintaining a high standard of academic integrity. This robust system of continuous assessment and feedback fosters a culture of academic excellence and skill development among students.

g) Feedback and Continuous Improvement Mechanisms:

K.R. Mangalam University is deeply committed to academic excellence through a robust **feedback and continuous improvement system**. This system is designed to gather comprehensive input from a diverse range of stakeholders, including **students, faculty, alumni, employers, and academic peers**. Feedback is systematically collected and thoroughly analyzed to identify areas for enhancement in **curricula, teaching methodologies, and academic processes**. Based on the insights gained, actionable measures are formulated and communicated to the appropriate bodies for timely implementation.

This structured feedback mechanism ensures that the university's programs remain aligned with **industry trends and societal needs**, providing students with a cutting-edge education that prepares them for real-world challenges. Moreover, the university demonstrates its commitment to continuous improvement through **regular curriculum updates** and the integration of **innovative teaching strategies**, fostering an environment where both faculty and students can grow and excel. By maintaining this cycle of feedback and improvement, K.R. Mangalam University ensures the continuous advancement of its academic offerings and the overall learning experience.

h) Academic Integrity and Ethics:

K.R. Mangalam University upholds the highest standards of academic integrity and ethics as a core value of its educational philosophy. The university implements a zero-tolerance policy towards academic misconduct, including plagiarism and other unethical practices. To ensure

transparency and honesty in academic work, plagiarism detection software like Drillbit/Turnitin is used to maintain the originality of student submissions and research outputs. Students and faculty are regularly sensitized on the importance of ethical behavior through workshops, seminars, and classroom discussions. The university also integrates ethics and professional values into its curriculum across various disciplines, ensuring that graduates not only excel academically but also demonstrate integrity and responsibility in their professional and personal lives.

i) Capstone Project-Based Learning & Self-Learning Ecosystem:

The School of Engineering & Technology (SOET), K.R. Mangalam University strongly believes in experiential, project-based and self-driven learning as essential components of modern engineering education aligned with the vision of NEP 2020 and contemporary industry expectations. To strengthen practical understanding, innovation, problem-solving ability and lifelong learning skills, every course incorporates a Capstone Project Assignment as an integral academic component.

The Capstone Project is designed as the 5th assignment within each course, where students are expected to apply the cumulative knowledge, concepts, tools and techniques learned across all units of the subject to solve a practical mini-project or real-world problem. Unlike traditional assignments focused only on theoretical understanding, the Capstone Project emphasizes implementation, experimentation, creativity, analysis, design thinking and solution development.

The Capstone Assignment encourages students to work in a highly self-learning and exploratory mode. Students are expected to independently explore additional technologies, frameworks, documentation, research materials, online resources and practical implementation approaches beyond the prescribed syllabus. Faculty members act as mentors and facilitators, guiding students towards industry-oriented thinking, innovation and structured problem-solving rather than limiting learning only to classroom delivery.

As part of this initiative, students are expected to invest approximately 45–50 hours of additional self-learning and implementation effort in completing the Capstone Project Assignment. This approach helps students strengthen:

- practical implementation skills,
- coding and development capabilities,
- analytical and design thinking,
- documentation and presentation skills,
- research orientation,
- teamwork and collaboration,
- independent learning habits and
- professional confidence.

To further promote lifelong learning and continuous skill development, faculty members actively encourage students to undertake additional MOOCs, industry certifications, open-source learning activities, technical workshops, coding platforms and free online courses relevant to the course domain. Students are motivated to explore modern industry trends, emerging technologies and interdisciplinary applications beyond the prescribed curriculum to become self-sufficient learners and future-ready professionals.

The Capstone Project framework also promotes:

- experiential learning,
- innovation culture,
- portfolio development,
- industry readiness,
- practical problem-solving and
- outcome-based education practices.

Through this structured self-learning ecosystem, students gradually develop the ability to independently learn, adapt and innovate in rapidly evolving technological environments, thereby nurturing the qualities of lifelong learners, innovators and responsible technology professionals.

Capstone Project Evaluation Rubrics (recommended) (20 Marks)

Evaluation Component	**Description**	**Marks**
Problem Understanding & Solution Design (Viva-Based Evaluation)	Understanding of the problem statement, clarity of objectives, architecture/design thinking, justification of approach, technical decision-making and conceptual understanding evaluated through viva/personal interaction	5

Technical Implementation & Development Contribution (GitHub-Based Evaluation)	Quality of implementation, modularity, coding practices, repository management, version control practices, frequency and quality of GitHub commits, documentation and overall development contribution	5
Outcomes, Use-Case Relevance & Presentation	Practical outcomes achieved, demonstration of working solution, real-world applicability, innovation, use-case relevance, presentation quality and ability to communicate project outcomes effectively	5
Self-Learning & Additional Exploration	Evidence of independent learning, exploration of additional tools/technologies, online learning resources, certifications, research or implementation beyond classroom discussions	5
Total		**20**

Flexibility in Evaluation Rubrics

While the above rubric framework provides a standard guideline for Capstone Project evaluation across programs and courses, individual faculty members may suitably customize, refine or redefine the evaluation rubrics based on:

- the nature of the course,
- project complexity,
- domain-specific requirements,
- implementation expectations,
- industry relevance and
- learning outcomes of the subject.

Faculty members are encouraged to adopt flexible and outcome-oriented evaluation practices that appropriately assess innovation, practical implementation, technical depth, research orientation, creativity, teamwork and self-learning demonstrated by students during the Capstone Project work

14. Evaluation Scheme

Evaluation Scheme:

Evaluation Components	Weightage
A. Internal Assessments: **1. Continuous Assessment (40 Marks)** **(Minimum 5 components to be used and to be evenly spaced)** Project/ Quizzes/Test/ Assignments and Essays/ Presentations/ Class Participation/ Case Studies/ Reflective Journals/Lab Work	**40 Marks**
2. Mid Term Exam	**20 Marks**
B. External Assessments: End term Examination	**40 Marks**
Total	**100 Marks**

15. Scheme of Studies

Program Name	B.Tech. (Computer Science & Engineering)
Total Credits	165
Total Semesters	8

Credit Distribution Summary

Program Name	I	II	III	IV	V	VI	VII	VIII	Total Credits
B. Tech CSE	21	23	26	24	22	23	14	12	165

Note: For regular 4-credit theory-cum-practical courses having 3-0-2 L-T-P structure, the total academic engagement shall be considered as 120 hours. This includes 45 hours of classroom instruction, 30 hours of laboratory/practical instruction, and 45 hours of Course Capstone Assignment / guided self-learning activities.

Semester I								
Year	**#**	**Course_Code**	**Cat**	**Course Title**	**L**	**T**	**P**	**Cr**
	1	**MAT001**	Major	Probability, Statistics & Linear Algebra	3	0	2	4
	2	**CSE101**	Major	Programming for Problem Solving Using Python	3	0	2	4
	3	**CSE102**	Major	Web Dev I (HTML, CSS, JS Basics)	3	0	2	4
	4	**PHY001**	Major	Engineering Physics	3	0	2	4
	5	**CSE103**	Major	CS Fundamentals & Career Pathways	2	0	2	3
	6	**SEET001**	SEC	Design Thinking & Prototyping	1	0	2	2
	TOTAL				**15**	**0**	**12**	**21**

Semester II							
#	Course_Code	Cat	Course Title	L	T	P	Cr
1	MAT002	Major	Discrete Mathematics	3	0	2	4
2	CSE105	Major	Problem Solving with C++	3	0	2	4
3	CSE106	Major	Web Dev II (Advanced JS & React)	3	0	2	4
4	ECE001	Major	Basics of Electronics & Digital Logic Design	3	0	2	4
5	SEAML001	SEC	AI Fluency & Digital Intelligence	2	0	2	3
6	SEET002	SEC	Maker Lab: Tinkering with Technology	1	0	2	2
7		VAC	Environmental Studies	2	0	0	2
TOTAL				**17**	**0**	**12**	**23**
Summer Internship (6-8 weeks)							

Semester III								
Year	**#**	**Course_Code**	**Cat**	**Course Title**	**L**	**T**	**P**	**Cr**
	1	**CSE201**	Major	Data Structures	3	0	2	4
	2	**CSE202**	Major	Database Management Systems	3	0	2	4
	3	**CSE203**	Major	Computer Organization & System Architecture	3	0	2	4
	4	**CSE204**	Major	Web Dev III (Node.js & Express Backend)	3	0	2	4
	5		OE	OE-I	3	0	0	3
	6		AEC	Foundations of Everyday Communication	3	0	0	3
	7	**SEET003**	SEC	Competitive Coding I	2	0	0	2
	8	**SICSE002**	INT	Summer Internship-I (Assessment)	0	0	4	2
TOTAL					**20**	**0**	**12**	**26**

Semester IV							
#	Course_Code	Cat	Course Title	L	T	P	Cr
1	CSE205	Major	Design & Analysis of Algorithms	3	0	2	4
2	CSE206	Major	Operating Systems	3	0	2	4
3	CSE207	Major	Object-Oriented Programming with Java	3	0	2	4
4		OE	OE-II	3	0	0	3
5		AEC	Professional & Workplace Communication	3	0	0	3
6	SEET004	SEC	Competitive Coding II	2	0	0	2
7	PRCSE002	Proj	Minor Project I	0	0	4	2
8		VAC	Value Added Course II	2	0	0	2
TOTAL				19	0	10	24
Summer Internship (6-8 weeks)							

Semester V								
Year	**#**	**Course_Code**	**Cat**	**Course Title**	**L**	**T**	**P**	**Cr**
	1	**CSE301**	Major	Software Engineering & System Design	3	0	2	4
	2	**CSE302**	Major	Computer Networks	3	0	2	4
	3		OE	OE-III	3	0	0	3
	4		DSE	DSE-I	3	0	2	4
	5		AEC	Advanced Communication and Interview Mastery	3	0	0	3
	6	**SEET005**	SEC	Competitive Coding III	2	0	0	2
	7	**SICSE003**	INT	Summer Internship-II (Assessment)	0	0	4	2
TOTAL					**17**	**0**	**10**	**22**

Discipline Specific Electives-I					
#	**Course Title**	**L**	**T**	**P**	**Cr**
1	Next-Generation Programming Languages	**3**	**0**	**2**	**4**
2	Cyber Security	**3**	**0**	**2**	**4**
3	Deep Learning & Neural Networks	**3**	**0**	**2**	**4**
4	iOS Application Development with Swift & SwiftUI	**3**	**0**	**2**	**4**

Semester VI							
#	Course_Code	Cat	Course Title	L	T	P	Cr
1	CSE303	Major	Essentials of Machine Learning	3	0	2	4
2	CSE304	Major	Cloud Computing & DevOps	3	0	2	4
3	CSE013	Major	Professional Problem Solving & Career Readiness-I	2	0	4	4
4		DSE	DSE-II	3	0	2	4
5		AEC	Arithmetic & Reasoning	3	0	0	3
6		VAC	Value Added Course III	2	0	0	2
7	PRCSE003	Proj	Minor Project II	0	0	4	2
TOTAL				16	0	14	23
Summer Internship (6-8 weeks)							

Discipline Specific Electives-II					
#	Course Title	L	T	P	Cr
1	Distributed Systems	3	0	2	4
2	Compiler Design	3	0	2	4
3	Computer Vision & Image Processing	3	0	2	4
4	Game Development with Unity	3	0	2	4

Semester VII								
Year	**#**	**Course_Code**	**Cat**	**Course Title**	**L**	**T**	**P**	**Cr**
	1	**CSE014**	Major	Professional Problem Solving & Career Readiness-II	2	0	4	4
	2		DSE	DSE-III	3	0	2	4
	3	**PRCSE004**	PROJ	Capstone Project I	0	0	8	4
	4	**SICSE004**	INT	Summer Internship-III (Assessment)	0	0	4	2
				TOTAL	**5**	**0**	**18**	**14**

Discipline Specific Electives-III					
#	**Course Title**	**L**	**T**	**P**	**Cr**
1	Parallel & Concurrent Programming	**3**	**0**	**2**	**4**
2	Natural Language Processing & Generative AI	**3**	**0**	**2**	**4**
3	Introduction to Blockchain Technologies	**3**	**0**	**2**	**4**
4	Robotics Programming & ROS	**3**	**0**	**2**	**4**

Semester VIII							
Year	**#**	**Cat**	**Course Title**	**L**	**T**	**P**	**Cr**
	1	**PROJ**	**Capstone Project II / Internship**	**0**	**0**	**24**	**12**
			TOTAL	**0**	**0**	**24**	**12**
Sem VIII: Students choose one path—Industry Internship (minimum 5–6 months / 20–24 weeks) or Campus-Based Capstone Project. Both options carry 12 credits.							

16. Syllabus

Semester: I

PROBABILITY, STATISTICS & LINEAR ALGEBRA

Program Name	B.Tech. CSE (Core)			
Course Name: Probability, Statistics & Linear Algebra	Course Code	L-T-P	Credits	Contact Hours
	MAT001	3-0-2	4	45
Type of Course:	Major			
Semester:	I (First Semester)			
Pre-requisite(s):	Basic Mathematics (Class XII level – Calculus, Algebra, Statistics)			

Course Description

This course provides students with a practical and application-oriented foundation in Probability, Statistics, and Linear Algebra for modern engineering and computing applications. The course focuses on mathematical modelling, uncertainty analysis, statistical reasoning, matrix computations, and data-driven problem solving relevant to computer science, AI, machine learning, and engineering systems.

Students will learn probability distributions, statistical inference, regression techniques, vector spaces, matrices, eigenvalues, and matrix decompositions while applying these concepts using Python libraries such as NumPy, SciPy, Pandas, and Matplotlib. The course emphasizes computational experimentation, visualization, and practical implementation through Python-based laboratory sessions and mini-projects.

By the end of the course, students will be able to apply mathematical and statistical techniques to analyze datasets, solve engineering problems, and build computational models for modern computing applications.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply probability concepts for uncertainty modelling.	L3 – Apply
CO2	Perform statistical analysis on datasets.	L3 – Apply
CO3	Apply linear algebra techniques for engineering problems.	L3 – Apply
CO4	Implement mathematical computations using Python libraries.	L3 – Apply
CO5	Analyze and visualize mathematical models and datasets.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	–	2	1	–	–	–	–	–	1
CO2	3	3	–	3	2	–	–	–	1	1	1
CO3	3	2	1	2	1	–	–	–	–	–	–
CO4	2	2	1	1	3	–	–	–	1	–	2
CO5	2	3	–	3	3	–	–	–	1	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	2	–	1
CO2	3	3	2	1	2
CO3	3	3	3	1	2
CO4	3	3	2	2	3
CO5	3	3	3	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Probability Theory & Random Variables	No. of Hours: 12

- Introduction to Probability: sample space, events and axioms of probability
- Conditional Probability & Bayes' Theorem
- Independent Events & Total Probability Theorem
- Random Variables: discrete and continuous random variables
- Probability Distributions: Bernoulli, Binomial, Poisson, Uniform, Exponential and Normal distributions
- Mathematical Expectation & Variance
- Cumulative Distribution Function (CDF)
- Central Limit Theorem & Engineering Applications

- Applications in Reliability and Risk Analysis

Unit Number: 2	Statistical Methods & Data Analysis	No. of Hours: 12

- Measures of Central Tendency: mean, median and mode
- Measures of Dispersion: range, variance and standard deviation
- Data Visualization: histograms, box plots and scatter plots
- Hypothesis Testing: null hypothesis, alternative hypothesis and significance level
- z-test & t-test
- Chi-Square Test
- Correlation & Linear Regression
- Introduction to ANOVA
- Statistical Analysis Using Python

Unit Number: 3	Linear Algebra & Matrix Methods	No. of Hours: 11

- Matrices & Matrix Operations
- Rank of Matrix & Row Echelon Form
- Gaussian & Gauss-Jordan Elimination
- Solution of Linear Systems
- Vector Spaces & Linear Independence
- Basis, Dimension & Subspaces
- Linear Transformations & Rank-Nullity Theorem
- Eigenvalues & Eigenvectors
- Orthogonality & Applications in Computing

Unit Number: 4	Advanced Linear Algebra & Computing Applications	No. of Hours: 10

- LU & QR Decomposition
- Singular Value Decomposition (SVD)
- Principal Component Analysis (PCA)
- Iterative Methods: Jacobi and Gauss-Seidel methods
- Graph-Based Matrix Representation
- Applications in Image Compression & Recommendation Systems
- Data Visualization & Model Interpretation
- Python-Based Mathematical Computing Applications

Lab Experiments – CO Mapping Table

S.	Experiment Title	Mapped	Bloom's Level

N		CO(s)	
1	Python for Mathematical Computing	CO4	L3 – Apply
2	Probability Simulation & Random Experiments	CO1, CO4	L3 – Apply
3	Discrete & Continuous Distribution Analysis	CO1, CO4	L3 – Apply
4	Descriptive Statistics & Data Visualization	CO2, CO4	L3 – Apply
5	Hypothesis Testing & Regression Analysis	CO2, CO4	L3 – Apply
6	Matrix Operations & Linear System Solvers	CO3, CO4	L3 – Apply
7	Eigenvalues, SVD & PCA Applications	CO3, CO4, CO5	L4 – Analyze
8	Capstone: Statistical & Linear Algebra Mini Project	CO1–CO5	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Probability Simulation & Random Experiments	**CO1, CO4**
Problem Statement	Implement probability experiments and analyze uncertainty using Python-based simulations.
Sub-Problems	• Simulate coin toss and dice experiments • Verify probability laws using random simulations • Implement Bernoulli and Binomial distributions • Analyze Poisson and Normal distributions • Compute expectation and variance • Visualize probability distributions using charts
Unit 2 Lab Assignment: Statistical Analysis & Visualization	**CO2, CO4**
Problem Statement	Perform statistical analysis and visualization on real-world datasets using Python tools.
Sub-Problems	• Compute mean, median and standard deviation • Analyze skewness and data distribution • Create histograms, box plots and scatter plots • Perform z-test and t-test analysis • Implement correlation and regression analysis • Interpret statistical results and trends
Unit 3 Lab Assignment: Matrix Operations & Linear Algebra	**CO3, CO4**
Problem Statement	Implement matrix computations and solve engineering problems using linear algebra techniques.

Sub-Problems	• Perform matrix addition and multiplication • Compute determinant, rank and inverse of matrices • Solve systems of linear equations • Analyze vector spaces and linear independence • Compute eigenvalues and eigenvectors • Visualize matrix transformations using Python
Unit 4 Lab Assignment: Advanced Matrix Methods & Applications	**CO3, CO4, CO5**
Problem Statement	Apply advanced linear algebra techniques for modern computing and data analysis applications.
Sub-Problems	• Implement LU and QR decomposition • Perform SVD-based matrix analysis • Apply PCA on datasets • Solve systems using iterative methods • Analyze graph-based matrix structures • Visualize and interpret computational results
Capstone Lab Assignment: Statistical & Linear Algebra Mini Project	**CO1, CO2, CO3, CO4, CO5**
Problem Statement	Develop a mini-project integrating probability, statistics and linear algebra techniques for real-world data analysis.
Sub-Problems	• Load and preprocess datasets • Perform probability and statistical analysis • Apply matrix operations and decomposition methods • Visualize trends and analytical results • Interpret model outputs and findings • Present project report and conclusions

Text and Reference Books

Textbooks:

1. Walpole, R. E., Myers, R. H., Myers, S. L. & Ye, K. Probability & Statistics for Engineers and Scientists. Pearson Education.
2. Strang, G. Introduction to Linear Algebra, 5th Edition. Wellesley-Cambridge Press.
3. Ross, S. Introduction to Probability Models. Academic Press.

Recommended Reference Books:

1. Montgomery, D. C. & Runger, G. C. Applied Statistics and Probability for Engineers. Wiley.
2. Lay, D. C. Linear Algebra and Its Applications. Pearson Education.
3. VanderPlas, J. Python Data Science Handbook. O'Reilly Media.

Recommended Online Resources:

- NumPy Documentation — https://numpy.org/doc/
- SciPy Documentation — https://docs.scipy.org/doc/scipy/
- Pandas Documentation — https://pandas.pydata.org/docs/
- Matplotlib Documentation — https://matplotlib.org/stable/users/index.html
- StatQuest — https://www.youtube.com/c/joshstarmer
- Khan Academy Probability & Statistics — https://www.khanacademy.org/math/statistics-probability
- 3Blue1Brown Linear Algebra — https://www.3blue1brown.com/topics/linear-algebra

PROGRAMMING FOR PROBLEM SOLVING USING PYTHON

Program Name	B.Tech. CSE (Core)			
Course Name: Programming for Problem Solving using Python	Course Code	L-T-P	Credits	Contact Hours
	CSE101	3-0-2	4	45
Type of Course:	Major			
Semester:	I (First Semester)			
Pre-requisite(s):	No formal pre-requisite			

Course Description

This course introduces students to the fundamentals of programming using Python through a practical, problem-solving-oriented approach. The course is designed for first-semester learners to build computational thinking, logical reasoning, and structured programming skills using simple and real-world examples. Students learn essential programming constructs, modular programming practices, object-oriented concepts, file handling, and introductory data analysis and visualization techniques.

The course emphasizes hands-on learning through realistic programming problems, mini applications, and data-driven exercises that help students understand how programming can be applied to solve practical problems.

By the end of the course, students will be able to design, develop, debug, and analyze Python programs using foundational programming principles and beginner-level data processing techniques.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply basic Python constructs to solve computational problems.	L3 – Apply
CO2	Develop modular programs using functions and data structures.	L3 – Apply
CO3	Build object-oriented and file-based Python applications with proper error handling.	L3 – Apply
CO4	Analyze and visualize data using Python libraries for real-world insights.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	2	2	1	–	2	–	–	–	–	–	1
CO2	3	3	2	–	2	–	–	–	1	–	2
CO3	3	2	2	–	3	–	–	1	–	1	2
CO4	2	3	–	2	3	–	–	–	1	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	–	1
CO2	3	3	2	1	2
CO3	3	3	2	2	3
CO4	2	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	**Python Fundamentals & Problem-Solving Basics**	**No. of Hours: 12**

- Python environment setup: installation of Python, IDEs and editors, running scripts, Jupyter Notebook basics, introduction to virtual environments (venv)
- Core syntax & data types: variables, identifiers, keywords, numeric data types, strings, boolean values, user input, formatted output, type casting and type conversion
- Operators & expressions: arithmetic, relational, logical, assignment, membership and identity operators, operator precedence and expressions
- Control flow: conditional statements, nested conditions, loops, range function, break, continue and pass statements
- Basic problem-solving patterns: dry run techniques, step-wise problem solving, algorithmic thinking, flow of execution, logical program design

- Introduction to computational thinking: decomposition, pattern recognition, abstraction and simple algorithm design
- Hands-on problems: number-based problems, pattern printing, string manipulation, menu-driven programs and simple mathematical applications

Unit Number: 2	Functions, Data Structures & Coding Patterns	No. of Hours: 12

- Functions & modular coding: function definition, parameters and arguments, return statements, variable scope, recursive functions using simple examples
- Built-in data structures: lists, tuples, sets and dictionaries, indexing, slicing, traversal, insertion, deletion and common operations
- String & list problem solving: string operations, list manipulations, searching, sorting basics, transformations and simple data processing
- Python coding patterns: frequency counting, duplicate detection, palindrome checking, traversal techniques and simple logical patterns
- List comprehensions & modular thinking: concise data processing, nested structures and reusable code design
- Algorithmic thinking: recursion vs iteration, simple optimization techniques, logical decomposition and modular problem solving
- Intermediate programming problems: student record processing, text analysis, data summarization and collection-based applications

Unit Number: 3	Object-Oriented Programming & Real-World Programming	No. of Hours: 11

- Object-oriented programming: classes, objects, constructors, instance variables, methods, encapsulation, inheritance and polymorphism basics
- Advanced Python programming concepts: list and dictionary comprehensions, lambda expressions and introduction to generators
- File handling & data persistence: reading and writing text files, CSV file handling, JSON data handling and structured data storage
- Exception handling: try-except blocks, handling runtime errors, custom exceptions and robust programming practices
- Debugging & code quality: debugging techniques, code tracing, logical testing and clean coding practices
- Modular application development: menu-driven applications, reusable program modules and basic project structuring
- Mini application development: student management systems, inventory management and record maintenance applications

Unit Number: 4	Data Processing & Practical Applications	No. of Hours: 10

- NumPy fundamentals: introduction to arrays, array creation, indexing, slicing and basic vectorized operations
- Pandas for data analysis: DataFrame creation, filtering, sorting, grouping, handling missing values and basic data transformations
- Data visualization: introduction to Matplotlib and Seaborn, line charts, bar charts, pie charts, histograms, scatter plots and chart selection techniques
- Real-world data processing: loading datasets, cleaning and preprocessing data, summarization and trend identification
- Basic analytical applications: analyzing sales, student performance, attendance and survey datasets using Python
- Practical programming applications: mini case studies involving data analysis, visualization and report generation

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Daily Expense Tracker & Analyzer	CO1	L3 – Apply
2	Smart Text & Student Record Processor	CO2	L3 – Apply
3	Student Information Management System	CO3	L3 – Apply
4	Sales Data Analysis & Visualization System	CO4	L4 – Analyze
5	Capstone: Personal Productivity Analytics System	CO1–CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Daily Expense Tracker & Analyzer	**CO1**
Problem Statement	Design a command-line application that helps users record and analyze their daily expenses to understand spending habits and financial patterns.
Sub-Problems	• Accept user inputs for expense category, amount and date • Store and display expense records using appropriate data types • Implement conditional logic to categorize expenses • Calculate total and average expenses • Display expenses above a user-defined threshold • Generate simple summaries and formatted reports • Implement pattern printing and menu-driven interaction
Unit 2 Lab Assignment: Smart	**CO2**

Text & Student Record Processor	
Problem Statement	Develop a Python application that processes textual and student-related data using functions and data structures to generate useful insights.
Sub-Problems	• Input and process student or textual data • Count word frequency using dictionaries • Search and sort records using lists • Remove duplicate entries using sets • Perform palindrome and string analysis operations • Create reusable functions for each task • Generate summary reports and structured outputs
Unit 3 Lab Assignment: Student Information Management System	**CO3**
Problem Statement	Build a modular student information system that manages academic records using object-oriented programming and file handling techniques.
Sub-Problems	• Create a Student class with attributes and methods • Add, update, search and delete student records • Store records using text, CSV or JSON files • Handle invalid inputs using exception handling • Generate performance summaries and reports • Organize the project into reusable modules • Debug and test application functionality
Unit 4 Lab Assignment: Sales Data Analysis & Visualization System	**CO4**
Problem Statement	Create a data analysis application that processes real-world datasets and visualizes meaningful insights using Python libraries.
Sub-Problems	• Load datasets using Pandas • Clean and preprocess missing or inconsistent data • Perform filtering, sorting and grouping operations • Analyze trends and generate summaries • Create charts using Matplotlib and Seaborn • Compare different visualization techniques • Generate graphical reports and insights
Capstone Lab Assignment: Personal Productivity Analytics System	**CO1, CO2, CO3, CO4**

Problem Statement	Design an integrated Python-based productivity management system that helps users manage tasks, analyze productivity trends and generate visual insights.
Sub-Problems	• Create task management functionalities • Categorize and prioritize tasks • Store task records using files • Analyze productivity patterns using Pandas • Generate summaries and productivity scores • Visualize task completion trends using charts • Develop a modular and menu-driven application • Implement exception handling and clean program structure

Text and Reference Books

Textbooks:

1. Matthes, E. (2023). Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 3rd Edition. No Starch Press. ISBN: 978-1-71850-264-2.
2. Sweigart, A. (2019). Automate the Boring Stuff with Python: Practical Programming for Total Beginners, 2nd Edition. No Starch Press.

Recommended Reference Books:

1. **Al Sweigart,** *Automate the Boring Stuff with Python,* 2nd Edition, No Starch Press, 2019. , ISBN: 9781593279929
2. Zelle, J. (2016). Python Programming: An Introduction to Computer Science, 3rd Edition. Franklin, Beedle & Associates. ISBN: 9781590282755

Recommended Online Resources:

- Python Official Documentation — https://docs.python.org/3/
- W3Schools Python Tutorial — https://www.w3schools.com/python/
- Kaggle Learn Python — https://www.kaggle.com/learn/python
- Google's Python Class — https://developers.google.com/edu/python
- freeCodeCamp Python Course — https://www.freecodecamp.org/learn/scientific-computing-with-python/

WEB DEVELOPMENT – I (HTML, CSS, JAVASCRIPT BASICS)

Program Name	B.Tech. CSE (Core)			
Course Name: Web Development – I	Course Code	L-T-P	Credits	Contact Hours
	CSE102	3-0-2	4	45
Type of Course:	Major			
Semester:	I (First Semester)			
Pre-requisite(s):	No formal pre-requisite; basic computer literacy required			

Course Description

This beginner-friendly course introduces students to the fundamentals of front-end web development using HTML5, CSS3, and core JavaScript concepts. The course is designed for first-semester learners to build a strong foundation in creating structured, responsive, and visually appealing web pages while developing logical problem-solving skills using JavaScript.

Students begin by learning semantic HTML for creating clean and accessible web page structures. They then explore CSS for styling, layouts, responsive design, Flexbox, Grid, and basic animations to create polished user interfaces. In JavaScript, students focus on programming fundamentals such as variables, functions, loops, arrays, objects, and control flow to develop logical thinking and interactive behavior.

The course emphasizes hands-on learning through practical web-based projects, mini applications, and responsive design exercises using VS Code and browser developer tools.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply semantic HTML concepts to build well-structured and accessible web pages.	L3 – Apply
CO2	Develop visually consistent and responsive web interfaces using CSS styling techniques.	L3 – Apply
CO3	Apply core JavaScript programming concepts such as variables, functions, arrays, objects, and control structures to solve programming problems.	L3 – Apply
CO4	Analyze the integration of HTML, CSS, and JavaScript to create responsive and interactive front-end solutions.	L4 – Analyze
CO5	Design and develop a complete responsive web project demonstrating front-end structure, styling, and functionality.	L6 – Create

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	1	–	2	–	2	–	1	–	–	2	1
CO2	1	–	2	–	3	–	–	–	–	2	1
CO3	2	2	1	–	2	–	–	–	–	–	1
CO4	2	2	3	–	3	–	–	–	1	2	2
CO5	–	2	3	–	3	–	1	1	2	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	–	1
CO2	3	3	2	1	2
CO3	3	3	2	1	2
CO4	3	3	3	2	2
CO5	3	3	3	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	HTML & Web Foundation	No. of Hours: 12

- Introduction & environment setup: understanding the role of HTML in web development, relationship between HTML, CSS, and JavaScript, setting up VS Code, Live Server, browser tools, and HTML validators
- Basic HTML structure & common tags: HTML document structure, headings, paragraphs, line breaks, horizontal rules, comments, and text organization techniques
- Links, images & multimedia: hyperlinks for navigation, internal and external links, image insertion with responsive images, embedding audio and video using HTML5 multimedia tags
- Lists, tables & semantic HTML: ordered and unordered lists, nested lists, table creation using rows and columns, semantic tags such as <header>, <footer>, <section>, and <nav>
- Forms & user input handling: form creation using <form>, <input>, <textarea>, <select>, and <button>, use of labels, placeholders, required validation, and accessibility-focused form structure

- Practical web page development: building mini web pages such as profile pages, recipe cards, signup forms, and image galleries using proper semantic structure

Unit Number: 2	CSS Core Styling & Layouts	No. of Hours: 12

- Introduction to CSS & styling workflow: understanding the role of CSS, connecting stylesheets, browser developer tools, clean styling practices, and debugging techniques
- Types of CSS: inline, internal, and external CSS, their advantages, disadvantages, and maintainability considerations
- CSS colors & typography: color systems using HEX, RGB, and HSL, custom typography using Google Fonts and @font-face, font styling and text formatting
- CSS selectors & styling application: element, class, ID, attribute, and combinator selectors, specificity basics, applying styles efficiently to HTML elements
- Box model & spacing: understanding content, padding, border, and margin, spacing techniques, overflow handling, and visual alignment
- Positioning & layouts: static, relative, absolute, fixed, and sticky positioning, layout creation using columns, image galleries, centering techniques, and reusable styling structures
- Practical implementation: styling portfolio pages, forms, tables, navigation bars, buttons, and visual components using structured CSS design principles

Unit Number: 3	Advanced CSS & Responsive Design	No. of Hours: 11

- Display properties & layout fundamentals: understanding inline, block, inline-block, flex, and grid display properties for modern layouts
- Flexbox fundamentals: flex containers, flex items, justify-content, align-items, flex-direction, flex-wrap, and responsive alignment techniques
- CSS Grid fundamentals: defining rows and columns, grid-template-areas, grid-gap, placing items using grid lines, and structured responsive layouts
- Responsive design principles: mobile-first design, media queries, viewport-based layouts, and adaptive user interfaces for multiple screen sizes
- CSS units & scalable design: relative units (rem, em, %, vw, vh) and absolute units (px), scalable typography and spacing strategies
- CSS effects & animations: transitions, transforms, hover effects, keyframe animations, and interactive UI enhancement techniques
- Practical implementation: building responsive portfolio layouts, animated navigation systems, responsive grids, and polished user interfaces using Flexbox and Grid

Unit Number: 4	JavaScript Essentials	No. of Hours: 10

- Introduction & environment setup: understanding JavaScript in web development, using browser console, linking JavaScript with HTML using internal and external scripts
- Variables & data types: var, let, const, primitive and reference data types, type conversion, template

literals, and variable scope basics

- Core methods & operators: string methods, number methods, Math object usage, arithmetic and logical operators, comparison operators, ternary operator
- Control flow & loops: conditional statements using if, else, switch, loops including for, while, do-while, and for...of, along with break and continue statements
- Functions & scope: function declaration, expression, arrow functions, anonymous functions, parameters, return values, default parameters, closures, and scope concepts
- Arrays & array methods: array creation, traversal, CRUD operations, higher-order methods such as map(), filter(), reduce(), sorting, destructuring, and spread operator
- Objects & object handling: object creation, property access, CRUD operations, object methods, destructuring, spread operator, shallow vs deep copy, and iteration
- • Practical JavaScript applications: integrating JavaScript with HTML/CSS projects to add dynamic behavior, user interaction, validation logic, and simple interactive features

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Personal Portfolio Website (HTML)	CO1	L3 – Apply
2	Personal Portfolio Website (CSS Core)	CO2	L3 – Apply
3	Responsive Personal Portfolio Website (CSS Advanced)	CO2, CO4	L4 – Analyze
4	Prompt Quizzer – A Quiz Game	CO3	L3 – Apply
5	Capstone: Responsive Website Project	CO1–CO5	L6 – Create

Lab Assignments

Unit 1 Lab Assignment: Personal Portfolio Website (HTML)	**CO1**
Problem Statement	Build a one-page portfolio website using semantic HTML tags to showcase personal information, projects, skills, and contact details while following accessibility and clean structural design practices.
Sub-Problems	• Build a one-page portfolio using semantic tags such as <header>, <nav>, <section>, and <footer> • Add navigation links to About, Projects, and Contact sections • Create a hero section with title and introduction • Add a profile image with proper alt text • Display projects using lists or article elements • Use a table to showcase technical skills or tools • Build a contact form with name, email, and message fields

	• Apply labels, placeholders, and required validation in forms • Add a skip navigation link for accessibility • Structure HTML cleanly for future CSS styling
Unit 2 Lab Assignment: Personal Portfolio Website (CSS Core)	**CO2**
Problem Statement	Create a visually styled portfolio webpage using external CSS stylesheets, typography customization, spacing, positioning, and consistent design principles.
Sub-Problems	• Apply external CSS stylesheet to the HTML portfolio • Use element, class, and ID selectors for styling • Create a consistent color theme using HEX or RGB values • Add custom fonts using Google Fonts or @font-face • Apply box model properties for spacing and layout • Style the navigation bar and hyperlinks • Use positioning methods for sticky headers or buttons • Style tables, forms, and buttons for better UI presentation • Add visual separators using borders and horizontal rules • Organize CSS cleanly for maintainability and debugging
Unit 3 Lab Assignment: Responsive Personal Portfolio Website (CSS Advanced)	**CO2, CO4**
Problem Statement	Design and develop a responsive portfolio website using modern CSS layout techniques including Flexbox, Grid, media queries, transitions, and animations.
Sub-Problems	• Use Flexbox for alignment and responsive layouts • Create at least one structured section using CSS Grid • Implement mobile-first responsive design using media queries • Use relative CSS units for scalable layouts and typography • Add hover effects with transitions on links and buttons • Apply CSS transforms and animations for interactivity • Create keyframe animations for visual enhancement • Ensure layouts adapt across mobile, tablet, and desktop devices • Build polished UI sections with modern responsive design principles • Optimize spacing, alignment, and user experience
Unit 4 Lab Assignment: Prompt	**CO3**

Quizzer – A Quiz Game	
Problem Statement	Develop a console-based JavaScript quiz game using prompt() and alert() to practice programming logic, loops, arrays, functions, and user interaction.
Sub-Problems	• Create predefined quiz questions using arrays or objects • Accept user responses using prompt() • Validate answers using trim() and toLowerCase() • Use loops to navigate through questions • Display feedback using alert() • Track user score throughout the quiz • Display final score and performance summary • Organize quiz logic using reusable functions • Practice conditional statements and control flow • Run the program directly in the browser console
Capstone Lab Assignment: Responsive Website Project	**CO1, CO2, CO3, CO4, CO5**
Problem Statement	Build a complete responsive web interface using HTML5 and CSS3 to demonstrate semantic structure, responsive layouts, modern UI styling, and real-world web design practices.
Sub-Problems	• Create semantic HTML structure for the selected project theme (Online Learning Platform / E-commerce / Travel Agency / Food Delivery App, etc.) • Build responsive layouts using Flexbox and CSS Grid • Add navigation menus, forms, images, and content sections • Apply media queries for mobile, tablet, and desktop responsiveness • Design visually consistent UI using typography and color schemes • Add hover effects, transitions, and polished styling • Organize layouts with reusable CSS structures • Build responsive cards, grids, and interactive sections • Ensure accessibility and proper content organization • Deliver a complete polished static front-end project

Text and Reference Books

Textbooks:

1. Robbins, J. N. (2018). Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics, 5th Edition. O'Reilly Media. ISBN: 978-1491960202
2. Castro, E. & Hyslop, B. (2021). HTML & CSS: Visual QuickStart Guide, 9th Edition. Peachpit Press (Pearson). ISBN: 978-0136581444

Recommended Reference Books:

1. Duckett, J. (2011). HTML and CSS: Design and Build Websites. Wiley Publications. ISBN: 9781118008188
2. Duckett, J. (2014). JavaScript and JQuery: Interactive Front-End Web Development. Wiley Publications. ISBN: 9781118531648
3. Flanagan, D. (2020). JavaScript: The Definitive Guide, 7th Edition. O'Reilly Media. ISBN: 9781491952023

Recommended Online Resources:

- MDN Web Docs — https://developer.mozilla.org/
- W3Schools HTML, CSS & JavaScript Tutorials — https://www.w3schools.com/
- freeCodeCamp Web Development Curriculum — https://www.freecodecamp.org/
- CSS Tricks — https://css-tricks.com/
- JavaScript.info — https://javascript.info/
- Google Web Fundamentals — https://web.dev/

K.R. MANGALAM UNIVERSITY

ENGINEERING PHYSICS

Program Name	B.Tech. CSE (Core)			
Course Name: Engineering Physics	Course Code	L-T-P	Credits	Contact Hours
	PHY001	3-0-2	4	45
Type of Course:	Major			
Semester:	I (First Semester)			
Pre-requisite(s):	Basic conceptual knowledge of Physics			

Course Description

This course introduces fundamental physical principles underlying modern computing, communication, sensing, and emerging technologies. The emphasis is on semiconductor materials, optical systems, lasers, fiber-optic communication, nanomaterials, and advanced sensing technologies used in computing infrastructure, data centers, communication networks, storage devices, and intelligent systems. The course adopts an application-oriented approach suitable for Computer Science and Engineering students, minimizing overlap with Electronics and Digital Logic Design courses while strengthening interdisciplinary understanding of technology-enabling physics.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Explain the physical principles of semiconductor materials, nanomaterials, and optical systems used in modern computing technologies.	L2 – Understand
CO2	Analyze the role of lasers, optical fibers, and photonic devices in communication and information processing systems.	L4 – Analyze
CO3	Apply concepts of advanced materials and sensors in emerging computing and intelligent systems.	L3 – Apply
CO4	Evaluate contemporary technological solutions based on optics, photonics, and semiconductor materials for real-world engineering applications.	L5 – Evaluate

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	-	-	-	-	-	-	1	-
CO2	3	3	2	1	-	-	-	-	-	1	-
CO3	2	3	3	2	1	-	-	-	-	1	-

CO4	2	3	3	3	2	-	-	-	-	2	1

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	-	-
CO2	3	3	2	1	-
CO3	2	3	3	2	1
CO4	2	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	**Semiconductor Materials and Nanotechnology for Computing**	**No. of Hours: 12**

Semiconductor Materials

- Energy band theory
- Conductors, Semiconductors and Insulators
- Intrinsic and Extrinsic Semiconductors
- Carrier concentration and Fermi level concept
- Hall Effect and its applications

Nanotechnology

- Introduction to nanomaterials
- Size-dependent properties
- Carbon Nanotubes (CNTs)
- Graphene and its applications
- Quantum Dots
- Applications in processors, memory devices, sensors and displays

Computing Applications

- Semiconductor materials in CPUs, SSDs and memory systems
- Nanotechnology in next-generation computing devices

(Note: PN Junction, Diode Characteristics, BJT, MOSFET, Biasing removed due to overlap with Electronics.)

Unit Number: 2	Wave Optics and Optical Storage Technologies	No. of Hours: 11

Interference

- Principle of superposition
- Interference of light
- Young's Double Slit Experiment
- Applications in precision measurements

Diffraction

- Fraunhofer diffraction
- Diffraction Grating
- Resolving power

Optical Storage Technologies

- CD, DVD and Blu-ray technology
- Optical data storage mechanisms
- Emerging holographic storage systems

Holography

- Principle of holography
- Recording and reconstruction

Applications in data security and 3D imaging

Unit Number: 3	Optical Physics in Modern Computing	No. of Hours: 11

Laser Fundamentals

- Spontaneous and stimulated emission
- Population inversion
- Einstein coefficients
- Characteristics of laser

Types of Lasers

- He-Ne Laser
- Semiconductor Laser

- Fiber Laser

Applications

- Optical storage
- Laser printers
- Barcode scanners
- LiDAR systems
- Medical and industrial applications

Photonics

- Introduction to integrated photonics
- Photonic computing
- Silicon photonics
- Future of optical computing

Unit Number: 4	Optical Communication and Fiber Optics	No. of Hours: 11

Optical Fibers

- Total Internal Reflection
- Structure and working of optical fiber
- Numerical Aperture
- Acceptance angle
- Fiber losses

Optical Communication

- Fiber optic communication systems
- Data transmission in internet infrastructure
- Data centers and cloud computing networks

Sensors and Intelligent Systems

- Optical sensors
- Biosensors
- MEMS sensors
- Fingerprint and biometric sensors

- IoT sensing technologies

Emerging Trends

- Quantum communication
- Quantum computing (introductory concepts)
- Smart materials and intelligent devices

Lab Experiments – CO Mapping Table

Group-A: Optics and Lasers (Experiments 1–6)

Group-B: Fiber Optics and Sensors (Experiments 7–10)

Group-C: Semiconductor Materials – Any Two as Open Experiments

- Hall Effect and determination of Hall coefficient.
- Four Probe method for measurement of semiconductor resistivity.
- Temperature dependence of semiconductor conductivity.
- Study of photovoltaic solar cell characteristics.

S.N	Experiment Title	Mapped CO(s)	Bloom's Level
Group-A: Optics and Lasers			
1	Determination of wavelength of sodium light using Newton's Rings	CO1, CO2	L3 – Apply
2	Determination of wavelength of mercury spectrum using diffraction grating	CO1, CO2	L3 – Apply
3	Determination of refractive index of prism using spectrometer	CO1, CO2	L3 – Apply
4	Study of optical activity using polarimeter	CO1, CO2	L3 – Apply
5	Determination of wavelength of He-Ne laser using diffraction grating	CO2, CO3	L3 – Apply
6	Study of laser beam divergence and intensity distribution	CO2, CO3	L3 – Apply
Group-B: Fiber Optics and Sensors			
7	Determination of Numerical Aperture of Optical Fiber	CO2, CO3	L3 – Apply

8	Study of attenuation losses in optical fiber	CO2, CO3	L4 – Analyze
9	Characterization of fiber optic communication link	CO2, CO3	L4 – Analyze
10	Study of optical sensor / fiber optic displacement sensor	CO3, CO4	L4 – Analyze
Group-C: Semiconductor Materials (Any Two as Open Experiments)			
11	Hall Effect and determination of Hall coefficient	CO1, CO3	L4 – Analyze
12	Four Probe method for measurement of semiconductor resistivity	CO1, CO3	L3 – Apply
13	Temperature dependence of semiconductor conductivity	CO1, CO3	L4 – Analyze
14	Study of photovoltaic solar cell characteristics	CO3, CO4	L4 – Analyze

Text and Reference Books

Textbooks:

1. B.K. Pandey and S. Chaturvedi, Engineering Physics, Cengage Learning India Pvt. Ltd., 2023/2024 Edition. ISBN: 978-9386668525
2. H.K. Malik and A.K. Singh, Engineering Physics, McGraw Hill Education, 2nd Edition, 2017. ISBN: 978-0070704558

Recommended Reference Books:

3. M.N. Avadhanulu and P.G. Kshirsagar, A Textbook of Engineering Physics, S. Chand & Company, 10th Edition, 2014. ISBN: 978-8121908375
4. R.K. Gaur and S.L. Gupta, Engineering Physics, Dhanpat Rai Publications, 8th Edition, 2013. ISBN: 978-8189928100
5. C. Kittel, Introduction to Solid State Physics, Wiley India, 8th Edition, 2005. ISBN: 978-8126508181
6. Eugene Hecht, Optics, Pearson Education, 5th Edition, 2017. ISBN: 978-0133977226

Recommended Online Resources:

- NPTEL – Engineering Physics — https://nptel.ac.in/courses/115/103/115103111/
- MIT OpenCourseWare – Physics — https://ocw.mit.edu/courses/physics/
- LLNL Laser Science Resources — https://lasers.llnl.gov/education
- Fiber Optic Association — https://www.thefoa.org
- Nanoscience & Nanotechnology Resources — https://www.nano.gov
- PhET Interactive Simulations — https://phet.colorado.edu

COMPUTER SCIENCE FUNDAMENTALS & CAREER PATHWAYS

Program Name	B.Tech. CSE (Core)			
Course Name: Computer Science Fundamentals & Career Pathways	Course Code	L-T-P	Credits	Contact Hours
	CSE103	2-0-2	3	30
Type of Course:	Major			
Semester:	I (First Semester)			
Pre-requisite(s):	No formal pre-requisite			

Course Description

This course introduces first-semester B.Tech CSE students to the foundational concepts of modern computer science through a highly practical and industry-oriented approach. The course helps students understand how computers work internally, how software is executed, how operating systems manage resources, and how modern applications interact with computing systems and networks.

Students will develop computational thinking and systems-level understanding while simultaneously gaining hands-on exposure to Linux environments, developer workflows, version control systems, internet technologies, APIs, and modern developer productivity tools. The course also introduces students to emerging computing domains such as Artificial Intelligence, Data Science, Cybersecurity, Cloud Computing, and Software Engineering from an application-oriented perspective.

The course emphasizes experiential learning through practical activities, command-line workflows, debugging exercises, automation tasks, system exploration, GitHub collaboration, API interaction, and portfolio-oriented development practices. By the end of the course, students will possess a strong foundational understanding of computing systems and modern developer ecosystems required for future computer science learning.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply computational thinking and computing fundamentals to solve problems.	L3 – Apply
CO2	Use Linux tools and shell scripting for system operations and automation.	L3 – Apply
CO3	Implement Git, GitHub, APIs and documentation workflows.	L3 – Apply
CO4	Analyze computing domains and build a beginner developer portfolio.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	1	–	1	–	–	–	–	–	2
CO2	2	2	–	–	3	–	–	–	–	–	2
CO3	–	–	2	–	3	–	–	2	2	2	2
CO4	–	2	–	–	2	1	1	2	1	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	2	–	1
CO2	3	3	2	1	2
CO3	3	3	3	2	3
CO4	2	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Foundations of Computing & Computational Thinking	No. of Hours: 8

- Introduction to Computing Systems: hardware, software and characteristics of modern computer systems
- How Computers Work: CPU, memory, storage and input/output devices
- Binary & Data Representation: binary number system, bits, bytes and basic data representation
- Program Execution Basics: source code, compilation vs interpretation and execution flow
- Role of Operating Systems: files, processes, memory and user interaction
- Computational Thinking: abstraction, decomposition, pattern recognition and logical problem solving
- Algorithms, Pseudocode & Flowcharts: designing step-wise solutions for practical problems
- Introduction to Computing Domains: software development, AI & data science, cybersecurity and cloud computing

Unit Number: 2	Linux Environment & Command-Line Computing	No. of Hours: 8

- Linux Basics: kernel, shell, terminal and filesystem structure
- File System Operations: file and directory management using terminal
- Permissions & Process Basics: users, permissions and process monitoring
- Command-Line Utilities: grep, sort, uniq, wc, pipes and redirection
- Shell Scripting Fundamentals: variables, loops and conditional statements
- Task Automation Using Shell Scripts
- Developer Environment Setup: VS Code and terminal integration
- Debugging & Troubleshooting Basics: reading errors and basic debugging practices

Unit Number: 3	Internet Technologies, Git & Developer Workflow	No. of Hours: 7

- Computer Network Basics: internet, client-server model and web communication
- How the Internet Works: DNS, IP addresses and HTTP/HTTPS basics
- Introduction to APIs & JSON Data
- Git Fundamentals: repositories, commits, logs and branches
- GitHub Collaboration: push/pull operations and repository management
- Documentation Using Markdown & README Files
- Debugging & Error Analysis: syntax, runtime and logical errors
- Introduction to AI-Assisted Developer Tools: responsible usage of generative AI for coding and learning support

Unit Number: 4	Modern Computing Domains & Professional Development	No. of Hours: 7

- Overview of Modern Computing Domains: software engineering, AI & machine learning, cybersecurity, cloud & DevOps
- Digital Professionalism & Cybersecurity Awareness: passwords, phishing, digital footprint and safe online practices
- Developer Portfolio Building: GitHub profile, resume and LinkedIn basics
- Introduction to Coding & Learning Platforms: HackerRank, LeetCode and GitHub
- Structured Learning Roadmap: semester-wise technical skill planning
- Professional Communication & Technical Presentation Practices

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Computer System Exploration & Binary Representation	CO1	L3 – Apply
2	Algorithm & Flowchart Design	CO1	L3 – Apply
3	Linux File System & Command-Line Operations	CO2	L3 – Apply
4	Linux Text Processing & Shell Scripting	CO2	L3 – Apply
5	Git Version Control & GitHub Repository Management	CO3	L3 – Apply

6	API Exploration & JSON Data Analysis	CO3	L3 – Apply
7	Developer Portfolio & Professional Profile Setup	CO4	L4 – Analyze
8	Capstone: Beginner Developer Ecosystem Portfolio	CO1–CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Computer System Exploration & Binary Representation	**CO1**
Problem Statement	Explore the basic working principles of computer systems and understand how data is represented and processed inside a computer.
Sub-Problems	• Identify CPU, memory and storage specifications • Perform binary and decimal conversions • Observe running processes and memory usage • Analyze files and storage organization • Design simple algorithms and flowcharts • Compare compilation and interpretation concepts
Unit 2 Lab Assignment: Linux Environment & Shell Scripting	**CO2**
Problem Statement	Perform Linux-based system operations and automate basic computing tasks using shell scripting.
Sub-Problems	• Navigate and manage files using terminal • Use Linux text-processing commands • Monitor processes and permissions • Create shell scripts using variables and loops • Automate simple file operations • Debug shell script errors
Unit 3 Lab Assignment: GitHub Collaboration & API Interaction	**CO3**
Problem Statement	Implement modern developer workflows using Git, GitHub and APIs.
Sub-Problems	• Create and manage Git repositories • Track and commit changes • Push projects to GitHub • Create Markdown documentation • Access public APIs and analyze JSON data • Debug common programming and workflow errors
Unit 4 Lab Assignment: Developer Portfolio & Career Roadmap	**CO4**

Problem Statement	Build a beginner-level developer portfolio and structured technical learning roadmap.
Sub-Problems	• Create GitHub and LinkedIn profiles • Prepare technical resume • Organize project repositories • Explore coding platforms • Design semester-wise learning roadmap • Present portfolio professionally
Capstone Lab Assignment: Beginner Developer Ecosystem Portfolio	**CO1, CO2, CO3, CO4**
Problem Statement	Build an integrated beginner developer portfolio combining Linux workflows, GitHub repositories, documentation and technical learning roadmap.
Sub-Problems	• Perform Linux-based practical tasks • Create and manage GitHub repositories • Document projects using Markdown • Access and analyze API data • Organize developer portfolio • Present structured learning roadmap

Text and Reference Books

Textbooks:

1. Brookshear, J. G. & Brylow, D. (2021). Computer Science: An Overview, 13th Edition. Pearson Education.
2. Shotts, W. E. Jr. (2019). The Linux Command Line: A Complete Introduction, 2nd Edition. No Starch Press.

Recommended Reference Books:

1. Petzold, C. Code: The Hidden Language of Computer Hardware and Software. Microsoft Press.
2. Ward, B. How Linux Works, 3rd Edition. No Starch Press.
3. Chacon, S. & Straub, B. Pro Git, 2nd Edition. Apress.

Recommended Online Resources:

- Linux Journey — https://linuxjourney.com/
- GitHub Skills — https://skills.github.com/
- MDN Web Docs — https://developer.mozilla.org/
- freeCodeCamp — https://www.freecodecamp.org/
- HackerRank — https://www.hackerrank.com/
- LeetCode — https://leetcode.com/
- Postman API Learning Center — https://learning.postman.com/
- Visual Studio Code Documentation — https://code.visualstudio.com/docs

DESIGN THINKING & PROTOTYPING

Program Name	B.Tech. CSE (Core)			
Course Name: Design Thinking & Prototyping	Course Code	L-T-P	Credits	Contact Hours
	SEET001	1-0-2	2	30
Type of Course:	SEC			
Semester:	I (First Semester)			
Pre-requisite(s):	No formal pre-requisite			

Course Description

This course introduces first-year CSE students to human-centered design thinking and digital prototyping. Through hands-on studio sessions and guided lab work, students will learn to identify real problems, generate creative solutions, and build clickable digital prototypes of their ideas using industry-standard tools — preparing them to design and communicate software project concepts from their very first semester.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Identify user needs and frame design problems using empathy-driven research techniques	L2 – Understand
CO2	Generate and evaluate solution ideas using structured ideation methods	L3 – Apply
CO3	Build low-fidelity and mid-fidelity digital prototypes using Figma	L3 – Apply
CO4	Conduct usability evaluations and iterate prototypes based on structured feedback	L4 – Analyze
CO5	Present design solutions with clear storytelling and documented design rationale	L3 – Apply

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	1	3	–	2	1	2	1	1	–	2	2
CO2	–	3	2	2	1	1	–	–	2	1	2
CO3	–	2	3	–	3	–	–	–	2	1	2

CO4	–	2	2	3	2	1	–	1	2	2	2
CO5	–	1	2	–	2	–	–	2	2	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	2	–	1
CO2	3	3	2	1	2
CO3	3	3	3	2	3
CO4	2	3	3	2	2
CO5	2	3	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	**Empathy & Problem Framing**	**No. of Hours: 8**

- What is Design Thinking? — the five stages, IDEO Shopping Cart case study, why it matters for software engineers
- Empathy techniques — user interviews (structured vs. unstructured), observation, contextual inquiry
- Empathy Maps — capturing thoughts, feelings, actions, and pain points
- User Personas — demographics, goals, behaviors, pain points
- Problem framing — Point-of-View (POV) statements and How Might We (HMW) questions
- Introduction to User Stories — "As a [user], I want to [action] so that [goal]" format as a bridge from empathy to design

Lab Activities:

IDEO Shopping Cart analysis and discussion workshop

Conduct 2–3 user interviews on a chosen campus problem

Build one empathy map and one user persona per team

Write 3 HMW questions and 3 user stories derived from research

Unit Number: 2	Ideation, User Flow & Wireframing	No. of Hours: 8

- Ideation strategies — brainstorming, SCAMPER, Crazy 8s
- Idea evaluation — feasibility, desirability, viability framework
- User flow diagrams — mapping the steps a user takes through a solution *(simple 5–7 step flows)*
- Information architecture basics — organizing screens and navigation logically
- Paper wireframing — sketching screens, basic UI components (buttons, inputs, cards, navigation bars)
- Structured peer feedback — "I like / I wish / What if" critique format

Lab Activities:

Team Crazy 8s session — generate 8 ideas in 8 minutes, select top 2

Draw a user flow diagram for the chosen solution (pen-and-paper or Miro)

Create a paper wireframe of 3–4 key screens

Conduct one peer feedback session using the critique format

Unit Number: 3	Figma-Based Digital Prototyping	No. of Hours: 7

- Figma interface orientation — frames, layers, components, and assets panel
- Building screens — text, shapes, images, icons, basic layout
- Auto-layout fundamentals — creating responsive, structured UI elements *(kept simple: rows, columns, padding)*
- Navigation and interaction — linking frames to create a clickable prototype flow
- Component basics — creating and reusing a button, card, or input field *(one reusable component per team)*
- Accessibility basics — readable font sizes, sufficient color contrast, clear tap targets
- What makes a good prototype — fidelity levels and when each is appropriate

Lab Activities:

Guided Figma tutorial exercise — recreate a given 2-screen UI from a reference

Convert paper wireframes into 4–6 screen Figma prototype with clickable navigation

Apply one reusable component across all screens

Mid-course pitch — present problem, persona, user flow, and prototype to peers (5 minutes per team)

Unit Number: 4	Usability Testing, Iteration & Final Presentation	No. of Hours: 7

- Usability testing basics — setting objectives, writing task prompts, think-aloud protocol
- Nielsen's 10 Usability Heuristics — simplified and applied to student prototypes
- SUS (System Usability Scale) — administering and interpreting the 10-question usability score *(lightweight, standardized)*

- Iterative design — prioritizing feedback, making targeted improvements
- Design rationale documentation — recording *why* design decisions were made
- Presentation and storytelling — problem → user → solution → prototype → learnings structure

Lab Activities:

Conduct a usability test with 2 peers using a task script and think-aloud method

Collect SUS scores and identify top 3 issues from feedback

Iterate the Figma prototype — document before/after changes with rationale

Final showcase — 8-minute team presentation + 2-minute Q&A with faculty panel

Team Structure & Individual Accountability

Team size: 3–4 students per team, formed in Week 1

Role rotation across units:

Unit	Role A	Role B	Role C	Role D
Unit 1	Lead Researcher	Note-taker	Synthesizer	Presenter
Unit 2	Ideation Lead	Flow Designer	Wireframer	Critic
Unit 3	Figma Lead	Component Designer	Interaction Designer	Tester
Unit 4	Test Facilitator	Data Recorder	Iteration Lead	Presenter

Every student submits an individual reflection journal (300–400 words) at the end of each unit documenting their contribution and learning.

Evaluation Scheme

Component	Weight	What's Assessed
Studio Participation & Weekly Activities	15%	Engagement, in-class work quality, peer collaboration
Unit Reflection Journals (individual)	15%	Personal learning, contribution, critical thinking
Midterm Pitch (Unit 3 milestone)	20%	Problem clarity, user flow, initial Figma prototype

Final Prototype (team)	30%	Fidelity, interactivity, iteration evidence, accessibility
Final Presentation & Viva (individual + team)	20%	Storytelling, rationale, response to questions

Weekly Plan (14 weeks)

Week	Unit	Focus
1	1	Intro to Design Thinking, team formation, case study
2	1	User interviews and observation
3	1	Empathy maps, personas, POV + HMW + user stories
4	2	Ideation — brainstorming, SCAMPER, Crazy 8s
5	2	User flows and information architecture
6	2	Paper wireframing + peer critique
7	3	Figma orientation — guided tutorial
8	3	Building screens, auto-layout basics
9	3	Components, interactions, clickable prototype
10	3	Mid-course pitch
11	4	Usability testing — task writing, think-aloud, SUS
12	4	Analyzing results, planning iterations
13	4	Prototype iteration + design rationale documentation
14	4	Final showcase & viva

Recommended Resources

Core References:

1. Stanford d.school Bootcamp Bootleg (free PDF — primary reading)

2. IDEO Design Kit — designkit.org

3. Lean UX — Jeff Gothelf (replaces Norman as the primary book — directly relevant to software product design)

Supplementary Reading:

1. Don Norman, The Design of Everyday Things (conceptual depth, recommended for self-paced reading)

2. Steve Krug, Don't Make Me Think (UX intuition, excellent companion for Unit 4)

Figma Learning:

1. Figma Education Resources — figma.com

2. Figma YouTube Channel — official tutorials

3. FreeCodeCamp Figma Crash Course (YouTube)

Textbook:

1. The Design Thinking Toolbox — Lewrick, Link & Leifer, Wiley, 2020. ISBN: 978-1119629191

SEMESTER: II

DISCRETE MATHEMATICS

Program Name	B.Tech. CSE (Core)			
Course Name: Discrete Mathematics	Course Code	L-T-P	Credits	Contact Hours
	MAT002	3-0-2	4	45
Type of Course:	Major			
Semester:	II (Second Semester)			
Pre-requisite(s):	Basic Mathematics (Class XII level)			

Course Description

This course introduces the fundamental concepts of Discrete Mathematics required for computer science, software engineering, data science, cybersecurity, artificial intelligence, and modern computing systems. The course focuses on developing logical reasoning, mathematical modelling, analytical thinking, and problem-solving skills essential for algorithm design, programming, databases, networks, cryptography, and computational systems.

Students will learn propositional logic, predicate logic, sets, relations, functions, combinatorics, graph theory, trees, recurrence relations, and discrete structures through practical problem solving and computational applications. The course emphasizes the direct application of discrete mathematical concepts in computer science domains such as search engines, social networks, routing systems, scheduling, data organization, compiler design, and cybersecurity.

The laboratory component focuses on logical problem solving, graph visualization, combinatorial computation, truth-table analysis, recurrence simulation, and implementation of discrete mathematics concepts using Python and modern computational tools.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply logical reasoning and proof techniques.	L3 – Apply
CO2	Solve problems using sets, relations and functions.	L3 – Apply
CO3	Apply combinatorics and recurrence relations in computing problems.	L3 – Apply
CO4	Analyze graphs, trees and network structures.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	–	1	–	–	–	–	–	–	2
CO2	3	2	1	–	1	–	–	–	–	–	1
CO3	3	3	–	2	1	–	–	–	–	–	2
CO4	3	3	2	2	2	1	–	–	–	–	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	2	1	1
CO2	3	3	2	1	2
CO3	3	3	2	2	2
CO4	3	3	3	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Logic, Proof Techniques & Mathematical Reasoning	No. of Hours: 12

- **Introduction to Discrete Mathematics & Computing Applications:** role of discrete mathematics in algorithms, AI, databases, cryptography and networks
- **Propositional Logic:** propositions, logical connectives, truth tables, logical equivalence and tautologies
- **Predicate Logic:** predicates, quantifiers and logical inference
- **Normal Forms:** conjunctive normal form (CNF) and disjunctive normal form (DNF)
- **Methods of Proof:** direct proof, contradiction, contrapositive and mathematical induction
- **Recursive Thinking & Recursive Definitions**
- **Logic in Computing Systems:** Boolean logic in programming, search conditions and digital systems
- **Practical Logic Analysis & Truth Table Generation**

Unit Number: 2	Sets, Relations, Functions & Algebraic Structures	No. of Hours: 12

- **Set Theory Fundamentals:** sets, subsets, power sets, Cartesian products and set operations
- **Venn Diagrams & Inclusion-Exclusion Principle**
- **Relations:** reflexive, symmetric, transitive and equivalence relations
- **Partial Ordering & Posets**
- **Functions:** injective, surjective, bijective and composite functions
- **Hashing & Function Applications in Computing**
- **Introduction to Algebraic Structures:** groups, semigroups and monoids with computing relevance
- **Applications in Databases, Search Systems & Access Control**

Unit Number: 3	Combinatorics, Counting & Recurrence Relations	No. of Hours: 11

- **Basic Counting Principles:** sum rule, product rule and pigeonhole principle
- **Permutations & Combinations:** arrangements, selections and binomial coefficients
- **Advanced Counting Techniques:** inclusion-exclusion principle and combinatorial reasoning
- **Discrete Probability Basics:** probability concepts used in computing systems
- **Recurrence Relations:** recursive sequences, linear recurrence relations and characteristic equations
- **Generating Functions Basics**
- **Applications in Algorithm Analysis & Dynamic Programming**
- **Combinatorial Optimization & Scheduling Problems**

Unit Number: 4	Graph Theory, Trees & Network Structures	No. of Hours: 10

- **Introduction to Graph Theory:** graphs, directed graphs, weighted graphs and graph representation techniques
- **Graph Traversal Algorithms:** BFS and DFS with applications in search and navigation systems
- **Shortest Path & Spanning Tree Concepts**
- **Trees & Binary Trees:** tree traversal, spanning trees and hierarchical data structures
- **Graph Coloring & Bipartite Graphs**
- **Applications of Graph Theory:** social networks, recommendation systems, routing and dependency management
- **Discrete Structures in AI & Computer Networks**
- **Graph Visualization & Network Analysis**

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Logic Analyzer & Proof Verification System	CO1	L3 – Apply
2	Set, Relation & Function Modeling Toolkit	CO2	L3 – Apply
3	Combinatorics & Probability Problem Solver	CO3	L3 – Apply

4	Graph Modeling & Network Visualization System	CO4	L4 – Analyze
5	Capstone Project: Smart University Analytics System using Discrete Mathematics	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Logic Analyzer & Proof Verification System	**CO1**
Problem Statement	A software testing company wants to automate the verification of logical conditions used in decision-making systems. Design Python-based tools to validate logical expressions and mathematical reasoning.
Sub-Problems	1. Generate truth tables for user-defined logical expressions. 2. Verify whether a logical expression is a tautology, contradiction, or contingency. 3. Check logical equivalence between two expressions. 4. Convert logical expressions into CNF and DNF forms. 5. Evaluate simple predicate logic statements using quantifiers. 6. Verify mathematical induction for simple summation and series formulas.
Unit 2 Lab Assignment: Set, Relation & Function Modeling Toolkit	**CO2**
Problem Statement	An organization wants to analyze employee skills, project assignments, and access permissions using mathematical models. Develop Python tools to perform set operations and relationship analysis.
Sub-Problems	1. Perform union, intersection, difference, and symmetric difference operations. 2. Generate power sets and Cartesian products. 3. Determine whether a relation is reflexive, symmetric, and transitive. 4. Represent and analyze relations using matrix representation.

	5. Verify whether a function is injective, surjective, or bijective. 6. Simulate simple hashing using modular arithmetic functions.
Unit 3 Lab Assignment: Combinatorics & Probability Problem Solver	**CO3**
Problem Statement	A digital event management company needs software tools to calculate seating arrangements, team formation possibilities, and probability-based outcomes for decision-making.
Sub-Problems	1. Calculate permutations and combinations for various scenarios. 2. Solve counting problems using sum and product rules. 3. Implement applications of the pigeonhole principle. 4. Simulate probability experiments using coins, dice, and cards. 5. Generate binomial coefficients and Pascal's Triangle. 6. Solve scheduling and allocation problems using combinatorial reasoning.
Unit 4 Lab Assignment: Graph Modeling & Network Visualization System	**CO4**
Problem Statement	A social networking platform wants to study relationships between users and communities. Design a graph-based visualization and analysis tool.
Sub-Problems	1. Create graph representations using adjacency matrices. 2. Visualize graphs using Python graph libraries. 3. Calculate degree of vertices and identify isolated nodes. 4. Analyze graph connectivity using graph properties. 5. Identify bipartite graphs and perform simple graph coloring. 6. Model real-world networks such as social networks and communication systems.
Capstone Lab Assignment: Smart University Analytics System using Discrete Mathematics	**CO1, CO2, CO3, CO4**
Problem Statement	A university wants a software solution that can analyze student activities, club memberships, event participation, and academic eligibility using concepts of discrete

	mathematics.
Sub-Problems	Project Tasks Module 1: Logical Eligibility Verification 1. Evaluate scholarship eligibility using logical conditions. 2. Generate logical decision reports using truth tables. Module 2: Student & Club Membership Analysis 3. Analyze student memberships using set operations. 4. Generate club overlap and participation statistics. Module 3: Event Planning & Resource Allocation 5. Calculate event team formations using permutations and combinations. 6. Estimate participation probabilities using probability simulations. Module 4: Campus Network Analysis 7. Model student interactions using graph representations. 8. Visualize departmental and club networks. 9. Analyze connectivity patterns and community structures. Dashboard & Reporting 10.Generate analytical reports and visual summaries using Python libraries.

Text and Reference Books

Textbooks:

1. Rosen, K. H. (2019). Discrete Mathematics and Its Applications, 8th Edition. McGraw-Hill Education. ISBN: 9781259676512
2. Gersting, J. L. (2014). Discrete Mathematics for Computer Scientists, 7th Edition. W. H. Freeman & Company. ISBN: 9781429215107
3. Doerr, A. & Levasseur, K. (2016). Applied Discrete Structures, Online Open Textbook Edition. University of Massachusetts Lowell.

Recommended Reference Books:

1. Graham, R. L., Knuth, D. E. & Patashnik, O. (1994). Concrete Mathematics, 2nd Edition. Addison-Wesley Professional. ISBN: 9780201558029
2. Kolman, B., Busby, R. & Ross, S. C. (2008). Discrete Mathematical Structures, 6th Edition. Pearson Education. ISBN: 9788120334588
3. West, D. B. (2001). Introduction to Graph Theory, 2nd Edition. Pearson Education. ISBN: 9788131720899

Recommended Online Resources:

- MIT Mathematics for Computer Science — https://ocw.mit.edu/courses/6-042j-mathematics-for-computer-science-fall-2010/
- NPTEL Discrete Mathematics — https://nptel.ac.in/courses/111106102
- VisuAlgo — https://visualgo.net/en
- Wolfram MathWorld — https://mathworld.wolfram.com/
- GeeksforGeeks Discrete Mathematics — https://www.geeksforgeeks.org/discrete-mathematics-tutorial/
- Brilliant Discrete Mathematics — https://brilliant.org/courses/discrete-mathematics/

PROBLEM SOLVING WITH C++

Program Name	B.Tech. CSE (Core)			
Course Name: Problem Solving With C++	Course Code	L-T-P	Credits	Contact Hours
	CSE105	3-0-2	4	45
Type of Course:	Major			
Semester:	II (Second Semester)			
Pre-requisite(s):	Programming for Problem Solving using Python or basic programming knowledge			

Course Description

This course introduces students to problem solving and software development using C++ through a practical and application-oriented approach. The course focuses on building logical thinking, structured programming skills, modular coding practices, and object-oriented programming concepts using modern C++ features.

Students will learn how to design efficient solutions using functions, arrays, strings, classes, and Standard Template Library (STL) components while solving real-world computational problems. The course emphasizes hands-on learning through coding exercises, debugging activities, mini applications, and lab-oriented programming tasks.

By the end of the course, students will be able to write structured, modular, reusable, and efficient C++ programs using modern development practices and problem-solving techniques.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply programming fundamentals using C++.	L3 – Apply
CO2	Solve problems using arrays, strings and functions.	L3 – Apply
CO3	Develop object-oriented applications using C++.	L3 – Apply
CO4	Use STL containers and algorithms for efficient solutions.	L3 – Apply

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	2	2	1	–	2	–	–	–	–	–	1
CO2	3	3	2	–	2	–	–	–	–	–	2
CO3	3	2	3	–	2	–	–	1	–	1	2
CO4	3	3	2	–	3	–	–	–	–	–	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	–	1
CO2	3	3	2	1	2
CO3	3	3	2	1	3
CO4	3	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Programming Fundamentals & Problem Solving	No. of Hours: 12

- Introduction to C++: structure of C++ program, compilation and execution process
- Input/Output Operations: cin, cout and formatted output
- Data Types & Variables: variables, constants, operators and expressions
- Decision Making & Loops: if-else, switch, for, while and do-while loops
- Functions & Modular Programming: function definition, parameters, return values, scope and inline functions
- Recursion Basics: factorial, Fibonacci and simple recursive problems
- Debugging & Error Handling: syntax, logical and runtime errors
- Problem-Solving Techniques: dry run, stepwise refinement and logical thinking

Unit Number: 2	Arrays, Strings & Modular	No. of Hours: 12

	Programming	

- Arrays: one-dimensional and two-dimensional arrays
- String Handling: C-style strings and std::string operations
- Passing Arrays & Strings to Functions
- Searching & Sorting Basics: linear search, binary search and basic sorting techniques
- Problem-Solving Patterns: traversal, counting and simple optimization approaches
- Pointer Basics: pointers and simple memory access concepts
- File Handling Basics: reading and writing text files
- Real-World Applications: student records, text analysis and matrix operations

Unit Number: 3	Object-Oriented Programming in C++	No. of Hours: 11

- Classes & Objects: class design, objects and member functions
- Constructors & Destructors
- Encapsulation & Abstraction
- Inheritance: single and multilevel inheritance
- Polymorphism: function overloading and operator overloading
- Friend Functions & Static Members
- Introduction to Templates
- File Handling with Objects
- Mini Application Development: student management and banking systems

Unit Number: 4	STL & Modern C++ Concepts	No. of Hours: 10

- Introduction to STL & Generic Programming
- Sequential Containers: vector, array, list and deque
- Container Adapters: stack and queue
- Associative Containers: set, map and unordered_map
- Iterators & Range-Based Loops
- STL Algorithms: sort, reverse, find, count and binary_search
- Lambda Expressions & Custom Comparators
- Efficient Problem Solving Using STL
- Mini Applications Using Multiple STL Components

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Smart Calculator & Number Analysis System	CO1	L3 – Apply
2	Pattern Generator & Recursive Problem Solver	CO1	L3 – Apply

3	Student Marks & Matrix Operations System	CO2	L3 – Apply
4	String Processing & Text Analysis Toolkit	CO2	L3 – Apply
5	Searching & Sorting Applications	CO2	L3 – Apply
6	Student Management System using OOP	CO3	L3 – Apply
7	Banking & Employee Management Applications	CO3	L3 – Apply
8	File-Based Object Storage System	CO3	L3 – Apply
9	STL-Based Data Processing Applications	CO4	L3 – Apply
10	Capstone: STL-Based Mini Application	CO1–CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Programming Fundamentals & Problem Solving	**CO1**
Problem Statement	Develop structured C++ programs to solve basic mathematical and logical problems using programming fundamentals.
Sub-Problems	• Create menu-driven calculator applications • Analyze numbers for prime, palindrome and Armstrong properties • Generate patterns using loops • Implement modular billing systems using functions • Solve factorial and Fibonacci problems using recursion • Debug syntax and logical errors
Unit 2 Lab Assignment: Arrays, Strings & Data Processing	**CO2**
Problem Statement	Implement data processing and text analysis applications using arrays, strings and modular programming concepts.
Sub-Problems	• Store and process student marks using arrays • Perform matrix operations using 2D arrays • Implement string operations such as palindrome and anagram checking • Analyze text data and count word frequency • Implement searching and sorting algorithms • Process files using basic file handling operations
Unit 3 Lab Assignment: Object-Oriented Programming Applications	**CO3**
Problem Statement	Develop real-world applications using object-oriented programming concepts in C++.

Sub-Problems	• Create class-based student management systems • Simulate banking operations using classes • Implement inheritance-based employee systems • Perform operator overloading for custom objects • Store and retrieve object data using files • Design modular OOP-based applications
Unit 4 Lab Assignment: STL & Modern C++ Applications	**CO4**
Problem Statement	Develop optimized C++ applications using STL containers and algorithms.
Sub-Problems	• Manage dynamic datasets using vectors • Count frequencies using map and unordered_map • Implement stack and queue applications • Optimize searching and sorting using STL algorithms • Use lambda expressions and custom comparators • Develop mini applications using multiple STL components
Capstone Lab Assignment: STL-Based Mini Application	**CO1, CO2, CO3, CO4**
Problem Statement	Build a modular C++ mini application integrating programming fundamentals, object-oriented programming and STL components.
Sub-Problems	• Design application workflow and modules • Implement data storage using STL containers • Apply searching and sorting operations • Use object-oriented programming concepts • Process and display structured outputs • Debug and optimize application performance

Text and Reference Books

Textbooks:

1. Bjarne Stroustrup, Programming: Principles and Practice Using C++, 3rd Edition, Addison-Wesley Professional, 2024.
2. Tony Gaddis, Starting Out with C++: From Control Structures through Objects, Pearson Education.

Recommended Reference Books:

1. Scott Meyers, *Effective Modern C++*, O'Reilly Media, 2015.
2. Stanley B. Lippman, *C++ Primer*, 5th Edition, Addison-Wesley.

3. Herbert Schildt, *C++: The Complete Reference*, McGraw-Hill Education.

Recommended Online Resources:

- C++ Reference — https://en.cppreference.com/
- LearnCpp — https://www.learncpp.com/
- GeeksforGeeks C++ — https://www.geeksforgeeks.org/c-plus-plus/
- HackerRank C++ — https://www.hackerrank.com/domains/cpp
- LeetCode — https://leetcode.com/
- Visual Studio Code C++ Documentation — https://code.visualstudio.com/docs/languages/cpp

WEB DEVELOPMENT II (ADVANCED JAVASCRIPT & REACT)

Program Name	B.Tech. CSE (Core)			
Course Name: Web Development II	Course Code	L-T-P	Credits	Contact Hours
	CSE106	3-0-2	4	45
Type of Course:	Major			
Semester:	II (Second Semester)			
Pre-requisite(s):	Web Development I (HTML, CSS, JavaScript Basics)			

Course Description

This course introduces students to modern front-end web development using Advanced JavaScript and ReactJS through a practical, industry-oriented, and project-based approach. The course is designed to help students build strong foundations in DOM manipulation, asynchronous JavaScript, browser runtime behaviour, component-based UI development, routing, API integration, and global state management.

Students learn how modern web applications are built using React, Hooks, Context API, and REST APIs while following scalable development practices and responsive UI principles. The course emphasizes hands-on implementation through real-world mini projects, debugging exercises, API-driven applications, and a capstone project involving authentication, CRUD operations, and performance optimization.

By the end of the course, students will be able to design, develop, debug, optimize, and deploy modern React-based web applications using industry-standard tools and workflows.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply JavaScript for interactive web development.	L3 – Apply
CO2	Analyze asynchronous JavaScript execution and workflows.	L4 – Analyze
CO3	Develop component-based user interfaces using React.	L3 – Apply
CO4	Implement routing, forms and API integration in React applications.	L3 – Apply
CO5	Build and deploy scalable React applications.	L5 – Evaluate / L6 – Create

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	2	2	2	–	3	–	–	–	–	–	1
CO2	2	3	–	2	3	–	–	–	–	–	2
CO3	–	2	3	–	3	–	–	1	1	–	2
CO4	–	2	3	–	3	–	1	–	1	–	2
CO5	–	2	3	2	3	–	–	2	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	2	1	1	1	1
CO2	2	2	2	1	1
CO3	2	2	1	1	2
CO4	2	2	2	2	2
CO5	2	3	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Advanced JavaScript & DOM Manipulation	No. of Hours: 12

- Document Object Model (DOM): introduction to DOM, DOM structure, DOM tree hierarchy and browser rendering concepts
- DOM traversal & element selection: getElementById(), getElementsByClassName(), querySelector(), querySelectorAll(), parent-child traversal
- Dynamic content & style manipulation: changing content using innerText, innerHTML, textContent and modifying styles dynamically
- Event handling in JavaScript: addEventListener(), removeEventListener(), event-driven programming concepts and handling user interactions
- Types of events: click, mouseover, keydown, keyup, submit events and keyboard & mouse interaction handling
- Event propagation concepts: event bubbling, event capturing, event flow and event delegation

techniques

- Default browser behavior handling: preventing default actions using event.preventDefault(), form handling and browser interaction control
- JavaScript engine, execution context & call stack: Google V8 Engine, JavaScript execution process, Global Execution Context (GEC), Function Execution Context (FEC), call stack
- Temporal Dead Zone (TDZ): hoisting concepts, let & const behavior and variable initialization rules

Unit Number: 2	Asynchronous JavaScript & JavaScript Runtime	No. of Hours: 12

- Asynchronous programming fundamentals & timers: need for asynchronous programming, synchronous vs asynchronous execution model, setTimeout(), setInterval()
- Callbacks & Promises: callback functions, callback hell, drawbacks of nested callbacks, introduction to Promises, promise states (pending, resolved, rejected), promise chaining
- JavaScript runtime & event loop: JavaScript event loop, task queue, execution order, microtasks, macrotasks, browser runtime architecture
- Web APIs & browser internals: browser APIs, asynchronous browser features and interaction between JavaScript engine and browser environment
- Async/Await & error handling: async/await syntax, asynchronous function handling, try...catch blocks, runtime error handling
- Fetch API & browser storage: Fetch API for HTTP requests, handling JSON data, Local Storage vs Session Storage, storage lifecycle
- • Advanced JavaScript functions: function generators, generator functions, function currying and reusable functional programming concepts

Unit Number: 3	React Fundamentals & Component Architecture	No. of Hours: 11

- Introduction to React & its features: overview of React, advantages of React, component-based architecture and modern front-end development concepts
- Virtual DOM & Real DOM: Virtual DOM vs Real DOM, DOM reconciliation process and efficient UI rendering concepts
- JSX & React project setup: JSX syntax and expressions, embedding JavaScript inside JSX and creating the first React application using Vite
- React components & architecture: functional components vs class components, reusable component design and component composition concepts
- State & props management: state and props, passing data through props, dynamic data rendering and component communication techniques
- Conditional rendering & lifecycle methods: conditional rendering techniques, dynamic UI updates and React component lifecycle methods
- • Styling in React applications: inline styling, CSS Modules, Styled Components, Tailwind CSS, theming concepts and global styling approaches

Unit Number: 4	React Hooks, Forms,	No. of Hours: 10

	Routing & API Integration	

- React hooks & state management: React state and hooks, managing component state using useState, updating state and component re-rendering concepts
- useEffect & side effect handling: useEffect for side effects, cleanup function in useEffect and fetching data using useEffect
- useRef & DOM interaction: useRef hook, direct DOM manipulation and managing references inside React applications
- Forms & input handling in React: controlled vs uncontrolled components, handling forms in React, input handling, form validation and form submission techniques
- React Router & navigation: React Router setup, multiple routes, Link, NavLink, route parameters, nested routes, useParams and useNavigate
- Context API & global state management: Prop Drilling & best practices, problems with Prop Drilling, creating Context, using useContext Hook
- API integration & asynchronous data handling: API integration using Fetch API and Axios, handling API errors, loading states and server communication techniques
- API optimization techniques: debouncing API calls, pagination, infinite scroll and efficient data rendering strategies
- Build tools & application bundling: Babel and Webpack, code transpilation and module bundling concepts

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Smart Event Dashboard	CO1	L3 – Apply
2	Async Weather Tracker	CO2	L4 – Analyze
3	React Student Scoreboard	CO3	L3 – Apply
4	E-commerce React Application	CO3, CO4	L4 – Analyze
5	Capstone: Advanced React Application	CO1–CO5	L5 – Evaluate

Lab Assignments

Unit 1 Lab Assignment: Smart Event Dashboard	CO1
Problem Statement	Design a dynamic web application that allows users to create, manage, update, and interact with event cards using JavaScript DOM manipulation and event handling techniques.
Sub-Problems	• Create an event form to accept user inputs such as event title, description, date, and category • Dynamically create, display, update, and remove event cards using createElement(), appendChild(), and remove()

	• Implement DOM traversal and element selection using getElementById(), querySelector(), and querySelectorAll() • Update content and styles dynamically using innerText, innerHTML, textContent, style, and class manipulation • Handle user interactions using addEventListener() for click, mouseover, keydown, and submit events • Implement event bubbling, event delegation, and prevent default browser behavior using event.preventDefault() • Debug DOM behavior using browser DevTools
Unit 2 Lab Assignment: Async Weather Tracker	**CO2**
Problem Statement	Design a JavaScript-based weather tracking application that demonstrates asynchronous programming concepts, API integration, and JavaScript runtime behavior using Vanilla JavaScript and Fetch API.
Sub-Problems	• Create a weather search interface that accepts user input for city names and displays weather information dynamically • Fetch weather data from a public API using Fetch API, Promises, and async/await • Implement loading indicators and auto-refresh functionality using setTimeout() and setInterval() • Handle asynchronous operations and promise states using then(), catch(), finally(), and try...catch • Demonstrate callback hell scenarios and refactor nested callbacks using Promises and async/await • Store and retrieve previously searched cities using Local Storage for persistent browser-based data management • Analyze JavaScript runtime behavior using console-based call stack tracing, event loop execution logs, microtasks, and macrotasks
Unit 3 Lab Assignment: React Student Scoreboard	**CO3**
Problem Statement	Design a React-based student scoreboard application that displays, manages, and updates student performance data using component-based architecture and modern React development concepts.
Sub-Problems	• Create a React application using Vite and organize the project using reusable component-based architecture • Develop functional React components using JSX syntax for header, student table, score card, and result display sections • Implement state and props to manage, display, and update student information dynamically • Use component composition and parent-to-child communication techniques for passing data through props • Apply conditional rendering to display pass/fail status and

	dynamic UI updates based on student scores • Design responsive and visually structured user interfaces using Tailwind CSS or CSS Modules • Demonstrate Virtual DOM updates through state changes and refactor components for better readability and reusable design
Unit 4 Lab Assignment: E-commerce React Application	**CO3, CO4**
Problem Statement	Design a multi-page React-based e-commerce application that allows users to browse products, manage cart operations, navigate between pages, and interact with API-driven product data using React Hooks and React Router.
Sub-Problems	• Create a multi-page React application using React Router with separate routes for product listing, product details, cart, and checkout pages • Manage product state, cart functionality, and form validation using useState and dynamic state updates • Fetch product data from a public API using useEffect and display products dynamically with loading and error handling • Implement controlled components for checkout forms, user detail forms, input handling, and form validation techniques • Use useRef for direct DOM interaction such as search input focus handling and form element referencing • Implement routing and navigation using BrowserRouter, Route, Link, NavLink, useParams(), and useNavigate() • Develop add-to-cart and remove-from-cart functionality with dynamic UI rendering and state synchronization • Implement debounced product search, pagination, and infinite scroll techniques for optimized product browsing experience • Handle API loading states, asynchronous errors, and clean user feedback mechanisms during server communication • Design responsive and modern user interfaces using Tailwind CSS and reusable React component architecture
Capstone Lab Assignment: Advanced React Application Development	**CO1, CO2, CO3, CO4, CO5**
Problem Statement	Design and develop a scalable React-based application that solves a real-world problem using modern front-end development practices, API integration, global state management, routing, performance optimization, and responsive UI techniques.
Sub-Problems	• Select a unique real-world problem domain (education, healthcare, finance, e-commerce, travel, sports analytics, environment, logistics, or social media analytics) and define project objectives • Create a React application using Vite and organize the project

	using reusable component-based architecture and modular folder structure • Implement routing and navigation using React Router including multiple routes, dynamic routing, nested routes, protected routes, and programmatic navigation • Manage application-level data using Redux Toolkit or Context API for global state management and dynamic UI synchronization • Integrate external APIs or datasets using Fetch API or Axios and implement CRUD operations for dynamic data management • Develop responsive and modern user interfaces using Tailwind CSS, CSS Modules, or Styled Components with reusable UI components and dark mode support • Implement advanced application features such as authentication, pagination, infinite scrolling, debounced API calls, search, filtering, sorting, charts, multi-step forms, or real-time data refresh • Handle asynchronous operations, loading states, runtime errors, and API failures using proper error handling and debugging techniques • Apply performance optimization techniques including lazy loading, code splitting, memoization, optimized rendering, and efficient component management • Deploy the application using platforms such as Vercel or Netlify and maintain project documentation, GitHub repository, and project presentation materials

Text and Reference Books

Textbooks:

1. Duckett, J. (2014). JavaScript and jQuery: Interactive Front-End Web Development, 1st Edition. Wiley Publishing. ISBN: 978-1118531648
2. Banks, A. & Porcello, E. (2024). Learning React: Modern Patterns for Developing React Apps, 3rd Edition. O'Reilly Media. ISBN: 978-1098132248
3. Haverbeke, M. (2024). Eloquent JavaScript, 4th Edition. No Starch Press. ISBN: 978-1718503927

Recommended Reference Books:

1. Flanagan, D. (2020). JavaScript: The Definitive Guide, 7th Edition. O'Reilly Media. ISBN: 978-1491952023
2. Osmani, A. (2023). Learning JavaScript Design Patterns, 2nd Edition. O'Reilly Media. ISBN: 978-1098139865
3. Wieruch, R. (2024). The Road to React, 2024 Edition. Self-Published. ISBN: 978-1720043992
4. Simpson, K. (2020). You Don't Know JS Yet: Get Started, 2nd Edition. O'Reilly Media. ISBN: 978-1098124045

Recommended Online Resources:

- MDN JavaScript Documentation — https://developer.mozilla.org/en-US/docs/Web/JavaScript
- React Official Documentation — https://react.dev/learn
- JavaScript.info — https://javascript.info/
- freeCodeCamp React Course — https://www.freecodecamp.org/learn/front-end-development-libraries/react/
- Tailwind CSS Documentation — https://tailwindcss.com/docs
- Redux Toolkit Documentation — https://redux-toolkit.js.org/
- Vite Documentation — https://vitejs.dev/guide/
- Axios Documentation — https://axios-http.com/docs/intro

BASICS OF ELECTRONICS & DIGITAL LOGIC DESIGN

Program Name	B.Tech. CSE (Core)			
Course Name: Basics of Electronics & Digital Logic Design	Course Code	L-T-P	Credits	Contact Hours
	ECE001	3-0-2	4	45
Type of Course:	Major			
Semester:	II (Second Semester)			
Pre-requisite(s):	Basic knowledge of Mathematics and fundamental concepts of Physics related to electricity and electronic devices.			

Course Description

This course introduces fundamental concepts of electronics and digital logic design essential for computer science students. It covers basic electronic components, number systems, Boolean algebra, combinational and sequential circuits, and their applications in computing systems.
The course emphasizes conceptual understanding and practical implementation using simulation tools, helping students understand how hardware supports software systems. It builds a strong foundation for advanced subjects such as computer architecture, embedded systems, IoT, and hardware-aware programming.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply basic electronics and number system concepts.	L3 – Apply
CO2	Design combinational logic circuits using Boolean algebra.	L3 – Apply
CO3	Analyze sequential circuits and digital systems.	L4 – Analyze
CO4	Implement basic digital systems for computing applications.	L3 – Apply

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	–	–	2	–	–	–	–	–	1
CO2	3	2	3	–	3	–	–	–	–	–	1

CO3	3	3	–	2	3	–	–	–	–	–	2
CO4	2	2	3	–	3	–	–	1	–	1	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	–	1
CO2	3	3	2	1	2
CO3	3	3	3	1	2
CO4	3	3	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Electronics Fundamentals & Digital Number Systems	No. of Hours: 12

- **Introduction to Electronics & Computing Hardware:** role of electronics in computing systems, analog vs digital electronics, hardware-software relationship and modern digital devices
- **Basic Electrical Concepts:** voltage, current, resistance, power, Ohm's law and Kirchhoff's laws with practical circuit analysis
- **Electronic Components & Devices:** resistor, capacitor, inductor, diode, transistor and FETs, their characteristics, symbols and applications in digital circuits
- **Series & Parallel Circuits:** equivalent resistance, voltage division, current division and basic circuit design concepts
- **Number Systems & Data Representation:** binary, octal, decimal and hexadecimal systems, conversion techniques and digital representation in computers
- **Binary Codes & Arithmetic:** BCD, Excess-3 and Gray codes, binary addition, subtraction, complement arithmetic and overflow conditions
- **Signed Number Representation:** sign-magnitude, 1's complement and 2's complement representation methods
- **Introduction to Circuit Simulation Tools:** Logisim, TinkerCAD and PSPICE-based simulation workflow
- **Practical Applications:** digital displays, sensor signals, embedded electronics and data representation in computing systems

Unit Number: 2	Boolean Algebra & Combinational Logic Design	No. of Hours: 12

- **Boolean Algebra Fundamentals:** Boolean variables, logic operations, Boolean laws and simplification techniques
- **De Morgan's Theorems & Logic Optimization:** algebraic simplification and optimization of digital logic expressions
- **Logic Gates & Gate-Level Design:** AND, OR, NOT, NAND, NOR, XOR and XNOR gates, truth tables and transistor-level understanding of logic gates
- **Universal Gates:** implementation of all logic gates using NAND and NOR gates
- **Canonical Forms:** SOP, POS and canonical representation of Boolean expressions
- **Karnaugh Maps (K-Maps):** simplification of Boolean functions up to 4 variables and minimization techniques
- **Arithmetic Circuits:** half adder, full adder, half subtractor, full subtractor and binary adder circuits
- **Data Selection & Conversion Circuits:** multiplexer, demultiplexer, encoder and decoder design
- **Applications of Combinational Circuits:** arithmetic logic circuits, digital displays, decision-making systems and control circuits

Unit Number: 3	Sequential Circuits & State-Based Systems	No. of Hours: 11

- **Sequential Logic Fundamentals:** difference between combinational and sequential circuits, clock signals and timing concepts
- **Flip-Flops & Memory Elements:** SR, JK, D and T flip-flops, truth tables, excitation tables and state transitions
- **Registers & Data Transfer:** shift registers (SISO, SIPO, PISO, PIPO), serial and parallel communication concepts
- **Counters & Timing Circuits:** synchronous and asynchronous counters, up/down counters and timing applications
- **Finite State Machines (FSM):** state diagrams, Mealy and Moore models and sequence detection systems
- **Sequential Circuit Analysis:** timing behavior, propagation delay and synchronization concepts
- **Applications of Sequential Systems:** digital clocks, counters, memory units and control systems
- **Simulation of Sequential Logic Circuits Using Digital Tools**

Unit Number: 4	Digital Systems, Computer Hardware & Embedded Applications	No. of Hours: 10

- **Digital System Design:** integration of combinational and sequential circuits into digital systems
- **Computer Hardware Fundamentals:** ALU, control unit, registers, buses and instruction execution

cycle

- **Memory Systems:** RAM, ROM, cache memory and memory hierarchy concepts
- **Input-Output Systems & Interfacing:** digital input-output concepts, sensors, LEDs and interfacing fundamentals
- **Embedded Systems & IoT Basics:** microcontrollers, embedded applications and smart systems overview
- **Hardware-Software Interaction:** relationship between digital hardware, operating systems and software execution
- **Digital Logic Applications in Modern Computing:** embedded systems, automation, IoT devices and processor-based systems
- **Mini Digital System Design & Simulation Projects**

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Electronic Component Identification & Circuit Verification	CO1	L3 – Apply
2	Number System Conversion & Binary Arithmetic Simulator	CO1	L3 – Apply
3	Logic Gate Design & Universal Gate Implementation	CO2	L3 – Apply
4	K-Map Simplification & Arithmetic Circuit Design	CO2	L3 – Apply
5	Multiplexer, Decoder & Digital Display System	CO2	L3 – Apply
6	Flip-Flop & Counter Simulation System	CO3	L4 – Analyze
7	Shift Register & FSM-Based Sequence Detector	CO3	L4 – Analyze
8	Memory & Digital Hardware Simulation	CO4	L3 – Apply
9	Embedded Input-Output Interface System	CO4	L3 – Apply
10	Capstone: Smart Digital Logic-Based Mini System	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Electronics Fundamentals & Number Systems	**CO1**
Problem Statement	Analyze electronic components, electrical circuits and digital number systems relevant to modern computing devices.
Sub-Problems	1. Identify and verify resistors, capacitors, diodes and transistors using multimeter tools

	2. Design and simulate series and parallel circuits 3. Verify Ohm's law using virtual and physical circuits 4. Perform binary, decimal, octal and hexadecimal conversions 5. Implement binary arithmetic operations and overflow analysis 6. Analyze signed number representation using complement systems 7. Simulate digital data representation using logic tools
Unit 2 Lab Assignment: Boolean Algebra & Combinational Logic	**CO2**
Problem Statement	Design and optimize combinational digital circuits using Boolean algebra and logic design techniques.
Sub-Problems	1. Design and simulate basic logic gates 2. Implement logic circuits using NAND and NOR gates 3. Simplify Boolean expressions using K-Maps 4. Design half adder, full adder and subtractor circuits 5. Implement multiplexer, demultiplexer, encoder and decoder systems 6. Analyze combinational circuit outputs using simulation tools 7. Develop mini arithmetic and decision-making circuits
Unit 3 Lab Assignment: Sequential Circuits & FSM Design	**CO3**
Problem Statement	Develop and simulate sequential circuits and state-based systems using flip-flops, registers and counters.
Sub-Problems	1. Simulate SR, JK, D and T flip-flops with timing analysis 2. Design synchronous and asynchronous counters 3. Implement SISO, SIPO, PISO and PIPO shift registers 4. Develop FSM-based sequence detectors using Mealy and Moore models 5. Analyze state transitions and propagation delays 6. Simulate timing and synchronization behavior of sequential systems 7. Build mini sequential control systems

Unit 4 Lab Assignment: Digital Systems & Embedded Applications	**CO4**
Problem Statement	Design digital systems relevant to computing hardware and embedded applications using logic design concepts.
Sub-Problems	1. Simulate RAM, ROM and memory operations 2. Design simple ALU-based arithmetic operations 3. Interface LEDs, switches and sensors using digital logic 4. Analyze hardware-software interaction concepts 5. Develop basic embedded input-output systems 6. Simulate digital communication between components 7. Test and validate digital system functionality
Capstone Lab Assignment: Smart Digital Logic-Based Mini System	**CO1, CO2, CO3, CO4**
Problem Statement	Build a smart digital system integrating electronics, combinational logic, sequential logic and embedded interfacing concepts.
Sub-Problems	1. Identify real-world application problem and define system objectives 2. Design combinational and sequential logic modules 3. Integrate counters, registers and digital display systems 4. Implement input-output interfacing using LEDs or sensors 5. Simulate complete digital system behavior and outputs 6. Test and optimize circuit functionality and performance 7. Present project implementation, logic flow and application relevance

Text and Reference Books

Textbooks:

1. M. Morris Mano & Michael D. Ciletti, Digital Design, 5th Edition, Pearson Education.
2. David Harris & Sarah Harris, Digital Design and Computer Architecture, 2nd Edition, Morgan Kaufmann, 2021.

Recommended Reference Books:

1. Thomas L. Floyd, Digital Fundamentals, Pearson Education.
2. Melvino & Leach, Digital Principles and Applications, McGraw-Hill Education.

3. R. P. Jain, Modern Digital Electronics, McGraw-Hill Education.

Recommended Online Resources:

- Neso Academy Digital Electronics — https://www.youtube.com/c/NesoAcademy
- Logisim Evolution — https://github.com/logisim-evolution/logisim-evolution
- Tinker CAD Circuits — https://www.tinkercad.com/circuits
- Desforges's Digital Logic — https://www.geeksforgeeks.org/digital-logic/
- Electronics Tutorials — https://www.electronics-tutorials.ws/
- Arduino Documentation — https://docs.arduino.cc/

AI FLUENCY & DIGITAL INTELLIGENCE

Program Name	B.Tech CSE (Core)			
Course Name: AI Fluency & Digital Intelligence	**Course Code**	**L-T-P**	**Credits**	**Contact Hours**
	SEAML001	2-0-2	3	30
Type of Course:	SEC			
Semester:	II (Second Semester)			
Pre-requisite(s):	No formal pre-requisite; basic computer usage skills recommended			

Course Description

This course introduces students to the practical and responsible use of Artificial Intelligence tools across academic, professional, and real-world workflows. The course focuses on developing AI fluency — the ability to effectively interact, collaborate, evaluate, and create using modern AI systems such as generative AI, large language models (LLMs), multimodal AI tools, and AI-powered productivity platforms.

Students will learn structured prompting, AI-assisted research, content creation, data analysis, automation workflows, and responsible AI practices through hands-on activities and interdisciplinary projects. The course also develops critical thinking skills required to evaluate AI outputs, identify hallucinations and biases, and make informed decisions about when and how AI should be used.

The course emphasizes experiential learning through AI labs, collaborative exercises, verification tasks, portfolio development, and capstone projects relevant to students' academic domains and career pathways.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Explain AI fundamentals and AI interaction models.	L2 – Understand
CO2	Apply structured prompting and AI collaboration techniques.	L3 – Apply
CO3	Use AI tools for research, writing and productivity tasks.	L3 – Apply
CO4	Analyze AI outputs and delegation decisions critically.	L4 – Analyze
CO5	Evaluate responsible and ethical AI practices.	L5 – Evaluate
CO6	Develop AI-assisted solutions for domain-specific problems.	L3 – Apply

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	2	1	–	–	2	1	1	–	–	–	2
CO2	–	2	1	–	3	–	–	2	2	–	3
CO3	–	2	–	1	3	–	–	1	2	1	3
CO4	–	3	–	3	2	1	2	–	–	–	3
CO5	–	–	–	–	–	2	3	–	1	–	2
CO6	1	2	3	–	3	–	–	2	2	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	2	2	1	–	1
CO2	2	3	2	1	2
CO3	2	3	2	1	3
CO4	3	3	3	2	2
CO5	2	2	3	3	1
CO6	3	3	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	AI Foundations & Human-AI Interaction	No. of Hours: 8

- Introduction to Artificial Intelligence: AI, machine learning, deep learning and generative AI
- How Large Language Models Work: tokens, prompts, prediction and conversational AI basics
- AI Ecosystem & Modern AI Tools: ChatGPT, Claude, Gemini and multimodal AI platforms
- Human-AI Interaction Models: automation, augmentation and AI agency concepts
- Applications of AI Across Disciplines
- AI Safety & Academic Integrity: privacy, bias, misinformation and responsible AI usage

- Understanding AI Limitations & Hallucinations
- Hands-on Exploration of AI Tools

Unit Number: 2	Structured Prompting & AI-Assisted Workflows	No. of Hours: 8

- Anatomy of Effective Prompts: role, task, context, constraints and output format
- Structured Prompting Techniques: iterative prompting and refinement strategies
- Collaboration Framing & Meta Prompting
- AI for Research & Information Gathering
- AI-Assisted Writing & Summarization
- Verification of AI-Generated Content & Citations
- AI for Learning, Productivity & Communication
- Prompt Optimization Through Practical Exercises

Unit Number: 3	AI-Augmented Creation & Productivity	No. of Hours: 7

- AI for Data Analysis & Visualization
- AI for Presentations & Content Creation
- Multimodal AI: image, audio and video generation tools
- AI Coding Assistants & Automation Tools
- No-Code & Low-Code AI Platforms
- AI for Workflow Automation & Productivity
- Evaluating AI-Generated Artifacts & Outputs
- Mini Applications Using AI Tools

Unit Number: 4	Critical Evaluation, Ethics & AI Applications	No. of Hours: 7

- Critical Evaluation of AI Outputs: hallucinations, reasoning errors and factual verification
- Delegation Decisions: when to use AI and when human effort is required
- Responsible AI Practices: transparency, attribution, IP ownership and privacy governance
- AI & Society: workforce impact, ethical concerns and policy perspectives
- AI Applications in Different Domains & Industries
- AI Portfolio & Professional Development
- Capstone Development Using AI Tools
- Presentation & Demonstration of AI-Assisted Solutions

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Exploring AI Tools & Interaction Models	CO1	L2 – Understand
2	Structured Prompting & Prompt Optimization	CO2	L3 – Apply
3	AI-Assisted Research & Writing Workflow	CO2, CO3	L3 – Apply
4	AI-Based Data Analysis & Visualization	CO3	L3 – Apply
5	Multimodal Content Creation Using AI	CO3	L3 – Apply
6	AI Coding & Automation Applications	CO3, CO6	L3 – Apply
7	AI Output Verification & Ethical Analysis	CO4, CO5	L4 – Analyze
8	Capstone: AI-Assisted Domain-Specific Solution	CO1–CO6	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: AI Foundations & Exploration	**CO1**
Problem Statement	Explore modern AI platforms and analyze different AI interaction models and applications.
Sub-Problems	• Compare outputs from multiple AI tools • Analyze automation, augmentation and AI agency examples • Identify strengths and limitations of AI systems • Explore AI applications in different domains • Analyze hallucinations and inaccurate outputs • Document AI usage observations and findings
Unit 2 Lab Assignment: Structured Prompting & AI Workflows	**CO2**
Problem Statement	Develop effective prompting strategies and AI-assisted workflows for research and productivity tasks.
Sub-Problems	• Design structured prompts with constraints and roles • Perform iterative prompt refinement • Generate research summaries using AI tools • Verify AI-generated citations and content • Create AI-assisted writing workflows

	• Compare prompt effectiveness and output quality
Unit 3 Lab Assignment: AI-Augmented Creation & Automation	**CO3**
Problem Statement	Use AI tools for content generation, automation and productivity enhancement.
Sub-Problems	• Analyze datasets using AI tools • Generate presentations and visual content using AI • Explore image, audio and video AI tools • Use AI coding assistants for development tasks • Build simple automation workflows using no-code tools • Evaluate AI-generated artifacts critically
Unit 4 Lab Assignment: Ethical AI & Domain Applications	**CO4, CO5**
Problem Statement	Analyze responsible AI practices and develop domain-specific AI-assisted solutions.
Sub-Problems	• Evaluate AI outputs for factual correctness and bias • Analyze ethical and privacy-related AI scenarios • Identify appropriate AI delegation decisions • Explore AI applications in specific disciplines • Develop AI-assisted professional portfolio components • Present responsible AI usage strategies
Capstone Lab Assignment: AI-Assisted Domain-Specific Solution	**CO1, CO2, CO3, CO4, CO5, CO6**
Problem Statement	Develop an AI-assisted solution for a real-world domain problem using responsible AI practices and structured workflows.
Sub-Problems	• Identify real-world problem and workflow requirements • Select suitable AI tools and prompting strategies • Build AI-assisted solution prototype • Evaluate outputs and verify correctness • Document ethical considerations and AI usage decisions • Present final project with demonstration and reflection

Text and Reference Books

Textbooks:

1. Ethan Mollick, Co-Intelligence: Living and Working with AI, Portfolio/Penguin, 2024.
2. UNESCO, AI Competency Framework for Students, 2024.
3. Jules White et al., Prompt Engineering for Generative AI, Vanderbilt University Resources.

Recommended Reference Books:

1. Bernard Marr, Artificial Intelligence in Practice, Wiley.
2. Russell & Norvig, Artificial Intelligence: A Modern Approach, Pearson Education.
3. David Shapiro, Generative AI for Everyone, O'Reilly Media.

Recommended Online Resources:

- OpenAI — https://platform.openai.com/docs
- Anthropic Claude — https://docs.anthropic.com/
- Google Gemini — https://deepmind.google/technologies/gemini/
- Hugging Face — https://huggingface.co/
- DeepLearning.AI — https://www.deeplearning.ai/
- Prompt Engineering Guide — https://www.promptingguide.ai/
- Google AI Studio — https://aistudio.google.com/
- Microsoft Learn AI — https://learn.microsoft.com/ai/

MAKER LAB: TINKERING WITH TECHNOLOGY

Program Name	B.Tech. CSE (Core)			
Course Name: Maker Lab: Tinkering With Technology	Course Code	L-T-P	Credits	Contact Hours
	SEET002	1-0-2	2	30
Type of Course:	SEC			
Semester:	II (Second Semester)			
Pre-requisite(s):	No formal pre-requisite; basic computer usage skills recommended			

Course Description

MakerLab is a hands-on, 2-credit interdisciplinary course for first-year B.Tech students (all branches). It is structured into two modules: Module 1 covers Electronics & Embedded Systems (EE) and Module 2 covers CAD, 3D Modeling & Digital Fabrication (ME). Each module runs for 6 weeks with 6 practicals. The course is supplemented by 6 hours of e-learning videos covering essential theory for each module.

In Module 1, students learn to design and construct circuits, program Arduino and ESP32 microcontrollers, interface sensors and actuators, and build IoT applications with cloud connectivity. In Module 2, students develop proficiency in 2D CAD drafting, 3D parametric modeling, assembly design, FDM 3D printing, and laser cutting and fabrication. The course culminates in a Capstone Project that integrates electronics, embedded systems, mechanical design, and manufacturing into a functional prototype.

The course emphasizes problem-solving through iteration, collaborative teamwork, technical communication, self-directed learning, and safety practices. Students experience the complete product development cycle — from concept through design, prototyping, testing, and presentation.

Course Objectives

- To introduce basic electronics, circuit design, and microcontroller programming for IoT applications.
- To foster innovation and entrepreneurial thinking through hands-on project-based learning.
- To provide students with foundational skills in engineering visualization, CAD modeling, and technical drawing principles.
- To develop proficiency in 3D modeling, digital fabrication techniques, and additive manufacturing.
- To cultivate problem-solving, teamwork, and interdisciplinary collaboration skills.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply electronics fundamentals for basic circuit design and analysis.	L3 – Apply
CO2	Develop Arduino and ESP32 programs for embedded applications.	L3 – Apply
CO3	Build IoT applications using sensors, actuators and cloud platforms.	L3 – Apply
CO4	Create 2D and 3D engineering models using CAD tools.	L3 – Apply
CO5	Perform 3D printing and laser cutting workflows.	L3 – Apply
CO6	Develop interdisciplinary engineering mini projects.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	2	1	2	–	–	1	–	–	1
CO2	2	2	2	–	3	–	–	1	–	–	2
CO3	2	2	3	1	3	2	1	2	1	1	2
CO4	2	–	3	–	3	–	–	2	1	1	2
CO5	–	–	2	–	3	2	1	2	–	1	1
CO6	–	2	3	2	3	2	2	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	–	2
CO2	3	3	2	1	2
CO3	3	3	2	2	3

CO4	3	2	2	1	3
CO5	2	3	2	2	3
CO6	3	3	3	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Module 1: Electronics & Embedded Systems (EE)

Duration: 6 Weeks | Total Practicals: 6

Essential Theory Topics (Covered in 6-Hour E-Learning Videos)

- Fundamentals of electronics: passive components (resistors, capacitors, inductors)
- Active components: diodes, transistors, voltage regulators
- Circuit analysis: Ohm's Law, Kirchhoff's Current and Voltage Laws, voltage dividers
- Microcontroller architecture: Arduino Uno (ATmega328P) and ESP32 overview
- Programming fundamentals: C/C++ for embedded systems, Arduino IDE basics
- Sensors: temperature (LM35, DHT11/22), light (LDR, photodiode), ultrasonic (HC-SR04), PIR motion
- Actuators: DC motors, servo motors, stepper motors, relays, buzzers
- Analog and digital signal processing: ADC, PWM, serial communication
- Communication protocols: UART, I2C, SPI basics
- ESP32 capabilities: WiFi connectivity, Bluetooth, dual-core processing
- IoT fundamentals: cloud platforms (ThingSpeak, Blynk), MQTT protocol, web servers

Practical Sessions – Arduino Based (Practicals 1–4)

Practical 1: Introduction to Arduino & Basic Electronics

Theory:

- Arduino board components, power supply, digital/analog pins
- Breadboard connections, resistor colour coding
- Arduino IDE setup, basic program structure (setup, loop)

Hands-on:

- LED blinking circuit, multimeter usage
- Digital output control, delay functions

Practical 2: Digital Sensors & Input Devices

Theory:

- Push buttons, switches, pull-up/pull-down resistors
- Ultrasonic sensor (HC-SR04) working principle, echo time calculation
- PIR motion sensor operation, digital input reading

Hands-on:

- Button-controlled LED circuit
- Distance measurement using ultrasonic sensor
- Motion detection system with PIR sensor

Practical 3: Analog Sensors & PWM Control

Theory:

- Analog-to-Digital Conversion (ADC), analogRead() function
- Temperature sensors: LM35, DHT11/DHT22
- Pulse Width Modulation (PWM), analogWrite() for LED brightness control

Hands-on:

- Temperature monitoring system
- PWM-based LED dimming and RGB LED colour mixing

EVALUATION 1: After Practical 3 – Quiz based on Practicals 1–3

Practical 4: Actuator Control & Motor Interfacing

Theory:

- DC motor operation, H-Bridge (L293D/L298N) driver circuits
- Servo motor control, PWM signal for angle positioning

Hands-on:

- DC motor speed and direction control
- Servo motor sweep and precision positioning

Practical Sessions – ESP32 Based (Practicals 5–6)

Practical 5: ESP32 WiFi & Web Server Development

Theory:

- ESP32 architecture, dual-core processor, WiFi capabilities
- WiFi modes: Station (STA), Access Point (AP), STA+AP

- HTTP protocol basics, web server creation
- HTML for web interface design, HTTP GET/POST requests

Hands-on:

- Connect ESP32 to WiFi network
- Create web server to control LEDs via web browser
- Display sensor data on web dashboard

Practical 6: IoT Cloud Integration & Data Logging

Theory:

- IoT cloud platforms: ThingSpeak, Blynk overview
- API keys, channel configuration, data visualization
- Real-time monitoring and remote control concepts

Hands-on:

- ESP32 + ThingSpeak data logging

EVALUATION 2: After Practical 6 – Quiz based on Practicals 4–6

Module 2: CAD, 3D Modeling & Digital Fabrication (ME)

Duration: 6 Weeks | Total Practicals: 6

Essential Theory Topics (Covered in 6-Hour E-Learning Videos)

- Introduction to engineering graphics and technical drawing standards (IS/ISO)
- Orthographic projections: first angle and third angle projection systems
- Computer-Aided Design (CAD) fundamentals: coordinate systems, sketching tools
- 2D drafting: lines, circles, arcs, dimensioning, layers, annotations
- 3D parametric modeling: sketches, features (extrude, revolve, sweep, loft)
- Assembly modeling: mates, constraints, interference detection
- Engineering drawing generation from 3D models
- Advanced features: patterns (linear, circular), fillets, chamfers, shells
- Design for Manufacturing (DFM) and Design for Assembly (DFA) principles
- Additive manufacturing technologies: FDM, SLA, SLS overview

- 3D printing workflow: STL export, slicing software (Cura, PrusaSlicer)
- Print parameters: layer height, infill, supports, print speed, temperature
- Laser cutting fundamentals: vector vs raster, kerf compensation
- Materials for fabrication: PLA, ABS, PETG for 3D printing; acrylic, plywood for laser cutting

Practical Sessions

Practical 1: CAD Introduction – Basic 2D Drafting

Theory:

- AutoCAD/DraftSight interface, command line, toolbars
- Basic drawing commands: Line, Circle, Arc, Rectangle, Polygon
- Modify commands: Move, Copy, Rotate, Mirror, Trim, Extend
- Dimensioning standards, text and annotation tools

Hands-on:

- Create 2D technical drawing of simple mechanical parts
- Practice dimensioning and annotation

Practical 2: 3D Modeling Basics – Simple Part Creation

Theory:

- SolidWorks/Fusion 360 interface and workflow
- Sketching plane selection, sketch tools, constraints
- 3D features: Extrude Boss/Base, Extrude Cut, Revolve
- Feature tree, design intent, parametric relationships

Hands-on:

- Model simple mechanical parts (flange, bracket, shaft)
- Use extrude and revolve features

Practical 3: Assembly Modeling & Engineering Drawings

Theory:

- Assembly environment, component insertion
- Mate types: coincident, concentric, distance, angle

- Interference detection, motion simulation
- Generating 2D drawings from 3D models: views, sections, dimensions

Hands-on:

- Create multi-part assembly (e.g., vice grip, simple mechanism)
- Generate engineering drawing with multiple views

EVALUATION 3: After Practical 3 – Quiz based on Practicals 1–3

Practical 4: Advanced CAD Modeling – Features & Patterns

Theory:

- Advanced features: Sweep, Loft, Boundary features
- Patterns: Linear, Circular, Mirror, Derived patterns
- Fillets, chamfers, shells, ribs, draft angles
- Design optimization techniques

Hands-on:

- Create complex parts using advanced features
- Apply patterns (gear, honeycomb structure, perforated plate)

Practical 5: 3D Printing Workflow – Model Preparation & FDM Printing

Theory:

- FDM printer components: hotend, build plate, extruder, motion system
- STL file format and mesh quality
- Slicing software: layer height, wall thickness, infill patterns
- Support structures, bed adhesion, print orientation
- Material properties: PLA (beginner-friendly), ABS (stronger, warping)

Hands-on:

- Export CAD model as STL file
- Slice model using Bamboo/Cura/PrusaSlicer
- 3D print custom designed part – phone stand, enclosure component (15–20 min each group)

Practical 6: Laser Cutting – Design & Fabrication

Theory:

- Laser cutter operation principles, CO2 vs fiber lasers
- Vector graphics (cutting lines) vs raster (engraving)
- Kerf compensation for precise joints
- Material selection: acrylic (3mm, 5mm), plywood, MDF
- Joint design: finger joints, T-slots, press-fit assemblies

Hands-on:

- Design 2D vector file for laser cutting (box, nameplate, gears)
- Set cutting parameters (power, speed, frequency)
- Fabricate acrylic/wood components and assemble (15–20 min each group)

EVALUATION 4: After Practical 6 – Quiz based on Practicals 4–6

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Introduction to Arduino & Basic Electronics	CO1, CO2	L3 – Apply
2	Digital Sensors & Input Devices	CO2, CO3	L3 – Apply
3	Analog Sensors & PWM Control	CO1, CO2	L3 – Apply
4	Actuator Control & Motor Interfacing	CO2, CO3	L3 – Apply
5	ESP32 WiFi & Web Server Development	CO3	L3 – Apply
6	IoT Cloud Integration & Data Logging	CO3	L3 – Apply
7	CAD Introduction – Basic 2D Drafting	CO4	L3 – Apply
8	3D Modeling Basics – Simple Part Creation	CO4	L3 – Apply

Capstone Lab Assignment: End-to-End Interdisciplinary Prototype

Problem Statement

Design and fabricate an interdisciplinary prototype that integrates electronics, embedded systems, mechanical design, and digital manufacturing into a functional product. Students will work in teams to complete the full product development cycle — from concept to prototype.

Sub-Problems

- Identify a real-world problem and propose a solution using both EE and ME competencies
- Design circuit schematics and select appropriate sensors/actuators
- Program the microcontroller (Arduino/ESP32) to implement control logic
- Design the mechanical housing or structural components using 3D CAD
- Fabricate components using 3D printing and/or laser cutting
- Integrate electronic and mechanical subsystems into a functional prototype
- Log and visualize sensor data using IoT cloud platforms where applicable
- Prepare engineering documentation: circuit diagrams, CAD drawings, and lab reports
- Present the prototype to technical and non-technical audiences

Concepts Covered: Complete integration of electronics, microcontroller programming, IoT, 2D/3D CAD, additive manufacturing, laser cutting, and professional documentation.

Expected Learning Outcomes

Technical Competencies

- Electronics Fundamentals: Understand and apply Ohm's Law, Kirchhoff's Laws, voltage dividers, and basic circuit analysis. Design and construct circuits using passive and active components on breadboards.
- Microcontroller Programming: Write, debug, and deploy C/C++ programs for Arduino Uno and ESP32 platforms. Implement control algorithms for digital/analog I/O, PWM motor control, and interrupt handling.
- Sensor Integration and Data Acquisition: Interface and calibrate sensors (temperature, light, ultrasonic, PIR) with microcontrollers. Process analog data using ADC, apply filtering techniques, and display measurements.
- Actuator Control: Control DC motors, servo motors, and stepper motors using driver circuits (H-bridge, relay modules). Implement speed control, direction control, and precise positioning.
- IoT Application Development: Configure ESP32 for WiFi, create web servers for remote control, integrate cloud platforms (ThingSpeak, Blynk) for data logging, and implement MQTT protocol.
- 2D CAD Proficiency: Create technical drawings using AutoCAD/DraftSight with proper layering, dimensioning per IS/ISO standards, and annotation. Generate manufacturing-ready drawings.
- 3D Parametric Modeling: Design complex 3D parts using SolidWorks/Fusion 360 with sketches, constraints, and parametric features. Apply advanced features including patterns, fillets, chamfers, and shells.
- Assembly Design: Create multi-component assemblies with appropriate mates and constraints. Perform interference detection, motion simulation, and generate exploded views.
- Additive Manufacturing: Prepare 3D models for FDM printing by optimizing orientation, generating

supports, and selecting print parameters. Understand PLA, ABS, and PETG material properties.

- Laser Cutting and Fabrication: Design vector graphics for laser cutting with kerf compensation. Create interlocking joints and select appropriate materials and cutting parameters.

Professional and Soft Skills

- Problem-Solving Through Iteration: Apply design thinking methodology — identify problems, brainstorm, prototype, test, and iterate. Develop resilience and a growth mindset through experimentation.
- Collaborative Teamwork: Work effectively in diverse teams, delegate tasks, manage conflicts, and deliver integrated projects combining contributions from multiple members.
- Technical Communication: Document work through lab reports, CAD drawings, and circuit diagrams. Present ideas clearly through oral presentations to technical and non-technical audiences.
- Self-Directed Learning: Navigate e-learning platforms independently, seek additional resources, troubleshoot issues systematically, and take ownership of personal skill development.
- Safety and Professional Practice: Follow laboratory safety protocols, use PPE appropriately, handle electrical components and power tools safely, and maintain organized workspaces.

Integration and Application

- Interdisciplinary Integration: Combine electronics, embedded systems, mechanical design, and manufacturing into cohesive projects. Understand how engineering domains interact in real-world products.
- End-to-End Product Development: Take a project from concept through design, prototyping, testing, and final presentation. Experience the complete product development cycle.

Assessment and Evaluation

Component	Marks (%)
Internal Assessment	**40**
• Weekly quizzes based on e-learning content	20
• Viva after 3 weeks Practical (Progressive difficulty)	20
Midterm Assessment	**20**
• Midterm Evaluation (MCQ based)	20
End Term Assessment	**40**

• Capstone Project (Design + Prototype + Presentation)	40
TOTAL	**100**

Recommended Reference Books & Resources

Module 1 – Electronics & Embedded Systems

- Paul Scherz & Simon Monk, Practical Electronics for Inventors, 4th Edition, McGraw-Hill, 2016. ISBN: 9781259587542
- Jeremy Blum, Exploring Arduino: Tools and Techniques for Engineering Wizardry, 2nd Edition, Wiley, 2019. ISBN: 9781119405375
- Neil Kolban, Kolban's Book on ESP32, Free e-Book, 2018. Available at: https://leanpub.com/kolban-ESP32

Module 2 – CAD, 3D Modeling & Digital Fabrication

- David C. Madsen & David P. Madsen, Engineering Drawing and Design, 5th Edition, Delmar Cengage, 2012. ISBN: 9781111309572
- Matt Lombard, SolidWorks 2022 Bible, Wiley, 2022. ISBN: 9781119823643
- Paul Boggard, Mastering Autodesk Fusion 360, Packt Publishing, 2022. ISBN: 9781803243030

Recommended Online Resources

- Arduino Official Documentation: https://docs.arduino.cc
- ESP32 Documentation (Espressif): https://docs.espressif.com/projects/esp-idf/en/latest/
- ThingSpeak IoT Platform: https://thingspeak.com
- Autodesk Fusion 360 Learning: https://www.autodesk.com/products/fusion-360/learn-support
- Tinkercad (Free Online CAD): https://www.tinkercad.com
- Ultimaker Cura Slicer: https://ultimaker.com/software/ultimaker-cura

Standard Operating Procedure (SOP)

Value Added Courses (VAC)

School of Engineering & Technology (SOET)

K.R. Mangalam University

---

1. Introduction

In alignment with the vision and recommendations of the National Education Policy (NEP) 2020 and the guidelines of the University Grants Commission (UGC), the School of Engineering & Technology (SOET) offers Value Added Courses (VAC) as an integral component of holistic and multidisciplinary education. These courses are designed to complement the core academic curriculum by enabling students to acquire additional knowledge, practical skills, ethical values and interdisciplinary competencies essential for personal growth, professional development and lifelong learning.

VAC courses are intended to broaden students' perspectives beyond their primary discipline and help them develop competencies relevant to emerging technologies, societal needs, innovation, entrepreneurship, communication, sustainability, digital literacy, employability and responsible citizenship.

2. Objectives of VAC Courses

The VAC framework aims to:

- Promote multidisciplinary and holistic learning as envisioned in NEP 2020
- Enhance students' employability, practical exposure and professional competencies
- Develop critical thinking, creativity, innovation and problem-solving abilities
- Encourage experiential, skill-based and application-oriented learning
- Provide exposure to emerging technologies, industry trends and societal challenges
- Strengthen ethical values, environmental consciousness and social responsibility
- Support self-learning, lifelong learning and professional growth

3. Nature of VAC Courses

- VAC courses shall be interdisciplinary and skill-oriented in nature
- The courses may focus on:
    - emerging technologies,
    - innovation,

- entrepreneurship,
- digital skills,
- sustainability,
- communication,
- design thinking,
- ethics,
- problem solving,
- financial literacy,
- professional skills and related areas

- The courses shall emphasize practical implementation, experiential learning and real-world applications rather than purely theoretical delivery
- VAC courses may be offered in:
  - offline,
  - online,
  - blended or
  - bootcamp-based modes

4. Course Structure & Delivery

- VAC courses shall generally be short-duration and flexible learning courses
- The delivery model may include:
  - workshops,
  - practical sessions,
  - mini projects,
  - case studies,
  - simulations,
  - industry interactions,
  - self-learning activities,
  - hackathons,
  - assignments and
  - collaborative learning exercises
- Industry experts, practitioners, entrepreneurs and domain specialists may be invited for course

delivery and mentoring support

- The courses shall encourage active participation, creativity and peer learning

5. Learning Outcomes

- VAC courses are expected to help students:
  - Acquire additional practical and interdisciplinary competencies
  - Develop industry-relevant and future-ready skills
  - Improve communication, teamwork and professional confidence
  - Apply knowledge in solving real-world and societal problems
  - Strengthen innovation, entrepreneurial thinking and digital readiness
  - Enhance adaptability and lifelong learning capabilities

6. Assessment & Evaluation

- The assessment methodology shall focus primarily on continuous and competency-based evaluation
- Evaluation components may include:
  - assignments,
  - practical activities,
  - projects,
  - quizzes,
  - presentations,
  - participation,
  - case studies,
  - demonstrations and
  - reflective learning reports
- The evaluation process shall emphasize practical understanding, implementation skills and active engagement rather than rote learning

7. Self-Learning & Skill Development

- VAC courses may include self-learning components through:
  - MOOCs,

- certification programs,
- online learning platforms,
- industry learning pathways and
- guided independent learning activities

- Students may be encouraged to complete industry-recognized certifications and skill-development programs aligned with the course objectives

8. Industry & Experiential Learning Integration

- VAC courses may incorporate:
  - industry projects,
  - field activities,
  - innovation challenges,
  - hackathons,
  - product-building exercises,
  - startup-oriented learning and
  - practical implementation tasks
- The courses shall promote experiential and application-oriented learning environments aligned with modern industry practices and emerging technology ecosystems

9. Quality Assurance & Continuous Improvement

- VAC courses shall be periodically reviewed and updated based on:
  - industry trends,
  - technological advancements,
  - learner feedback,
  - societal requirements and
  - employability needs
- The School may seek feedback from:
  - students,
  - faculty,
  - industry experts and
  - stakeholders for continuous enhancement of course quality and relevance

10. Alignment with NEP 2020

- The VAC framework supports the core principles of NEP 2020 by promoting:
  - Holistic and multidisciplinary education
  - Skill development and employability enhancement
  - Flexibility and learner-centric education
  - Experiential and competency-based learning
  - Critical thinking and problem-solving abilities
  - Digital learning and self-learning culture
  - Innovation, entrepreneurship and lifelong learning
  - Value-based and socially responsible education

11. Expected Outcomes

- Through the VAC ecosystem, students are expected to become:
  - Industry-ready and skill-oriented professionals
  - Creative thinkers and problem solvers
  - Socially responsible and ethically aware individuals
  - Self-driven lifelong learners
  - Innovators capable of adapting to emerging technologies and evolving global challenges

SEMESTER: III

DATA STRUCTURES

Program Name	B.Tech. CSE (Core)			
Course Name: Data Structures	Course Code	L-T-P	Credits	Contact Hours
	CSE201	3-0-2	4	45
Type of Course:	Major			
Semester:	III (Third Semester)			
Pre-requisite(s):	Basic Programming Knowledge (C/C++/Python)			

Course Description

This course introduces students to the fundamental principles of data organization, representation, and efficient problem solving through the systematic study of data structures. Students learn how abstract data types, complexity awareness, arrays, linked structures, stacks, queues, trees, heaps, hashing, tries, and graphs are used to store, process, retrieve, and manage data efficiently in computational systems.

The course emphasizes conceptual clarity, algorithmic thinking, and hands-on implementation through practical problems, structured lab work, and real-world use cases such as search, routing, autocomplete, indexing, and connectivity analysis.

By the end of the course, students will be able to choose suitable data structures, implement core operations, analyze basic efficiency trade-offs, and solve application-oriented problems using appropriate linear and non-linear structures.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	**Explain** fundamental data structures and algorithm complexity.	L2 – Understand
CO2	**Implement** linear data structures and their operations.	L3 – Apply
CO3	**Analyze** non-linear data structures and graph algorithms.	L4 – Analyze
CO4	**Evaluate** appropriate data structures for efficient problem solving.	L5 – Evaluate

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	2	1	2	–	–	–	1	1	–

CO2	3	3	2	1	2	–	–	–	1	1	1
CO3	3	3	3	2	2	–	–	–	1	1	1
CO4	3	3	3	2	3	1	–	1	2	1	1

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	2	–	1
CO2	3	3	2	1	2
CO3	3	3	3	2	3
CO4	2	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Foundations of Data Structures, Complexity & Sequential Storage	No. of Hours: 12

- **Introduction to Data Structures:** Data, information, data organization, need for data structures, primitive and non-primitive data structures.
- **Classification of Data Structures:** Linear and non-linear structures, static and dynamic structures.
- **Abstract Data Types (ADT):** Concept, characteristics, and importance of abstract data types in software development.
- **Fundamental Operations on Data Structures:** Traversal, insertion, deletion, searching, sorting, and updating operations.
- **Algorithm Efficiency:** Basic understanding of time and space complexity, best-case, average-case, and worst-case scenarios.
- **Arrays:** One-dimensional and two-dimensional arrays, memory representation, traversal, insertion, deletion, updating, and merging operations.
- **Matrix Operations:** Matrix representation, matrix addition, transpose, and simple matrix processing applications.
- **Strings as Linear Structures:** Character storage, traversal, manipulation, comparison, and basic string-processing techniques.

- **Searching Techniques:** Linear Search and Binary Search, their working principles and comparative analysis.
- **Sorting Techniques:** Bubble Sort, Selection Sort, and Insertion Sort with pass-wise understanding.
- **Recursion Fundamentals:** Recursive functions, base case, recursive case, recursion versus iteration, and simple recursive problem solving.

Unit Number: 2	Dynamic Linear Data Structures & Expression Processing	No. of Hours: 12

- **Dynamic Memory Concepts:** Limitations of arrays, dynamic storage allocation, pointers/references, and node-based representation.
- **Singly Linked Lists:** Structure, traversal, insertion, deletion, searching, and reversal operations.
- **Doubly Linked Lists:** Bidirectional traversal and update operations.
- **Circular Linked Lists:** Circular representation and traversal behavior.
- **Arrays vs Linked Lists:** Comparative analysis of static and dynamic data structures.
- **Stack ADT:** Concepts, push, pop, peek, overflow, and underflow conditions.
- **Stack Implementation:** Array-based and linked-list-based implementation approaches.
- **Applications of Stacks:** Recursion support, expression evaluation, parenthesis matching, undo operations, and browser history management.
- **Queue ADT:** Concepts, enqueue, dequeue, front, and rear operations.
- **Queue Variants:** Circular queues, double-ended queues (Deque), and introductory priority queues.
- **Queue Implementation:** Array-based and linked-list-based implementation approaches.
- **Applications of Queues:** Scheduling systems, buffering, printer queues, and service management systems.
- **Expression Processing:** Infix, postfix, and prefix expressions; infix-to-postfix conversion and postfix evaluation.
- **Hashing Fundamentals:** Hash tables, hash functions, load factor, and collision concepts.
- **Collision Resolution Techniques:** Chaining and linear probing methods.
- **Applications of Hashing:** Fast searching, indexing, caching, and retrieval systems.

Unit Number: 3	Non-Linear and Associative Data Structures	No. of Hours: 11

- **Introduction to Hierarchical Data Structures:** Need and applications of hierarchical organization of data.

- **Tree Terminology:** Root, parent, child, sibling, leaf node, internal node, level, height, and depth.
- **Binary Trees:** Concepts, properties, and common representations of binary trees.
- **Tree Traversals:** Preorder, inorder, postorder, and level-order traversal techniques.
- **Binary Search Trees (BST):** Concept, insertion, searching, deletion, minimum and maximum element retrieval.
- **Ordered Traversal:** Sorted data retrieval using inorder traversal of BSTs.
- **Limitations of BSTs:** Impact of skewed trees on search and update efficiency.
- **Balanced Trees:** Need for balancing and introduction to AVL trees.
- **AVL Tree Concepts:** Balancing intuition and conceptual understanding of LL and RR rotations.
- **Heaps:** Min heap, max heap, complete binary tree property, and array representation.
- **Heap Operations:** Insertion, deletion, and heapify operations.
- **Heap Sort:** Heap construction and sorting process using heaps.
- **Applications of Trees and Heaps:** Search systems, priority scheduling, database indexing, and memory management.

Unit Number: 4	**Graphs, Traversals, Connectivity & Data Structure Selection**	**No. of Hours: 10**

- **Introduction to Graphs:** Graph terminology, vertices, edges, directed and undirected graphs.
- **Graph Types:** Weighted and unweighted graphs, sparse and dense graphs.
- **Graph Representation:** Adjacency matrix and adjacency list representations.
- **Breadth First Search (BFS):** Traversal methodology and applications.
- **Depth First Search (DFS):** Traversal methodology and applications.
- **Connectivity Concepts:** Connected components and graph reachability.
- **Disjoint Sets and Union-Find:** Basic concepts and connectivity applications.
- **Array vs Linked List:** Selection criteria and practical trade-offs.
- **Stack vs Queue:** Comparative analysis and application scenarios.
- **BST vs Heap vs Hashing:** Comparative study of searching and retrieval structures.
- **Tree vs Graph:** Structural differences and suitability for problem solving.
- **Data Structure Selection:** Choosing appropriate data structures for different computational problems.
- **Applications of Graphs:** Social networks, routing systems, recommendation systems, dependency management, and communication networks.

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Array, Sorting, Searching & Recursion Workbench	CO1	L3 – Apply
2	Dynamic Linked Structures, Stack & Queue Processor	CO2	L3 – Apply
3	Tree, Heap, Hashing & Trie Explorer	CO3	L4 – Analyze
4	Graph Traversal & Connectivity Analyzer	CO4	L4 – Analyze
5	Capstone: Multi-Structure Data Management & Navigation System	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Array, Sorting, Searching & Recursion Workbench	**CO1**
Problem Statement	Develop C++ programs to perform array manipulation, searching, sorting, and recursion-based problem solving for efficient data processing
Sub-Problems	1. Implement array traversal, insertion, deletion, updating, and merging operations. 2. Implement Linear Search and Binary Search and compare their performance. 3. Implement Bubble Sort, Selection Sort, and Insertion Sort. 4. Implement Merge Sort and Quick Sort using recursion. 5. Perform matrix operations such as addition and transpose. 6. Implement recursive programs such as factorial, Fibonacci series, and sum of first n natural numbers. 7. Solve the Second Largest Element problem. 8. Rotate an array by k positions. 9. Merge two sorted arrays into a single sorted array. 10. Find the maximum subarray sum in a given array.
Unit 2 Lab Assignment: Dynamic Linked Structures, Stack & Queue Processor	**CO2**
Problem Statement	Develop C++ programs using classes, pointers, and dynamic memory allocation to implement linear data

	structures and solve practical computational problems.
Sub-Problems	1. Implement singly linked list operations: insertion, deletion, traversal, and searching. 2. Implement doubly linked list operations. 3. Implement stack operations using arrays and linked lists. 4. Implement queue operations using arrays and linked lists. 5. Solve parenthesis matching using stacks. 6. Implement infix-to-postfix conversion and postfix evaluation. 7. Implement a simple hash table using linear probing. 8. Reverse a linked list. 9. Find the middle element of a linked list. 10. Implement a stack using two queues. 11. Implement a queue using two stacks. 12. Design a browser history simulator using stacks.
Unit 3 Lab Assignment: Tree, Heap, Hashing & Trie Explorer	**CO3**
Problem Statement	Develop C++ programs to implement hierarchical data structures for efficient searching, sorting, and priority-based processing.
Sub-Problems	1. Construct binary trees and perform preorder, inorder, and postorder traversals. 2. Implement Binary Search Tree (BST) insertion and searching. 3. Perform BST deletion and ordered traversal. 4. Find the minimum and maximum elements in a BST. 5. Implement min heap and max heap operations. 6. Implement heapify and heap construction. 7. Implement Heap Sort. 8. Calculate the height of a binary tree. 9. Count the number of leaf nodes in a binary tree. 10. Find the kth smallest element in a BST. 11. Find the top-k largest elements using heaps. 12. Compare BST and heap applications for different problem scenarios.

Unit 4 Lab Assignment: Graph Traversal & Connectivity Analyzer	**CO4**
Problem Statement	Develop C++ programs to represent graphs and perform traversal and connectivity analysis for network-based applications.
Sub-Problems	1. Represent graphs using adjacency matrix and adjacency list. 2. Implement Breadth First Search (BFS). 3. Implement Depth First Search (DFS). 4. Find connected components in a graph. 5. Implement basic Union-Find operations. 6. Determine whether a path exists between two vertices. 7. Count the number of connected components in a graph. 8. Detect cycles in an undirected graph. 9. Find the shortest path in an unweighted graph using BFS. 10. Determine whether a graph is bipartite. 11. Find all reachable vertices from a given source node. 12. Model a simple social network or campus connectivity system.
Capstone Lab Assignment: Multi-Structure Data Management & Navigation System	**CO1, CO2, CO3, CO4**
Problem Statement	Design and develop a menu-driven Student Record & Course Management System in C++ that integrates multiple data structures for efficient data storage, retrieval, prioritization, and relationship management.
Sub-Problems	1. Store and manage student records using arrays or linked lists. 2. Implement student search and update operations. 3. Manage registration requests using stacks and queues. 4. Organize course information using Binary Search Trees. 5. Prioritize registration requests using heaps.

	6. Maintain course prerequisite relationships using graphs. 7. Generate reports on students, courses, and registrations. 8. Implement menu-driven interaction with proper validation. 9. Compare the efficiency of different data structures used in the project. 10. Demonstrate appropriate data structure selection for each module.

Text and Reference Books

Textbooks:

1. Mark Allen Weiss, Data Structures and Algorithm Analysis in C++, 4th Edition, Pearson, 2014.
2. Thareja, R. (2014). Data Structures Using C, 2nd Edition. Oxford University Press.

Recommended Reference Books:

1. Karumanchi, N. (2017). Data Structures and Algorithms Made Easy. CareerMonk Publications.
2. Goodrich, M. T., Tamassia, R. & Goldwasser, M. H. (2013). Data Structures and Algorithms in Python. Wiley.
3. Horowitz, E., Sahni, S. & Anderson-Freed, S. Fundamentals of Data Structures in C, 2nd Edition. Universities Press.

Recommended Online Resources:

- NPTEL courses on Data Structures and Programming — https://nptel.ac.in
- GeeksforGeeks – Data Structures Portal — https://www.geeksforgeeks.org/data-structures/
- VisuAlgo – visual learning platform for data structures — https://visualgo.net/en
- MIT OpenCourseWare – algorithmic and problem-solving resources — https://ocw.mit.edu
- Programiz / freeCodeCamp resources — https://www.programiz.com

DATABASE MANAGEMENT SYSTEMS

Program Name	B.Tech. CSE (Core)			
Course Name: Database Management Systems	**Course Code**	**L-T-P**	**Credits**	**Contact Hours**
	CSE202	3-0-2	4	45
Type of Course:	Major			
Semester:	III (Third Semester)			
Pre-requisite(s):	Fundamental knowledge of data structures, basic programming concepts, and understanding of file organization			

Course Description

This course provides a comprehensive, hands-on introduction to database concepts, relational database design, SQL programming, query optimization, transactions, database security, and modern industry tools. It also covers NoSQL databases, cloud databases, and real-world applications to equip students with industry-relevant skills.

Students will gain in-depth knowledge of the entity-relationship model, relational algebra, normalization theory, SQL programming, transaction management, concurrency control, and database administration. The course bridges classical database theory with modern data storage and retrieval technologies used in web applications, enterprise systems, and cloud platforms.

The laboratory component provides hands-on experience through real-world application scenarios including college management, hospital systems, e-commerce backends, and movie booking systems, culminating in a full-stack employee payroll capstone project.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Understand the core principles of database systems and relational database design.	L2 – Understand
CO2	Apply SQL-based techniques for database creation, manipulation, retrieval, and reporting.	L3 – Apply
CO3	Analyze database transaction, security, and integrity management mechanisms.	L4 – Analyze
CO4	Evaluate database performance methods and emerging database technologies.	L5 – Evaluate

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	2	1	1	-	-	1	1	-	2
CO2	3	2	2	-	3	-	-	1	1	-	2
CO3	3	3	2	2	2	1	3	1	1	-	2
CO4	2	3	2	2	3	1	1	1	1	1	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	–	1
CO2	3	3	2	1	2
CO3	3	3	3	2	2
CO4	3	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Database Design & Relational Model	No. of Hours: 12

- Introduction to DBMS: Database vs. File Systems, characteristics, applications and DBMS architecture (1-tier, 2-tier, 3-tier)
- Data Abstraction and Data Independence: Physical, Logical, and View Levels; Logical and Physical Data Independence
- DBMS vs. RDBMS with E. F. Codd's Rules
- ER Model: Entities, Attributes, Relationships, Generalization, Specialization, Aggregation; attribute and entity types
- Relationship Constraints: Cardinality, Participation, Degree of Relationship, and Mapping Constraints
- ER-to-Relational Mapping: Conversion of ER Diagrams into Relational Schemas
- Relational Model: Schema, Tuples, Attributes, Domains, Keys and Integrity Constraints
- Keys: Primary Key, Foreign Key, Candidate Key, Alternate Key, Composite Key
- Normalization Techniques: Functional Dependencies, Normal Forms from 1NF to BCNF

• Star and Snowflake Schemas for Analytics

Unit Number: 2	SQL Queries & Advanced Operations	No. of Hours: 12
• SQL Fundamentals: DDL, DML, DCL, TCL Commands • Complex Queries: Joins (Inner, Outer, Self), Subqueries, Set Operations • Views and Indexing: Creating, Modifying, and Optimizing Queries with Indexes • Advanced SQL: Window Functions, Common Table Expressions (CTEs)		

Unit Number: 3	Transactions, PL/SQL & Database Security	No. of Hours: 11
• Transactions & Concurrency Control: ACID Properties, Commit, Rollback, Savepoints • Locking & Deadlocks: Isolation Levels, Optimistic vs. Pessimistic Locking • PL/SQL Programming: Stored Procedures, Functions, Cursors, Triggers • Database Security: User Roles, Privileges, Role-Based Access Control (RBAC), SQL Injection Prevention		

Unit Number: 4	Database Performance, NoSQL & Real-World Applications	No. of Hours: 10
• Query Optimization Techniques: Execution Plans, Indexing (B-Trees, Hash Indexing) • NoSQL Database Concepts: Key-Value Stores, Document Stores (MongoDB, Firebase) • Cloud Databases & Distributed Storage (Conceptual): AWS RDS, Google BigQuery, Azure SQL • Big Data & Data Warehousing (Conceptual): ETL Processes, Data Lake vs. Data Warehouse		

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Database Design & Basic SQL – College Record Management	CO1	L3 – Apply
2	Transactions, Triggers & Procedural SQL – Hospital Appointment System	CO2	L4 – Analyze
3	Data Integration and Web Connectivity – E-Commerce Backend	CO3	L4 – Analyze
4	Advanced Queries & Access Control – Movie Ticket Booking System	CO4	L5 – Evaluate
5	Capstone: Employee Payroll & Attendance Management System	CO4	L5 – Evaluate

Lab Assignments

Unit 1 Lab Assignment: Database Design & Basic SQL – College Record Management	**CO1**
Problem Statement	Design a relational database from an ER model, normalize it, and perform basic CRUD operations using SQL.
Sub-Problems	• Draw ER diagram and normalize data up to 3NF for entities: Student, Faculty, Course, Exam • Implement schema using DDL commands in MySQL/PostgreSQL • Perform DML operations: INSERT, UPDATE, DELETE, and SELECT with filters and joins • Apply integrity constraints: Primary Key, Foreign Key, Unique, Not Null • Test with real-world student and course datasets • Generate academic reports and views using SQL queries
Unit 2 Lab Assignment: Transactions, Triggers & Procedural SQL – Hospital Appointment System	**CO2**
Problem Statement	Implement real-world logic using triggers and procedures and handle time-sensitive operations securely.
Sub-Problems	• Design tables for Doctor, Patient, Appointment, Treatment, and Prescription • Write SQL triggers to avoid overlapping appointments and log changes • Use transactions to ensure atomicity while booking and modifying appointments • Create stored procedures for patient registration and treatment entry • Implement cursor-based processing for batch operations • Test transaction rollback and recovery scenarios
Unit 3 Lab Assignment: Data Integration & Web Connectivity – E-Commerce Backend	**CO3**
Problem Statement	Connect the database with a simple frontend using backend scripting to simulate real-time order processing.
Sub-Problems	• Create normalized schema for Users, Products, Orders, and Payments • Implement SQL queries and triggers for inventory management

	• Connect database to frontend using Python Flask / PHP / Node.js • Test data insertion, deletion, and querying through a web form or API call • Implement role-based access for admin and customer users • Generate order summary and inventory reports
Unit 4 Lab Assignment: Advanced Queries & Access Control – Movie Ticket Booking System	**CO4**
Problem Statement	Work with complex queries, user input handling, and enforce access rules using SQL constraints and roles.
Sub-Problems	• Create schema for Movies, Theatres, Show Timings, Users, and Bookings • Write nested queries and joins to fetch available seats by date and time • Apply constraints to prevent double booking and overbooking • Implement basic role-based access, e.g., Admin vs. Customer roles • Use window functions for seat ranking and availability reporting • Analyze query execution plans and apply optimization techniques
Capstone Lab Assignment: Employee Payroll & Attendance Management System	**CO1, CO2, CO3, CO4**
Problem Statement	Integrate all learned concepts into a full-stack DBMS application simulating a secure payroll and HR management system.
Sub-Problems	• Design a relational schema including Employees, Attendance, Leave, Salary, and Payroll • Write stored functions and procedures for salary calculation, tax deductions, and monthly slip generation • Implement complex queries to generate reports: monthly attendance, salary slips, late marks • Develop a basic role-based web or CLI interface where HR/admin can manage employees and employees can view their records • Implement data security with role-based access control and input validation • Present final system with documentation and performance analysis

Text and Reference Books

Textbooks:

1. Silberschatz, A., Korth, H. F. & Sudarshan, S. Database System Concepts, 7th Edition. McGraw Hill.
2. Elmasri, R. & Navathe, S. B. Fundamentals of Database Systems, 7th Edition. Pearson.
3. Ramakrishnan, R. & Gehrke, J. Database Management Systems, 3rd Edition. McGraw Hill.

Recommended Reference Books:

1. Date, C. J. An Introduction to Database Systems, 8th Edition. Pearson.
2. Garcia-Molina, H., Ullman, J. D. & Widom, J. Database Systems: The Complete Book, 2nd Edition. Pearson.
3. Connolly, T. & Begg, C. Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition. Pearson.
4. Beaulieu, A. Learning SQL, 3rd Edition. O'Reilly Media.

Recommended Online Resources:

- MySQL Documentation — https://dev.mysql.com/doc/
- PostgreSQL Documentation — https://www.postgresql.org/docs/
- MongoDB Documentation — https://www.mongodb.com/docs/
- NPTEL – Database Management Systems — https://nptel.ac.in
- GeeksforGeeks – DBMS Tutorial — https://www.geeksforgeeks.org/dbms/
- SQLZoo — https://sqlzoo.net/

COMPUTER ORGANIZATION & SYSTEM ARCHITECTURE

Program Name	B.Tech. CSE (Core)			
Course Name: Computer Organization & System Architecture	Course Code	L-T-P	Credits	Contact Hours
	CSE203	3-0-2	4	45
Type of Course:	Major			
Semester:	III (Third Semester)			
Pre-requisite(s):	Digital Logic Design, Introduction to Programming, Mathematics			

Course Description

This course introduces the internal organization and working principles of modern computer systems with a strong focus on program execution, processor behavior, memory hierarchy, and system performance. The course bridges the gap between software programming and hardware execution by explaining how programs are translated, loaded, executed, and optimized inside a computer system.

Students will learn instruction set architecture, processor organization, datapath design, pipelining, memory hierarchy, cache behavior, storage systems, and CPU–operating system interaction through practical analysis and simulation-based learning. The course emphasizes how architectural decisions affect software performance, scalability, memory usage, and execution efficiency in modern computing systems.

The laboratory component focuses on RISC-V assembly programming, processor datapath tracing, pipeline analysis, cache simulation, runtime benchmarking, memory observation, and performance-oriented experimentation using modern simulation tools.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Explain program execution and computer organization concepts.	L2 – Understand
CO2	Apply RISC-V instructions and assembly programming concepts.	L3 – Apply
CO3	Analyze processor execution and pipelining behavior.	L4 – Analyze
CO4	Evaluate memory hierarchy and system performance.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	1	-	-	1	-	-	-	1	-	2
CO2	3	2	1	-	3	-	-	1	1	-	2
CO3	3	2	2	2	3	-	-	1	1	-	2
CO4	3	3	2	2	3	1	-	-	1	1	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	–	1
CO2	3	3	2	1	2
CO3	3	3	3	2	2
CO4	3	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Program Execution & Computer Organization Fundamentals	No. of Hours: 12

- Introduction to Computer Organization & Architecture: levels of abstraction, stored-program concept, Von Neumann and Harvard architectures
- Functional Units of a Computer: CPU, ALU, control unit, registers, buses, memory and I/O subsystems
- Program Execution Lifecycle: source code, compiler, assembler, linker, loader and executable generation process
- Process Memory Organization: stack, heap, code segment and data segment concepts
- Instruction Execution Cycle: fetch, decode, execute, memory access and write-back operations
- Basic Performance Concepts: clock cycle, clock rate, CPI, execution time, throughput and latency
- Data Representation Concepts: signed numbers, fixed-point and IEEE 754 floating-point representation overview
- Software-Hardware Interaction: relationship between operating systems, compilers and hardware execution

Unit Number: 2	Instruction Set Architecture & Assembly-Level Computing	No. of Hours: 12

- Instruction Set Architecture (ISA): role of ISA, instruction formats and machine-level execution
- RISC vs CISC Architectures: design philosophy, instruction complexity and performance comparison
- Addressing Modes: immediate, direct, indirect, register and indexed addressing modes
- Introduction to RISC-V Architecture: register organization, load-store architecture and instruction formats
- RISC-V Instructions: arithmetic, logical, load/store, branch and jump instructions
- Assembly-Level Program Execution: loops, conditional branching, arrays and function calls
- Stack & Function Call Mechanism: procedure calls, return addresses and stack frame basics
- Instruction Tracing & Debugging: register tracking and execution flow analysis using simulators

Unit Number: 3	Processor Organization, Datapath & Pipelining	No. of Hours: 11

- Processor Datapath Components: register file, ALU, multiplexers, instruction memory and data memory
- Single-Cycle & Multi-Cycle Processor Design: execution flow and resource utilization concepts
- Control Unit Design: hardwired control and microprogrammed control basics
- Instruction Pipelining: five-stage pipeline, pipelined datapath and pipeline performance
- Pipeline Hazards: structural, data and control hazards with forwarding and stalling techniques
- Branch Handling & Prediction Basics
- Instruction-Level Parallelism & Multi-Core Concepts: parallel execution overview and CPU scalability basics
- Performance Optimization Concepts: Amdahl's Law, cache locality and loop optimization overview
- Processor Simulation & Datapath Visualization

Unit Number: 4	Memory Hierarchy, Storage & System Performance	No. of Hours: 10

- Memory Hierarchy: registers, cache memory, main memory and secondary storage
- Cache Memory Organization: direct-mapped, associative and set-associative cache concepts
- Cache Performance Analysis: cache hits, misses, AMAT, replacement policies and write strategies
- Memory Locality & Cache-Aware Programming Concepts
- Storage Systems: SSD basics, flash memory concepts and storage performance overview
- CPU–Operating System Interaction: interrupts, DMA, context switching, privilege modes and virtual memory basics
- Input/Output Systems: programmed I/O, interrupt-driven I/O and DMA techniques
- Modern Computing Architectures: multi-core processors, GPU overview and AI hardware acceleration basics
- Performance Benchmarking & Scalability Analysis

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Program Execution & CPU Performance Analyzer	CO1	L3 – Apply
2	RISC-V Assembly Programming & Instruction Trace	CO2	L3 – Apply
3	Stack Memory & Function Call Visualization	CO2	L3 – Apply
4	Processor Datapath & Control Signal Analysis	CO3	L4 – Analyze
5	Pipeline Execution & Hazard Handling Simulator	CO3	L4 – Analyze
6	Cache Memory & AMAT Performance Analyzer	CO4	L4 – Analyze
7	Memory Hierarchy & System Benchmarking Toolkit	CO4	L4 – Analyze
8	Capstone: CPU Performance & Memory Optimization System	CO1–CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Program Execution & CPU Performance Analysis	**CO1**
Problem Statement	Analyze how programs are translated and executed inside a computer system and evaluate basic CPU performance parameters.
Sub-Problems	• Observe compilation and executable generation process • Analyze stack, heap and memory segment behavior • Convert simple programs into assembly representation • Calculate CPU execution time and CPI values • Compare performance of different processor configurations • Visualize program execution flow and memory usage
Unit 2 Lab Assignment: RISC-V Assembly Programming & Stack Analysis	**CO2**
Problem Statement	Develop and analyze assembly-level programs using RISC-V instructions and stack-based execution concepts.
Sub-Problems	• Implement arithmetic and logical operations using RISC-V assembly • Implement loops and conditional branching programs • Store and retrieve values from memory • Trace instruction execution and register updates

	• Analyze function calls and stack frame behavior • Debug assembly programs using simulator tools
Unit 3 Lab Assignment: Processor Datapath & Pipeline Analysis	**CO3**
Problem Statement	Analyze processor datapath behavior and evaluate pipelined instruction execution.
Sub-Problems	• Identify datapath components and control signals • Trace instruction execution through pipeline stages • Analyze single-cycle and multi-cycle execution flow • Detect structural, data and control hazards • Apply forwarding and stalling techniques • Compare pipelined and non-pipelined execution performance
Unit 4 Lab Assignment: Cache Memory & System Performance Analysis	**CO4**
Problem Statement	Evaluate how cache memory, storage hierarchy and I/O mechanisms affect system performance.
Sub-Problems	• Simulate direct-mapped and associative cache structures • Calculate cache hit ratio, miss ratio and AMAT • Compare LRU and FIFO replacement policies • Analyze memory locality using program execution patterns • Compare programmed I/O, interrupt-driven I/O and DMA • Benchmark memory and storage performance using datasets
Capstone Lab Assignment: CPU Performance & Memory Optimization System	**CO1, CO2, CO3, CO4**
Problem Statement	Design and develop a performance analysis system that evaluates CPU execution, cache behavior and memory optimization for different program workloads.
Sub-Problems	• Execute programs with different computational workloads • Analyze execution time and memory usage patterns • Evaluate cache hit and miss behavior for different access patterns • Compare recursive and iterative implementations • Benchmark processor and memory performance metrics • Generate performance visualization graphs and analytical reports

	• Analyze impact of cache locality on execution efficiency • Present optimization observations and system analysis results

Text and Reference Books

Textbooks:

1. Patterson, D. A. & Hennessy, J. L. (2014). Computer Organization and Design: The Hardware/Software Interface, 5th Edition. Morgan Kaufmann. ISBN: 9780124077263
2. Mano, M. M. (2017). Computer System Architecture, 3rd Edition. Pearson Education. ISBN: 9789332585607
3. Hamacher, C., Vranesic, Z. & Zaky, S. (2011). Computer Organization and Embedded Systems, 6th Edition. McGraw-Hill Education. ISBN: 9780073380650

Recommended Reference Books:

1. Stallings, W. (2018). Computer Organization and Architecture: Designing for Performance, 11th Edition. Pearson Education. ISBN: 9780134997193
2. Tanenbaum, A. S. & Austin, T. (2013). Structured Computer Organization, 6th Edition. Pearson Education. ISBN: 9789332518742
3. Harris, S. & Harris, D. (2021). Digital Design and Computer Architecture RISC-V Edition, 1st Edition. Morgan Kaufmann. ISBN: 9780128200643

Recommended Online Resources:

- RARS RISC-V Simulator — https://github.com/TheThirdOne/rars
- Ripes Processor Simulator — https://ripes.me/
- Venus RISC-V Simulator — https://venus.cs61c.org/
- MIT OpenCourseWare Computer Architecture — https://ocw.mit.edu/courses/6-004-computation-structures-spring-2017/
- NPTEL Computer Organization & Architecture — https://nptel.ac.in/courses/106106166
- GeeksforGeeks COA Tutorials — https://www.geeksforgeeks.org/computer-organization-and-architecture-tutorials/
- Logisim Evolution — https://github.com/logisim-evolution/logisim-evolution

WEB DEVELOPMENT III – NODE.JS & EXPRESS BACKEND

Program Name	B.Tech. CSE (Core)			
Course Name: Web Development III: Node.js & Express Backend	**Course Code**	**L-T-P**	**Credits**	**Contact Hours**
	CSE204	3-0-2	4	45
Type of Course:	Major			
Semester:	III (Third Semester)			
Pre-requisite(s):	Web Development II (Advanced JavaScript & ReactJS), Database concepts			

Course Description

This course focuses on modern backend application development using industry-standard technologies such as Node.js, Express.js, and MongoDB. The course introduces students to the design and development of scalable, secure, and maintainable server-side applications following current web development practices.

Students will learn to build RESTful APIs, perform database operations, manage server-side routing, implement authentication and authorization mechanisms, handle middleware, and integrate backend services with frontend applications. Emphasis is placed on practical learning through hands-on exercises, real-world case studies, and project-based implementation.

The course also covers essential backend engineering concepts such as API architecture, asynchronous programming, error handling, data validation, security practices, and deployment strategies. By the end of the course, students will be capable of developing production-ready backend systems and deploying full-stack web applications suitable for modern industry requirements.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Build server-side applications using Node.js and Express.	L3 – Apply
CO2	Design and develop RESTful APIs for web-based solutions.	L4 – Analyze
CO3	Integrate backend systems with databases for data persistence.	L3 – Apply
CO4	Create Full-Stack Applications using MERN Stack.	L6 – Create

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	2	-	3	-	-	1	1	1	2
CO2	3	3	3	1	3	-	-	1	1	1	2
CO3	3	2	2	1	3	-	1	2	1	1	2
CO4	3	3	3	2	3	1	2	3	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	3	2	2	2
CO2	3	3	2	1	2
CO3	3	3	2	2	3
CO4	3	3	3	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Foundations of Node.js	No. of Hours: 12

- Backend and HTTP Fundamentals: Introduction to Backend Development, HTTP Fundamentals, Node.js Runtime Environment, NPM and Package Management
- Modules and Core Modules in Node.js: Types of Modules in Node.js, Module Export and Import Methods, Core Modules (OS, Path, DNS, FS, Crypto)
- Process Management and Server Creation: Process Object and process.argv, HTTP Module and Server Creation, Handling Requests and Responses
- API Development and Testing Tools: API Testing Tools (Postman, Thunder Client)

Unit Number: 2	Express.js & REST API Development	No. of Hours: 12

- Express.js Fundamentals: Introduction to Express.js, Express Application Setup, MVC Architecture
- Routing and Middleware in Express: Routing (Static and Dynamic), Route Parameters, Middleware

Concepts and Types (Built-in, Custom, Third-party), Middleware Chaining

- Request Handling and REST API Development: Request and Response Objects, JSON Handling with express.json(), RESTful API Design, CRUD Operations
- Modular API Development: Express Router and Modular Routing, Cross-Origin Resource Sharing (CORS), Error Handling in Express

Unit Number: 3	MongoDB & Mongoose (ODM)	No. of Hours: 11

- Database Fundamentals and MongoDB Basics: Introduction to Databases (SQL vs NoSQL), MongoDB Basics and Installation, MongoDB Atlas/Local Setup and Configuration
- MongoDB CRUD and Query Operations: MongoDB CRUD Operations, Query Operations in MongoDB, Aggregation Pipeline, Indexing
- Data Modelling and Mongoose ODM: Data Modelling, Relationships in NoSQL Databases, Introduction to Mongoose ODM, Schema and Model Design, Data Validation
- Database Integration with Node.js: Connecting Node.js with MongoDB, Building REST APIs with Database Integration

Unit Number: 4	Authentication, Authorization & Real-Time Applications	No. of Hours: 10

- Environment and Security Fundamentals: Environment Variables and Configuration Management (dotenv), Authentication and Authorization Concepts
- Authentication and Access Control: Password Hashing (bcrypt), JSON Web Tokens (JWT), Access Tokens and Refresh Tokens, Middleware for Authentication, Protected Routes, Role-Based Access Control (RBAC)
- Event-Driven and Real-Time Communication: EventEmitter, WebSockets, Real-time Communication, Socket.IO
- Full-Stack Application Development: Full-stack Application Development, Integration of Frontend and Backend Systems

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Smart Utility Toolkit	CO1	L3 – Apply
2	Student Service Management System	CO2	L4 – Analyze
3	Inventory and Data Management System	CO3	L3 – Apply
4	Secure Connect and Communication Platform	CO4	L6 – Create
5	Capstone: Book My Show Full Stack Project	CO1-CO4	L6 – Create

Lab Assignments

Unit 1 Lab Assignment: Smart Utility Toolkit	**CO1**
Problem Statement	Develop a utility-based backend toolkit that performs command-line operations, file handling, modular programming, and server creation.
Sub-Problems	• Create a CLI-based calculator • Develop reusable custom modules • Build an HTTP server with routing • Implement file management operations • Generate random outputs using crypto module • Test endpoints using Postman or Thunder Client
Unit 2 Lab Assignment: Student Service Management System	**CO2**
Problem Statement	Design and develop a backend service management system capable of handling API requests, middleware processing, and CRUD operations.
Sub-Problems	• Create Express routes with static and dynamic routing • Develop CRUD APIs for student service management • Implement custom middleware for request validation and logging • Handle API responses and errors with proper status codes • Test APIs using Postman or Thunder Client • Implement Express Router for modular API organization
Unit 3 Lab Assignment: Inventory and Data Management System	**CO3**
Problem Statement	Develop a database-driven backend application with persistent storage, schema validation, querying, and API integration.
Sub-Problems	• Design schemas and models using Mongoose ODM • Perform CRUD operations with MongoDB database integration • Create collection relationships (One-to-Many, Many-to-Many) • Implement filtering and search functionality with query operations • Develop APIs using MVC architecture for inventory management • Apply data validation and error handling for robust database operations

Unit 4 Lab Assignment: Secure Connect and Communication Platform	**CO4**
Problem Statement	Create a secure and real-time backend application with authentication, authorization, protected APIs, and communication features.
Sub-Problems	• Develop signup and login system with password hashing using bcrypt • Implement JWT authentication with access tokens and refresh tokens • Create protected routes with authentication middleware • Build a secure Notes API with role-based access control • Develop real-time chat functionality using Socket.IO • Implement event-driven features using EventEmitter
Capstone Lab Assignment: Book My Show Full Stack Project	**CO1, CO2, CO3, CO4**
Problem Statement	Build and deploy a complete full-stack application with authentication, real-time communication, third-party integrations, and deployment support.
Sub-Problems	• Develop a complete backend-driven movie/event booking application • Integrate GitHub OAuth login for user authentication • Implement email sending functionality using Nodemailer or SendGrid • Create a real-time chat or notification communication module using Socket.IO • Deploy the application on a cloud platform (Render, Railway, or Heroku) • Maintain GitHub repository with proper documentation and API documentation using Postman collections

Text and Reference Books

Textbooks:

1. Brown, E. (2019). Web Development with Node and Express, 2nd Edition. O'Reilly Media.
2. Casciaro, M. & Mammino, L. (2020). Node.js Design Patterns, 3rd Edition. Packt Publishing.
3. Bradshaw, S., Chodorow, K. & Dirolf, M. (2020). MongoDB: The Definitive Guide, 3rd Edition. O'Reilly Media.

Recommended Reference Books:

1. Hahn, E. (2016). Express in Action. Manning Publications.
2. Masse, M. (2011). REST API Design Rulebook. O'Reilly Media.

3. Hoque, S. (2020). Full Stack React Projects. Packt Publishing.
4. Niederst Robbins, J. (2018). Learning Web Design: A Beginner's Guide, 5th Edition. O'Reilly Media. ISBN: 978-1491960202

Recommended Online Resources:

- Node.js Official Documentation — https://nodejs.org/en/docs
- Express.js Official Documentation — https://expressjs.com
- MongoDB Official Documentation — https://www.mongodb.com/docs
- Mongoose Documentation — https://mongoosejs.com/docs
- MDN Web Docs – Backend & JavaScript Resources — https://developer.mozilla.org/
- Postman Learning Center — https://learning.postman.com
- Socket.IO Documentation — https://socket.io/docs
- JWT Introduction and Documentation — https://jwt.io
- freeCodeCamp Backend Development Resources — https://www.freecodecamp.org/learn/back-end-development-and-apis/
- GeeksforGeeks Node.js Tutorials — https://www.geeksforgeeks.org/nodejs

FOUNDATIONS OF EVERYDAY COMMUNICATION

Program Name	B.Tech CSE (Core)			
Course Name: Foundations of Everyday Communication	Course Code	L-T-P	Credits	Contact Hours
		3-0-0	3	45
Type of Course:	AEC			
Semester:	III (Third Semester)			
Pre-requisite(s):	None			

Course Description

This course is designed to strengthen students' everyday communication abilities through continuous speaking practice, informal discussions, storytelling, collaborative activities, and reflective conversations. The course focuses on improving conversational fluency, confidence, listening skills, spontaneity, social intelligence, and clarity of expression in real-life situations.

Instead of emphasizing textbook-based communication theory or formal interview preparation, the course creates an interactive speaking environment where students actively participate in dialogues, group discussions, storytelling, media conversations, opinion-sharing, and spontaneous speaking tasks. Learners engage with contemporary topics, literature, emotional narratives, and real-world situations to build natural speaking competence.

The course emphasizes practical communication growth through peer interaction, conversational simulations, reflection activities, and guided speaking exercises. By the end of the course, students will be able to communicate more confidently, think while speaking, participate actively in conversations, and express ideas with improved fluency and clarity.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Communicate confidently in everyday conversations.	L3 – Apply
CO2	Express opinions and ideas with clarity and fluency.	L3 – Apply
CO3	Participate effectively in discussions and collaborative speaking tasks.	L3 – Apply
CO4	Deliver structured and reflective conversational presentations.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	-	-	-	-	-	-	-	2	3	-	2
CO2	-	1	-	-	-	1	1	2	3	-	2
CO3	-	1	-	-	-	1	2	3	3	1	2
CO4	-	1	-	-	-	2	2	2	3	1	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	2	2	1	–	1
CO2	2	3	2	1	1
CO3	2	3	2	1	2
CO4	2	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Finding Your Voice (Weeks 1–4)	No. of Hours: 8

- Interactive course overview and speaking diagnostic to understand learners' communication confidence and fluency levels
- Icebreaker circles, personal identity discussions and emoji-based self-introduction activities
- Reading and discussion of short literary narratives focusing on emotions, human experiences, and perspectives
- Listening to short narratives and reflective audio excerpts to build attentive listening habits
- Café conversation simulations based on everyday social situations such as first meetings and celebrations
- 'Explain a Reel' activities to improve explanation skills and conversational adaptability
- One-minute surprise topic speaking exercises to improve spontaneity and fluency
- Peer feedback discussions focused on clarity, confidence, and natural speaking tone

Unit Number: 2	Opinions & Speaking with Structure (Weeks 5–8)	No. of Hours: 8

- Informal group discussion on contemporary social topics and communication-related issues
- Friendly debate activities: presenting opinions using Idea → Reason → Example structure
- Small-group argument circles with rotating speaker roles to ensure equal participation
- Activities demonstrating how tone, voice, and expression influence communication meaning
- Discussion on conversation openings, attention-grabbing hooks, and meaningful closings
- Pair-based peer coaching: speaking practice where students observe and guide each other
- Students deliver short conversational talks on social, cultural, or personal experiences
- Self-reflection discussions on speaking progress and communication comfort

Unit Number: 3	**Media, Perspective & Language in Real Life (Weeks 9–12)**	**No. of Hours: 7**

- Comparison of headlines and media presentations to identify tone and framing differences
- Practice using contemporary communication-related vocabulary in real conversations
- Small-group discussions on trending topics, social issues, and student experiences
- Paraphrasing and reflective listening exercises where students restate others' viewpoints
- Activities on agreeing, disagreeing, appreciating, and questioning naturally
- Scenario-based communication activities involving misunderstandings, teamwork, and conflict resolution
- Story Circles: sharing personal experiences involving misunderstandings or life lessons
- Improvisation games: real-time role-switch communication games encouraging spontaneous responses

Unit Number: 4	**Storytelling, Persuasion & Capstone (Weeks 13–15)**	**No. of Hours: 7**

- Introduction to storytelling structure including situation, conflict, turning point, and reflection
- Learners share meaningful personal experiences and reflections with peers
- Conversational persuasive speaking on social awareness and everyday life topics
- Practice sessions focusing on voice modulation, eye contact, and body language
- Techniques to create stronger audience engagement through emotional connection
- Reflection activities improving awareness of communication impact
- Final Portfolio Sharing: 5-7 minute conversational presentation integrating storytelling, reflection, and researched content
- Final discussion on communication growth, confidence improvement, and future speaking goals

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level

1	Personal Introduction & Conversation Practice	CO1	L3 – Apply
2	Everyday Conversation & Group Interaction Activities	CO2	L3 – Apply
3	Opinion Sharing & Collaborative Discussion Tasks	CO3	L3 – Apply
4	Structured Personal Talk & Storytelling	CO4	L4 – Analyze
5	Capstone Conversational Presentation Portfolio	CO1–CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Building Conversational Confidence	**CO1**
Problem Statement	Participate in structured speaking activities to build comfort, fluency, and natural conversational tone.
Sub-Problems	• Introduce yourself through conversational prompts and icebreaker circles • Share personal experiences through open-ended discussions • Perform 'Explain a Reel' activity to improve explanation skills • Participate in café conversation simulation role-plays • Deliver one-minute surprise topic speeches • Give and receive structured peer feedback on communication style
Unit 2 Lab Assignment: Expressing Opinions & Structured Speaking	**CO2**
Problem Statement	Develop skills to express opinions clearly, engage in debates, and organize thoughts for effective speaking.
Sub-Problems	• Participate in group discussions on contemporary topics • Present structured opinions using Idea → Reason → Example format • Engage in friendly debate activities with respectful disagreement • Practice peer coaching through pair-based speaking exercises • Deliver a short personal talk on a chosen social or cultural experience • Reflect on speaking progress through self-assessment activities
Unit 3 Lab Assignment: Media Literacy & Practical Communication	**CO3**
Problem Statement	Develop critical thinking, active listening, and spontaneous communication skills through media

	analysis and real-life scenarios.
Sub-Problems	• Compare headlines and media presentations to identify tone differences • Participate in conversation circles on trending topics with rotating leadership • Practice paraphrasing and reflective listening techniques • Engage in scenario-based communication involving conflict resolution • Participate in improvisation games to develop spontaneous speaking • Use contemporary vocabulary in real conversations and speaking exercises
Unit 4 Lab Assignment: Storytelling & Persuasive Communication	**CO4**
Problem Statement	Develop emotionally engaging storytelling and persuasive speaking skills for authentic audience connection.
Sub-Problems	• Organize personal stories using situation-conflict-turning point-reflection structure • Share meaningful personal experiences with appropriate emotional expression • Deliver a persuasive talk on a social awareness topic • Practice voice modulation, eye contact, and body language techniques • Engage in audience interaction and question handling activities • Reflect on overall communication journey and identify areas for continued growth
Capstone Lab Assignment: Conversational Presentation Portfolio	**CO1, CO2, CO3, CO4**
Problem Statement	Deliver a final integrated conversational presentation demonstrating storytelling, reflection, and structured communication across personal and researched content.
Sub-Problems	• Prepare a 5-7 minute conversational presentation integrating storytelling and research • Include one personal experience, one researched idea, and one self-reflection component • Demonstrate appropriate voice modulation, eye contact, and audience engagement techniques • Incorporate persuasive elements and opinion-sharing within the presentation

	• Respond thoughtfully to peer questions and feedback during Q&A • Participate in final discussion on communication growth and future speaking goals

Text and Reference Books

Textbooks:

1. Carnegie, D. The Quick and Easy Way to Effective Speaking. Pocket Books, 2010.
2. Folse, K. S. Speaking of Speech: Basic Presentation Skills for Beginners. University of Michigan Press, 2019.

Recommended Reference Books:

1. Lucas, S. E. The Art of Public Speaking, 13th Edition. McGraw Hill, 2019.
2. Ladousse, G. P. Role Play. Oxford University Press, 2009.

Recommended Online Resources:

- TED Talks — https://www.ted.com/talks
- BBC Learning English — https://www.bbc.co.uk/learningenglish
- Toastmasters International — https://www.toastmasters.org/
- Coursera Communication Skills Courses — https://www.coursera.org
- YouTube Public Speaking & Conversation Channels

COMPETITIVE CODING-I

Program Name	B. Tech. CSE Core			
Course Name:	Course Code	L-T-P	Credits	Contact Hours
Competitive Coding-I	SEET003	2-0-0	2	30
Type of Course:	SEC			
Semester:	III (Third Semester)			
Pre-requisite(s):	Fundamentals of programming			

Course Description

This course introduces students to the fundamentals of competitive programming through a practical, problem-solving-oriented approach. It is designed to enhance programming logic, analytical thinking, and structured problem-solving skills using real-world coding challenges and industry-relevant examples. Students are introduced to popular coding platforms such as LeetCode and CodeChef to build consistency and exposure to coding contests.

The course emphasizes hands-on learning by covering essential data structures and algorithms including recursion, arrays, binary search, matrices, strings, linked lists, stacks, and queues. Students will learn to implement optimized solutions using standard techniques such as Kadane's Algorithm, Moore's Voting Algorithm, Dutch National Flag Algorithm, and Binary Search. Practical problem-solving sessions, company-specific questions, and debugging exercises are incorporated to improve coding efficiency and logical reasoning.

By the end of the course, students will be able to analyze programming problems systematically, design optimized solutions, and confidently participate in coding contests, technical assessments, and real-world programming challenges.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply recursion, arrays, and binary search techniques to solve computational and competitive programming problems efficiently.	L3 – Apply
CO2	Develop optimized solutions using algorithms such as Kadane's Algorithm, Moore's Voting Algorithm, Dutch National Flag Algorithm, and searching techniques.	L3 – Apply
CO3	Implement matrix, string, and linked list operations to solve real-world coding and interview-based problems.	L3 – Apply

CO4	Analyze and apply stack and queue data structures for solving complex competitive coding challenges and improving problem-solving efficiency.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	1	2	-	-	-	1	-	2
CO2	3	3	2	2	3	-	-	-	1	-	2
CO3	3	2	2	1	2	-	-	1	1	-	2
CO4	3	3	2	2	3	-	-	1	1	-	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	2	–	1
CO2	3	3	3	1	2
CO3	3	3	2	2	3
CO4	2	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Introduction to Competitive Programming & Recursion	No. of Hours: 8

- Introduction to competitive programming: overview of coding platforms such as LeetCode and CodeChef, coding environment setup, problem-solving approaches and logical thinking techniques.
- Recursion fundamentals: factorial calculation, power of a number, printing increasing/decreasing sequences, count digits, sum of array elements using recursion
- Advanced recursion problems: pivot index, remove duplicates, Fibonacci number generation, Tower of Hanoi with recursion tree visualization
- Problem-solving techniques: dry run methods, recursion tree analysis, step-wise execution tracing and optimization of recursive solutions
- Hands-on coding practice: implementation of recursive and logical programming

problems frequently asked in coding interviews and online assessments

Unit Number: 2	Arrays, Searching & Matrix Problems	No. of Hours: 8

- Array problem-solving techniques: maximum subarray sum using Kadane's Algorithm, maximum product subarray, majority element using Moore's Voting Algorithm
- Array manipulation problems: Two Sum, count elements with maximum frequency, left/right rotation of arrays by k positions
- Advanced array algorithms: sorting 0s, 1s and 2s using Dutch National Flag Algorithm, container with most water, trapping rain water problem
- Binary search applications: lower bound, upper bound, Koko Eating Bananas, First Bad Version, searching in rotated sorted arrays and Aggressive Cows problem
- Matrix-based problem solving: spiral traversal, searching elements in a matrix, printing matrix elements in sorted order, finding median in row-wise sorted matrix, identifying rows with maximum 1s and rotating matrices by 90 degrees
- Algorithmic thinking: optimization techniques, time complexity analysis and efficient searching approaches

Unit Number: 3	Strings & Linked List Programming	No. of Hours: 7

- String operations and problem solving: character frequency calculation, counting words in a string,
- Isomorphic Strings, Valid Anagram, longest prefix suffix and minimum repeat to make substring
- String manipulation techniques: pattern matching basics, frequency counting, traversal methods and interview-oriented string problems
- Linked List fundamentals: introduction to linked lists, insertion at any position, deletion from a given position, searching nodes and counting nodes
- Advanced linked list operations: reversing linked lists, finding middle nodes, merging two sorted
- linked lists
- Linked list applications: add two numbers represented by linked lists, rotate linked list and remove
- duplicates from sorted linked lists
- Coding practice: implementation of optimized string and linked list solutions for technical interview preparation

Unit Number: 4	Stack, Queue & Competitive	No. of Hours: 7

	Coding Applications	

- Stack implementation using arrays and stack-based problem solving techniques
- Stack applications: Next Greater Element, Previous Smaller Element (PSE), smaller element on left, valid parentheses checking and postfix expression evaluation
- Advanced stack problems: Min Stack, Asteroid Collision and Stock Span Problem
- Queue implementation concepts: queue using arrays, circular queue, queue using stacks and stack using queues
- Problem-solving strategies: efficient use of linear data structures for competitive coding challenges, optimization and debugging techniques
- Comprehensive revision: company-specific coding problems, mock coding practice and revision of all important concepts covered throughout the course

Recommended Reference Books

- Introduction to Algorithms by Cormen, Leiserson, Rivest, and Stein
- Cracking the Coding Interview by Gayle Laakmann McDowell
- Elements of Programming Interviews by Adnan Aziz, Tsung-Hsien Lee, and Amit Prakash

Recommended Online Resources

- LeetCode
- CodeChef
- Codeforces
- GeeksforGeeks
- HackerRank
- AtCoder
- TopCoder
- CS50 Problem Solving Resources

K.R. MANGALAM UNIVERSITY

SUMMER INTERNSHIP-I (ASSESSMENT)

Program Name	B.Tech. CSE (Core)			
Course Name: Summer Internship-I (Assessment)	Course Code	L-T-P	Credits	Contact Hours
	SICSE002	0-0-4	2	-
Type of Course:	INT			
Semester:	III (Third Semester)			
Pre-requisite(s):	-			

Course Description

The Summer Internship course is designed to provide students with practical exposure to real-world professional environments by integrating academic learning with industry-oriented experiences. Conducted during the summer break, the internship enables students to apply theoretical concepts, technical knowledge, and problem-solving skills in practical situations across industry, research, entrepreneurial, or organizational settings. Carrying a weightage of 2 academic credits, the course serves as a mandatory component of the curriculum and focuses on enhancing professional competence, workplace ethics, communication abilities, and industry readiness.

The course emphasizes experiential learning through active participation in technical, managerial, research, or innovation-based tasks under professional supervision. Students gain exposure to real-world practices, organizational workflows, and collaborative work environments, helping them develop self-learning abilities, critical thinking, adaptability, and professional responsibility. The internship duration of 6–8 weeks allows students to strengthen the connection between academic knowledge and practical applications while preparing them for future career opportunities and higher professional engagement.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Course Outcome	Bloom's Level
CO1	Apply academic knowledge and technical skills to perform assigned tasks in a professional work environment.	L3 – Apply
CO2	Demonstrate professional ethics, communication skills, and teamwork during internship activities.	L3 – Apply
CO3	Analyze real-world industry practices, organizational workflows, and problem-solving approaches observed during the internship.	L4 – Analyze
CO4	Prepare and present a comprehensive internship report reflecting learning experiences, outcomes, and professional growth.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	2	2	1	-	2	-	1	1	-	1	2
CO2	-	-	1	-	-	1	3	3	3	2	2
CO3	2	2	1	2	2	1	1	2	1	2	3
CO4	1	1	-	1	1	-	1	2	3	2	2

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	2	3	1	1	1
CO2	1	2	2	1	1
CO3	2	2	3	2	1
CO4	1	2	2	3	2

Internship Options:

Students can choose from the following options:

External (Outside KRMU) Tracks

Track 1: Industrial Internship

Track 2: Government / Research Institution

Track 3: Start-ups

Internal (Inside KRMU) Tracks

Track 4: Bootcamp

Track 5: University Specific Projects

Track 6: Ideas 4.0

Track 7: Global Certifications

Evaluation Criteria:

Each student must be evaluated on the following documents submitted after completing their internship/certification:

Deliverable	Description	Marks
Summer Internship Report File	A detailed Summer Internship report in the prescribed format including internship objectives, work completed, learning outcomes, and a signed one-page logbook with progress details endorsed by the industry mentor.	**20**
Attendance Record	Signed and stamped attendance record certified by the industry mentor/organization.	**10**
Hard Bound Completion Certificate	Submission of hard-bound copy of the internship completion certificate issued by the organization/industry.	**10**
Mentor Interaction through Projexa	Participation in four scheduled mentor interaction meetings conducted through Projexa for progress monitoring and evaluation.	**40**
Viva / Final Presentation	Final presentation and viva demonstrating internship work, technical learning, outcomes, and professional experience gained during the internship.	**20**

Rubrics:

Assessment Component	Excellent (9–10)	Good (7–8)	Satisfactory (5–6)	Needs Improvement (0–4)
Summer Internship Report File (20 Marks)	Report is highly organized, comprehensive, professionally formatted, and includes detailed learning outcomes, activities, and signed logbook with complete progress details.	Report is well-structured with relevant content and proper documentation; minor details may be missing.	Report covers basic internship details but lacks depth, clarity, or proper formatting.	Report is incomplete, poorly organized, or missing significant details/logbook.
Attendance Record (10 Marks)	Attendance is fully signed and stamped with complete authenticity and timely submission.	Attendance record is complete with minor formatting/documentation issues.	Attendance record is partially complete or submitted late.	Attendance proof is incomplete, unclear, or not authenticated.

Hard Bound Completion Certificate (10 Marks)	Properly submitted h ard-bound certificate with all required details and professional presentation.	Certificate submitted wi th minor issues in formatting or presentation.	Certificate submitted b ut lacks proper binding or some required details.	Certificate not submitte d or significantly incomplete.
Mentor Interaction through Pro jexa (40 Marks)	Attended all mentor meetings actively with excellent progress updates, responsiveness, and task completion.	Participated regularly with good interaction and satisfactory progress updates.	Limited participation or inconsistent progress reporting during meetings.	Poor participation or absence in mentor interactions.
Viva / Final Presentation (20 Marks)	Presentation is clear, confident, technically strong, and demonstrates ex cellent understanding of internship work and outcomes.	Presentation is well-delivered with good understanding and communication skills.	Presentation communicates basic work done but lacks clarity or confidence.	Presentation is unclear, incomplete, or demonstra tes insufficie nt understandin g.

SEMESTER: IV

DESIGN & ANALYSIS OF ALGORITHMS

Program Name	B.Tech. CSE (Core)			
Course Name: Design & Analysis of Algorithms	Course Code	L-T-P	Credits	Contact Hours
	CSE205	3-0-2	4	45
Type of Course:	Major			
Semester:	IV (Fourth Semester)			
Pre-requisite(s):	Data Structures, Discrete Mathematics, Programming in C/C++/Python			

Course Description

This course introduces students to the design, analysis, and optimization of algorithms for solving computational problems efficiently. The course focuses on algorithmic thinking, complexity analysis, problem-solving strategies, and implementation of efficient algorithms using modern programming practices.

Students will study searching, sorting, divide-and-conquer, greedy methods, dynamic programming, backtracking, graph algorithms, and optimization techniques through coding-based learning and practical experimentation. The course also emphasizes runtime benchmarking, scalability analysis, memory optimization, and comparative evaluation of algorithms for real-world applications.

The laboratory component focuses on hands-on implementation, visualization, performance analysis, and optimization of algorithms using modern development tools and competitive programming practices.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply complexity analysis for algorithm evaluation.	L3 – Apply
CO2	Design algorithms using standard problem-solving techniques.	L3 – Apply
CO3	Implement graph and optimization algorithms.	L3 – Apply
CO4	Analyze algorithm performance and scalability.	L4 – Analyze

CO–PO Articulation Matrix

Cos \	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11

POs											
CO1	3	3	1	2	2	–	–	–	1	1	–
CO2	3	3	3	2	3	–	–	–	1	1	1
CO3	3	3	3	3	3	1	–	1	2	1	1
CO4	3	3	2	3	2	1	–	1	2	1	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	2	1	1
CO2	3	3	2	2	2
CO3	3	3	3	2	3
CO4	2	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Foundations of Algorithm Analysis & Problem Solving	No. of Hours: 12

- **Introduction to Algorithm Design:** Characteristics of algorithms, correctness, efficiency, scalability, and role of algorithms in modern computing.
- **Asymptotic Analysis:** Big-O, Big-Ω, and Big-Θ notations.
- **Growth of Functions:** Constant, logarithmic, linear, linearithmic, polynomial, exponential, and factorial growth rates.
- **Time and Space Complexity:** Best-case, average-case, and worst-case analysis.
- **Recursion and Recurrence Relations:** Recursive algorithms, recursion trees, substitution method, and Master Theorem.
- **Amortized Analysis:** Aggregate method, accounting method, and potential method.
- **Lower Bounds for Sorting:** Decision tree model and Ω(n log n) lower bound for comparison-based sorting.

- **Algorithm Correctness:** Loop invariants, proof of correctness, and termination analysis.
- **Empirical Performance Evaluation:** Runtime benchmarking, scalability analysis, and memory profiling.

Unit Number: 2	Divide-and-Conquer, Greedy & Optimization Techniques	No. of Hours: 12

- **Divide-and-Conquer Strategy:** Problem decomposition, subproblem generation, and solution combination.
- **Divide-and-Conquer Algorithms:** Merge Sort, Quick Sort, Binary Search, and Closest Pair Problem.
- **Greedy Algorithm Design:** Greedy choice property and optimal substructure.
- **Greedy Algorithms:** Activity Selection, Fractional Knapsack, Job Sequencing, and Huffman Coding.
- **Dynamic Programming Fundamentals:** Overlapping subproblems and optimal substructure.
- **Memoization and Tabulation:** Design principles and implementation techniques.
- **Dynamic Programming Applications:** Fibonacci, 0/1 Knapsack, Longest Common Subsequence, and Matrix Chain Multiplication.
- **Comparative Analysis:** Divide-and-Conquer vs Greedy vs Dynamic Programming.

Unit Number: 3	Dynamic Programming, Backtracking & Graph Algorithms	No. of Hours: 11

- **Backtracking Fundamentals:** State-space trees, pruning techniques, and search-space exploration.
- **Backtracking Applications:** N-Queens, Subset Sum, and Graph Coloring.
- **Graph Representation:** Adjacency Matrix and Adjacency List.
- **Graph Traversal Review:** BFS and DFS (review and application perspective).
- **Shortest Path Algorithms:** Dijkstra's Algorithm and Bellman-Ford Algorithm.
- **Minimum Spanning Trees:** Prim's Algorithm and Kruskal's Algorithm.
- **Topological Sorting:** Directed acyclic graphs and dependency scheduling.
- **Algorithm Selection for Graph Problems:** Traversal, shortest path, spanning tree, and dependency analysis.
- **Applications:** Routing systems, recommendation systems, dependency management, and social networks.

Unit Number: 4	Advanced Algorithms, Complexity & Real-World Applications	No. of Hours: 10

Advanced Algorithmic Techniques

- **String Matching Algorithms:** Naïve Pattern Matching, Rabin-Karp Algorithm, and Knuth-Morris-Pratt (KMP) Algorithm.
- **Randomized Algorithms:** Concepts, randomized quicksort, and probabilistic analysis.
- **Approximation Algorithms:** Approximation techniques for Vertex Cover and Travelling Salesperson Problem.

Computational Complexity & Emerging Concepts

- **Branch and Bound:** Conceptual framework and state-space exploration.
- **Computational Complexity:** Tractable and intractable problems.
- **Complexity Classes:** P, NP, NP-Hard, and NP-Complete problems.
- **Complexity Examples:** SAT, Travelling Salesperson Problem, and Vertex Cover.
- **Emerging Algorithmic Paradigms:** Introduction to parallel algorithms, distributed algorithms, and AI-assisted optimization techniques.

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Algorithm Performance Analysis Workbench	CO1	L4 – Analyze
2	Smart Resource Allocation & Optimization System	CO1	L3 – Apply
3	Route Planning & Decision Optimization Engine	CO2	L3 – Apply
4	Intelligent Text Analytics & Complexity Explorer	CO2	L5 – Evaluate
5	Capstone: Academic Planning & Course Scheduling System	CO3	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Algorithm Performance Analysis Workbench	**CO1**
Problem Statement	A software engineering team wants to evaluate and compare the performance of different algorithmic approaches before selecting them for deployment. Develop a C++ toolkit that analyzes algorithm efficiency, scalability, and computational complexity.
Sub-Problems	1. Implement recursive and iterative versions of factorial and Fibonacci computations. 2. Experimentally analyze execution time for different input sizes. 3. Implement recurrence relation evaluation and compare observed versus theoretical growth. 4. Simulate dynamic array growth and perform amortized cost analysis.

	5. Benchmark Merge Sort and Quick Sort for increasing dataset sizes. 6. Compare memory usage of recursive and iterative approaches. 7. Generate complexity comparison reports for O(log n), O(n), O(n log n), and $O(n^2)$ algorithms.
Unit 2 Lab Assignment: Smart Resource Allocation & Optimization System	**CO2**
Problem Statement	A company wants to optimize task allocation, resource utilization, and scheduling decisions. Develop a C++ optimization system using divide-and-conquer, greedy, and dynamic programming approaches.
Sub-Problems	1. Implement Merge Sort and Quick Sort for task prioritization. 2. Solve the Activity Selection problem for meeting scheduling. 3. Implement Fractional Knapsack for resource allocation. 4. Implement Job Sequencing with Deadlines. 5. Generate Huffman Codes for efficient data compression. 6. Solve 0/1 Knapsack using Dynamic Programming. 7. Implement Longest Common Subsequence (LCS). 8. Solve Matrix Chain Multiplication using Dynamic Programming.
Unit 3 Lab Assignment: Route Planning & Decision Optimization Engine	**CO3**
Problem Statement	Develop a C++ decision-support system that analyzes routes, dependencies, and optimization constraints using graph algorithms and backtracking techniques.
Sub-Problems	1. Solve the N-Queens problem using backtracking. 2. Solve the Subset Sum problem. 3. Implement Graph Coloring. 4. Represent graphs using adjacency matrix and adjacency list. 5. Implement Dijkstra's shortest path algorithm. 6. Implement Bellman-Ford shortest path algorithm. 7. Implement Prim's or Kruskal's Minimum Spanning Tree algorithm. 8. Implement Topological Sorting for dependency resolution.
Unit 4 Lab Assignment: Intelligent Text Analytics & Complexity Explorer	**CO4**
Problem Statement	Develop a C++ text analytics and complexity exploration system that compares advanced algorithmic approaches for pattern discovery and computational problem solving.
Sub-Problems	1. Implement Naïve Pattern Matching.

	2. Implement Rabin-Karp Algorithm. 3. Implement KMP Pattern Matching Algorithm. 4. Compare runtime performance of all three pattern-matching algorithms. 5. Simulate Randomized Quick Sort and analyze performance. 6. Implement a simple approximation algorithm for Vertex Cover. 7. Analyze computational complexity of selected problems and classify them as polynomial or non-polynomial. 8. Study Branch-and-Bound concepts through a simplified Travelling Salesperson example.
Capstone Lab Assignment: Academic Planning & Course Scheduling System	**CO1, CO2, CO3, CO4**
Problem Statement	Design and develop a C++ based Academic Planning & Course Scheduling System that assists students in planning semester registrations while satisfying prerequisite constraints, optimizing academic workload, and evaluating alternative scheduling strategies.
Sub-Problems	Project Modules Module 1: Complexity & Performance Analysis 1. Compare execution performance of alternative scheduling algorithms. 2. Generate complexity reports for different scheduling strategies. Module 2: Optimization Engine 3. Use greedy techniques to allocate credits and elective choices. 4. Apply dynamic programming to optimize semester workload distribution. Module 3: Dependency Management 5. Represent course prerequisites as a directed graph. 6. Use topological sorting to generate valid course sequences. 7. Detect cyclic prerequisite structures. Module 4: Advanced Analysis 8. Apply pattern matching to search course information efficiently. 9. Evaluate alternative planning strategies using approximation and heuristic approaches. Final Deliverables 10. Menu-driven C++ application. 11. Performance comparison report. 12. Design documentation and justification of algorithm choices. 13. Demonstration and result analysis.

Text and Reference Books

Textbooks:

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2022). Introduction to Algorithms, 4th Edition. MIT Press. ISBN: 9780262046305
2. Skiena, S. S. (2020). The Algorithm Design Manual, 3rd Edition. Springer Nature. ISBN: 9783030542559
3. Levitin, A. (2023). Introduction to the Design and Analysis of Algorithms, 4th Edition. Pearson Education. ISBN: 9789356065482

Recommended Reference Books:

1. Kleinberg, J. & Tardos, E. (2005). Algorithm Design. Pearson Education. ISBN: 9780321295354
2. Bhargava, A. Y. (2024). Grokking Algorithms, 2nd Edition. Manning Publications. ISBN: 9781633438538
3. Sedgewick, R. & Wayne, K. (2011). Algorithms, 4th Edition. Addison-Wesley Professional. ISBN: 9780321573513

Recommended Online Resources:

- MIT OpenCourseWare Algorithms — https://ocw.mit.edu/courses/6-006-introduction-to-algorithms-fall-2011/
- VisuAlgo Algorithm Visualizer — https://visualgo.net/en
- GeeksforGeeks Algorithms — https://www.geeksforgeeks.org/fundamentals-of-algorithms/
- USACO Guide — https://usaco.guide/
- LeetCode — https://leetcode.com/
- HackerRank Algorithms — https://www.hackerrank.com/domains/algorithms
- NPTEL Design & Analysis of Algorithms — https://nptel.ac.in/courses/106101060

K.R. MANGALAM UNIVERSITY

OPERATING SYSTEMS

Program Name	B.Tech. CSE (Core)			
Course Name: Operating Systems	Course Code	L-T-P	Credits	Contact Hours
	CSE206	3-0-2	4	45
Type of Course:	Major			
Semester:	IV (Fourth Semester)			
Pre-requisite(s):	Computer Organization & Architecture, Programming in C/C++, Data Structures			

Course Description

This course introduces the concepts, structures and internal mechanisms of modern operating systems used for efficient management of computing resources in desktop, cloud, mobile and distributed environments. The course focuses on process management, CPU scheduling, synchronization, memory management, file systems, virtualization, containers, security and system performance optimization.

Students will learn how operating systems coordinate hardware and software resources, manage multitasking environments, provide secure execution platforms and support scalable modern computing systems. The course also introduces industry-relevant technologies such as Linux system programming, Docker containers, virtualization, multicore systems and cloud-oriented operating system concepts.

The laboratory component emphasizes practical implementation and simulation of operating system concepts using Linux, shell scripting, Python and system programming utilities. Students will gain hands-on experience in process scheduling, synchronization, memory management, system monitoring, containerization and performance analysis aligned with current industry requirements.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Explain modern operating system concepts and services.	L2 – Understand
CO2	Apply process scheduling and synchronization techniques.	L3 – Apply
CO3	Analyze memory management and virtual memory systems.	L4 – Analyze
CO4	Develop system-level solutions using Linux and OS tools.	L3 – Apply
CO5	Evaluate operating system performance, security and scalability.	L5 – Evaluate

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11

CO1	3	2	1	1	2	–	1	–	1	–	2
CO2	3	3	2	2	3	–	1	1	–	1	2
CO3	3	3	2	3	3	1	–	1	–	–	2
CO4	2	3	3	2	3	1	1	2	2	1	2
CO5	2	3	2	3	3	2	2	1	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	–	1
CO2	3	3	2	1	2
CO3	3	3	3	2	2
CO4	3	3	2	2	3
CO5	2	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	**Introduction to Operating Systems & Process Management**	**No. of Hours: 12**
Introduction to Operating Systems: objectives, services and evolution of operating systems **Types of Operating Systems:** batch, multitasking, real-time, distributed, mobile and cloud operating systems **Operating System Architecture:** monolithic, layered, microkernel and modular architectures **Processes & Threads:** process states, PCB, context switching and multithreading concepts **CPU Scheduling Algorithms:** FCFS, SJF, Priority and Round Robin scheduling **Inter-Process Communication:** shared memory and message passing techniques **Linux Process Management Basics:** process creation, process monitoring and shell commands		

Unit Number: 2	**Process Synchronization & Deadlock Management**	**No. of Hours: 12**
Process Synchronization: race conditions and critical section problem **Synchronization Mechanisms:** mutex locks, semaphores and monitors		

Classical Synchronization Problems: producer-consumer, readers-writers and dining philosophers problems
Deadlock Concepts & Conditions
Deadlock Handling Techniques: prevention, avoidance, detection and recovery
Banker's Algorithm & Resource Allocation
Multicore & Parallel Processing Concepts
Thread Scheduling & Concurrent Execution

Unit Number: 3	Memory Management & Virtual Memory	No. of Hours: 11

Memory Management Fundamentals: memory hierarchy and allocation techniques
Paging & Segmentation
Virtual Memory & Demand Paging
Page Replacement Algorithms: FIFO, LRU, Optimal and Clock algorithms
Memory Allocation Strategies: first fit, best fit and worst fit
Thrashing & Memory Optimization Techniques
File Caching & Buffering Mechanisms
Introduction to Virtualization & Containers: hypervisors, Docker containers and virtual machines

Unit Number: 4	File Systems, Security & Modern Operating Systems	No. of Hours: 10

File System Concepts: file organization, directory structures and file allocation methods
Disk Scheduling Algorithms: FCFS, SSTF, SCAN and LOOK
Input-Output Systems: interrupts, DMA and device management
Operating System Security: authentication, permissions, malware protection and secure access control
Linux System Administration: shell scripting, process management and file permissions
Cloud & Distributed Operating System Concepts
Case Studies: Linux, Android and Windows operating systems
System Monitoring & Performance Analysis Tools

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Tools / Technologies
1	Process Scheduling & Linux Process Analyzer	CO1, CO2	Linux, Python, Bash Shell
2	Synchronization & Multithreading System	CO2	Python Threading, POSIX Threads, Linux
3	Memory Management & Virtualization Analyzer	CO3	Python, Linux, Docker
4	File System, Security & System Monitoring Toolkit	CO4	Linux Commands, Bash, psutil

5	**Capstone:** Operating System Monitoring & Optimization Platform	CO1–CO5	Python, Linux, Docker, Grafana

Lab Assignments

Unit 1 Lab Assignment: Process Scheduling & CPU Utilization Simulator	**CO1, CO2**
Problem Statement	**Students** will design a simulator and monitoring toolkit that demonstrates how CPU scheduling works and how running processes can be observed in a Linux environment.
Recommended Tools	**Use** Linux terminal commands such as ps, top and htop for observation; use Python or Bash to implement scheduling logic, collect metrics and generate reports.
Sub-Problems / Student Tasks	**1.** Take process details such as process ID, arrival time, burst time and priority as input. **2.** Implement FCFS, SJF and Round Robin scheduling and display execution sequence for each algorithm. **3.** Calculate waiting time, turnaround time and response time for all processes and show averages. **4.** Simulate process states such as ready, running, waiting and terminated during execution. **5.** Use Linux process commands to observe real processes and compare them with simulator assumptions. **6.** Create a Gantt chart or tabular execution timeline for each scheduling algorithm. **7.** Write a short report on CPU utilization, throughput and algorithm suitability.
Concepts Covered	**Process** scheduling, Linux process management and CPU performance analysis
Unit 2 Lab Assignment: Synchronization & Deadlock Management System	**CO2**
Problem Statement	**Students** will develop concurrent programs that demonstrate shared-resource access, race conditions, synchronization control and basic deadlock handling.
Recommended Tools	**Use** Python threading or POSIX threads in Linux; use locks, semaphores and shared counters/queues to test synchronization behaviour.
Sub-Problems / Student Tasks	**1.** Create two or more threads that access a shared

	resource and demonstrate the race condition before applying synchronization. **2.** Use mutex locks and semaphores to protect critical sections and verify correct final output. **3.** Implement producer-consumer and readers-writers problems using suitable synchronization primitives. **4.** Simulate a deadlock condition using limited resources and multiple waiting threads/processes. **5.** Implement Banker's algorithm and identify safe and unsafe resource-allocation states. **6.** Measure synchronization overhead by comparing execution with and without locks. **7.** Submit code, output screenshots and a short explanation of observed concurrency issues.
Concepts Covered	**Synchronization,** deadlocks, threading and concurrent execution
Unit 3 Lab Assignment: Memory Management & Page Replacement Analyzer	**CO3**
Problem Statement	**Students** will create a memory management analyzer to simulate paging, page replacement, memory allocation and container-based resource observation.
Recommended Tools	**Use** Python for simulation, Linux commands for memory monitoring, and Docker for creating containers and observing basic resource utilization.
Sub-Problems / Student Tasks	**1.** Implement paging and segmentation concepts using logical address and physical address examples. **2.** Simulate FIFO, LRU and Optimal page replacement algorithms using a reference string entered by the user. **3.** Calculate page faults and page-hit ratio for each replacement algorithm. **4.** Implement first fit, best fit and worst fit allocation techniques and compare fragmentation. **5.** Monitor memory usage using Linux commands or Python libraries and record observations. **6.** Create a simple Docker container and observe its memory and CPU resource usage. **7.** Prepare a comparative report on page faults, memory utilization and container observations.
Concepts Covered	**Virtual** memory, page replacement, containers and virtualization
Unit 4 Lab Assignment: File System & Disk Scheduling Simulator	**CO4**

Problem Statement	**Students** will develop Linux-based utilities to understand file-system operations, disk scheduling, permissions, security checks and system monitoring.
Recommended Tools	**Use** Linux file commands, Bash scripts and Python psutil; use chmod/chown/ls for permissions and matplotlib or console tables for reporting.
Sub-Problems / Student Tasks	**1.** Implement file creation, deletion, directory listing and simple file-allocation simulation. **2.** Simulate disk scheduling algorithms such as FCFS, SSTF, SCAN and LOOK using request queues. **3.** Calculate total head movement and compare disk scheduling efficiency. **4.** Apply Linux file permissions and demonstrate read, write and execute access control. **5.** Monitor CPU, memory and disk status using commands or psutil-based scripts. **6.** Create logs for suspicious permission changes or abnormal resource usage. **7.** Generate a report containing disk scheduling results, security observations and monitoring screenshots.
Concepts Covered	**File** systems, Linux administration, security and monitoring
Capstone Lab Assignment: OS Performance Monitoring & Optimization System	**CO1, CO2, CO3, CO4, CO5**
Problem Statement	**Students** will integrate scheduling, memory, process and file-system monitoring into one OS performance dashboard or report-generation platform.
Recommended Tools	**Use** Python, Bash, Linux monitoring utilities, Docker and optional Grafana/Matplotlib; use scripts to collect metrics and visualize summaries.
Sub-Problems / Student Tasks	**1.** Design separate modules for CPU/process monitoring, memory monitoring, disk usage and process scheduling simulation. **2.** Collect real system metrics at fixed intervals and store them in CSV or log files. **3.** Detect high CPU usage, high memory consumption or abnormal process behaviour using threshold-based rules. **4.** Integrate page replacement or scheduling simulation results into the final platform. **5.** Visualize CPU, memory and disk trends using graphs or dashboard panels. **6.** Generate optimization suggestions such as terminating

	unnecessary processes or improving memory usage. **7.** Submit the final code, dataset/logs, screenshots and a short technical report.
Concepts Covered	**Operating** system internals, Linux monitoring, containers and performance optimization

Text and Reference Books

Textbooks:

1. Silberschatz, A., Galvin, P. B. & Gagne, G. (2018). Operating System Concepts, 10th Edition. Wiley. ISBN: 978-1119456339
2. Tanenbaum, A. S. & Bos, H. (2022). Modern Operating Systems, 5th Edition. Pearson. ISBN: 978-0137618886

Recommended Reference Books:

1. Stallings, W. (2018). Operating Systems: Internals and Design Principles, 9th Edition. Pearson. ISBN: 978-0134670955
2. Dhamdhere, D. M. (2017). Operating Systems: A Concept-Based Approach, 3rd Edition. McGraw Hill Education. ISBN: 978-9352606733
3. Arpaci-Dusseau, R. H. & Arpaci-Dusseau, A. C. (2018). Operating Systems: Three Easy Pieces, 1st Edition. Arpaci-Dusseau Books. ISBN: 978-1985086593

Recommended Online Resources:

- MIT OpenCourseWare – OS Engineering — https://ocw.mit.edu/courses/6-828-operating-system-engineering-fall-2012
- NPTEL – Operating Systems — https://nptel.ac.in/courses/106106144
- Linux Documentation Project — https://tldp.org/
- GeeksforGeeks – Operating Systems — https://www.geeksforgeeks.org/operating-systems/
- GNU Bash Documentation — https://www.gnu.org/software/bash/manual/
- Docker Documentation — https://docs.docker.com/
- Linux Journey — https://linuxjourney.com/
- GeeksforGeeks Operating System Tutorials — https://www.geeksforgeeks.org/operating-systems
- Stanford CS140e Operating Systems — https://cs140e.sergio.bz/

OBJECT-ORIENTED PROGRAMMING WITH JAVA

Program Name	B.Tech CSE (Core)			
Course Name: Object-Oriented Programming with Java	Course Code	L-T-P	Credits	Contact Hours
	CSE207	3-0-2	4	45
Type of Course:	Major			
Semester:	IV (Fourth Semester)			
Pre-requisite(s):	Problem Solving with C++/Python, Data Structures basics			

Course Description

This course introduces object-oriented programming and modern software development using Java. Students learn Java programming constructs, object-oriented design principles, collections, exception handling, multithreading, file processing, and functional programming features. The course further covers enterprise application development using Spring Boot, RESTful APIs, database integration, API documentation, and modern software architecture. Through hands-on laboratory exercises and projects, students develop industry-relevant skills for building scalable and maintainable Java applications.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply Java programming constructs and object-oriented principles to develop software solutions.	L3 – Apply
CO2	Develop reusable applications using collections, interfaces, generics, and modern Java features.	L3 – Apply
CO3	Implement exception handling, multithreading, and file processing for robust Java applications.	L3 – Apply
CO4	Develop RESTful applications using Spring Boot, database integration, and layered architecture principles.	L4 – Analyze
CO5	Evaluate Java software solutions using appropriate design principles and quality attributes.	L5 – Evaluate

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	–	2	–	–	–	–	–	2

CO2	3	2	3	1	2	–	–	1	–	1	2
CO3	3	3	2	2	3	–	1	–	–	–	2
CO4	2	2	3	1	3	1	–	1	2	1	2
CO5	2	3	2	3	3	2	1	2	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	–	1
CO2	3	3	2	1	2
CO3	3	3	3	2	2
CO4	3	3	2	2	3
CO5	2	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	**Java Fundamentals & Core Programming Concepts**	**No. of Hours: 10**

- Java Platform Architecture: JVM, JDK, JRE
- Java Program Structure and Compilation Process
- Variables, Data Types and Operators
- Control Statements and Loops
- Arrays and String Processing
- Classes, Objects and Encapsulation
- Constructors and Constructor Overloading
- Packages and Access Modifiers
- Wrapper Classes

Unit Number: 2	**Advanced Object-Oriented Design & Modern Java**	**No. of Hours: 12**

	Features	

- Inheritance and Polymorphism
- Abstract Classes and Interfaces
- Object Class Methods
- Generics Fundamentals
- Java Collections Framework
    - List
    - Set
    - Queue
    - Map
- Iterator and ListIterator
- Comparable and Comparator Interfaces
- Lambda Expressions and Functional Interfaces
- Stream API
- SOLID Principles
- Composition vs Inheritance
- Collection Performance Analysis

Unit Number: 3	Exception Handling, Multithreading & File Processing	No. of Hours: 12

- Exception Hierarchy
- Checked and Unchecked Exceptions
- Exception Handling Mechanisms
- Custom Exceptions
- Multithreading Fundamentals
- Thread Lifecycle
- Synchronization
- Executor Framework
- Java I/O Streams
- Java NIO Basics
- File Handling
- Serialization and Deserialization

- Stream-based Data Processing

Unit Number: 4	Enterprise Java Development with Spring Boot	No. of Hours: 11

- Introduction to Enterprise Application Development
- Maven Project Structure and Dependency Management
- Spring Boot Fundamentals
- Dependency Injection and Bean Management
- Layered Architecture
  - Controller Layer
  - Service Layer
  - Repository Layer
- Building RESTful APIs
- CRUD Operations
- Request Mapping
- JSON Processing using Jackson
- Database Connectivity using JDBC and Spring Data JPA
- Validation using Bean Validation API
- API Documentation using Swagger/OpenAPI
- API Testing using Postman
- Authentication and Authorization Concepts (Overview)

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Java Fundamentals & Object-Oriented Programming Toolkit	CO1	L3 – Apply
2	Collections Framework & Modern Java Programming Applications	CO2	L3 – Apply
3	Exception Handling, Multithreading & File Processing Framework	CO3	L4 – Analyze
4	Spring Boot REST API & Database Integration Project	CO4	L4 – Analyze
5	Capstone: Smart Campus Management REST Service	CO1–CO5	L5 – Evaluate

Lab Assignments

Unit 1 Lab Assignment: Java Fundamentals & Object-Oriented Programming Toolkit	**CO1**
Problem Statement	Design a Java-based toolkit that solves computational problems using structured programming and object-oriented concepts.
Sub-Problems	• Implement arithmetic and logical operations • Develop menu-driven applications • Perform array processing operations • Implement string manipulation utilities • Create classes and objects • Implement constructors and constructor overloading • Organize programs using packages • Generate structured reports using modular code
Unit 2 Lab Assignment: Collections Framework & Modern Java Programming Applications	**CO2**
Problem Statement	Develop a reusable Java application utilizing collections, generics, lambda expressions, and stream processing.
Sub-Problems	• Implement inheritance and polymorphism • Create abstract classes and interfaces • Develop generic classes and methods • Use ArrayList, HashSet, Queue and HashMap • Implement Comparable and Comparator • Traverse collections using Iterators • Implement Lambda Expressions • Perform filtering and aggregation using Stream API • Compare collection performance for different use cases
Unit 3 Lab Assignment: Exception Handling, Multithreading & File Processing Framework	**CO3**
Problem Statement	Develop a Java framework for concurrent processing and file management.
Sub-Problems	• Handle checked and unchecked exceptions

	• Create custom exceptions • Develop multithreaded applications • Implement synchronization mechanisms • Utilize Executor Framework • Read and write files using Java I/O • Process files using NIO APIs • Implement serialization and deserialization • Generate execution logs and performance reports
Unit 4 Lab Assignment: Spring Boot REST API & Database Integration	**CO4**
Problem Statement	Develop an enterprise-grade RESTful application using Spring Boot and relational database integration.
Sub-Problems	• Create Maven-based Spring Boot projects • Design Controller-Service-Repository architecture • Implement CRUD REST APIs • Handle HTTP GET, POST, PUT and DELETE requests • Integrate MySQL database using Spring Data JPA • Process JSON request and response objects • Implement API validation • Implement exception handling for APIs • Generate API documentation using Swagger • Test APIs using Postman • Compare monolithic and layered architectures
Capstone Lab Assignment: Smart Campus Management REST Service	**CO1, CO2, CO3, CO4, CO5**
Problem Statement	Design and implement a complete Java-based campus management system that integrates object-oriented programming, database connectivity and modern Java application development.
Sub-Problems	• Develop Student Management Module • Develop Faculty Management Module • Develop Course Registration Module • Implement role-based authentication concepts • Design REST APIs using Spring Boot • Integrate MySQL database using JPA • Implement validation and exception handling • Develop reusable object-oriented components • Generate Swagger API documentation • Test APIs using Postman • Evaluate system performance, maintainability and

	scalability

Text and Reference Books

Textbooks:

1. Schildt, H. (2021). Java: The Complete Reference, 12th Edition. McGraw Hill. ISBN: 978-1260463415
2. Horstmann, C. S. (2021). Core Java Volume I – Fundamentals, 12th Edition. Pearson. ISBN: 978-0137673625

Recommended Reference Books:

1. Balagurusamy, E. (2019). Programming with Java: A Primer, 6th Edition. McGraw Hill Education. ISBN: 978-9389811290
2. Deitel & Deitel (2018). Java: How to Program, 11th Edition. Pearson. ISBN: 978-0134743352
3. Sierra, K. & Bates, B. (2022). Head First Java, 3rd Edition. O'Reilly Media. ISBN: 978-1491910771
4. Eckel, B. Thinking in Java, 4th Edition. Pearson Education. ISBN: 978-0131872482

Recommended Online Resources:

- Oracle Java Documentation — https://docs.oracle.com/en/java
- JavaFX Documentation — https://openjfx.io/
- Spring Boot REST API Guides — https://spring.io/guides
- GeeksforGeeks – Java Programming — https://www.geeksforgeeks.org/java/
- NPTEL – Programming in Java — https://nptel.ac.in/courses/106105191
- HackerRank Java Practice — https://www.hackerrank.com/domains/java

PROFESSIONAL & WORKPLACE COMMUNICATION

Program Name	B.Tech CSE (Core)			
Course Name: Professional & Workplace Communication	Course Code	L-T-P	Credits	Contact Hours
		3-0-0	3	45
Type of Course:	AEC			
Semester:	IV (Fourth Semester)			
Pre-requisite(s):	Foundations of Everyday Communication or equivalent			

Course Description

This course focuses on developing practical social, professional, and workplace communication skills through hands-on activities, presentations, digital content creation, collaborative discussions, and real-world communication scenarios. The course emphasizes communication as an employability and professional growth skill rather than a theoretical subject.

Students will learn interpersonal communication, professional writing, digital communication, presentation delivery, storytelling, persuasion, networking, workplace interaction, and interview communication through application-based and portfolio-driven learning. The course integrates speaking, writing, listening, collaboration, and digital communication practices using real-world projects and communication simulations.

By the end of the course, students will develop a professional communication portfolio including presentations, digital profiles, written communication samples, professional content, and recorded speaking activities that demonstrate workplace readiness and communication confidence.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply interpersonal communication and collaboration skills in professional situations.	L3 – Apply
CO2	Create professional digital and written communication content.	L3 – Apply
CO3	Deliver presentations, pitches, and persuasive talks effectively.	L3 – Apply
CO4	Demonstrate professional communication skills in workplace interactions and interviews.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	1	1	2	2	3	1	2	3	3	1	2
CO2	1	3	1	2	3	1	2	2	3	1	2
CO3	1	2	1	1	3	2	2	2	3	2	2
CO4	1	2	2	3	3	2	3	3	3	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	2	3	2	1	1
CO2	2	3	2	1	2
CO3	2	3	3	2	2
CO4	2	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	The Communicator's Toolkit	No. of Hours: 12

- Communication psychology and communication barriers
- Active listening and empathetic communication
- Non-verbal communication and body language
- Emotional awareness and emotional intelligence
- Difficult conversations and conflict resolution
- Cultural communication and diversity awareness
- Giving and receiving constructive feedback
- Practical Activities: listening exercises, communication role-plays, conflict resolution scenarios, cross-cultural interaction activities, peer feedback circles and emotional intelligence reflection tasks

Unit Number: 2	Digital Communication Mastery	No. of Hours: 12

- Professional email writing and etiquette
- Business writing frameworks and formal communication
- Report writing and professional documentation
- LinkedIn profile optimization and networking
- Personal branding and professional identity
- Digital content creation and storytelling
- Professional blogging and online publishing
- Portfolio design and presentation
- Practical Activities: email drafting simulations, resume and cover letter creation, LinkedIn profile development, blog writing and publishing, business proposal writing, digital portfolio development

Unit Number: 3	The Art of Influence	No. of Hours: 11

- Storytelling techniques and narrative structures
- TED-style speaking and presentation strategies
- Slide design and visual communication
- Voice modulation and speaking presence
- Impromptu speaking techniques
- Persuasive communication strategies
- Audience engagement methods
- Question and answer handling techniques
- Practical Activities: presentation analysis and redesign, impromptu speaking rounds, storytelling workshops, pitch presentations, podcast recording activities, public speaking practice sessions

Unit Number: 4	Professional Excellence	No. of Hours: 10

- Professional presence and workplace etiquette
- Networking and relationship building
- Meeting communication and coordination
- Interview communication and response structuring
- Team communication and collaboration
- Leadership communication techniques
- Global workplace communication practices
- Career conversations and professional growth discussions
- Practical Activities: networking simulations, mock interviews, group discussion practice, meeting management exercises, team collaboration challenges, elevator pitch activities

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Interpersonal Communication & Listening Activities	CO1	L3 – Apply
2	Digital Writing & Professional Content Creation	CO2	L3 – Apply
3	Presentation, Storytelling & Public Speaking Tasks	CO3	L3 – Apply
4	Interview, Networking & Workplace Communication Practice	CO4	L4 – Analyze
5	Professional Communication Portfolio Project	CO1–CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Interpersonal Communication & Collaboration	**CO1**
Problem Statement	Develop foundational interpersonal communication skills through listening exercises, conflict resolution scenarios, and cultural interaction activities.
Sub-Problems	• Participate in listening and observation exercises • Engage in communication role-plays based on professional scenarios • Analyze and resolve conflict scenarios through dialogue • Participate in cross-cultural interaction and diversity awareness activities • Engage in peer feedback circles with constructive communication • Complete emotional intelligence reflection tasks and listening pattern analysis
Unit 2 Lab Assignment: Digital Writing & Professional Content Creation	**CO2**
Problem Statement	Create professional digital and written communication content including emails, LinkedIn profiles, blogs, and business documents.
Sub-Problems	• Draft professional emails using proper etiquette and formatting • Create a polished resume and cover letter for a target role • Develop or optimize a LinkedIn profile with professional branding • Write and publish a blog post on a professional topic

	• Draft a business proposal or report following formal writing frameworks • Compile a digital portfolio showcasing written and digital communication samples
Unit 3 Lab Assignment: Presentation, Storytelling & Public Speaking	**CO3**
Problem Statement	Develop public speaking, storytelling, and persuasive communication abilities through structured presentations and speaking activities.
Sub-Problems	• Analyze and redesign an existing presentation for clarity and visual impact • Participate in impromptu speaking rounds on surprise topics • Deliver a structured storytelling presentation with a clear narrative arc • Create and deliver a startup-style pitch presentation • Record a podcast episode on a professional or technical topic • Engage in public speaking practice sessions with peer feedback
Unit 4 Lab Assignment: Workplace Communication & Interview Practice	**CO4**
Problem Statement	Demonstrate professional communication skills in workplace interactions, networking, and interview scenarios.
Sub-Problems	• Participate in networking simulation activities and practice introductions • Engage in mock interviews with structured response techniques • Participate in group discussion practice on professional topics • Manage a meeting simulation as discussion leader or participant • Collaborate in team communication challenge with role-based responsibilities • Deliver an elevator pitch and professional self-introduction video
Capstone Lab Assignment: Professional Communication Portfolio Project	**CO1, CO2, CO3, CO4**
Problem Statement	Compile and present a comprehensive professional

	communication portfolio demonstrating workplace readiness across all communication domains.
Sub-Problems	• Compile LinkedIn profile, resume, cover letters and business writing samples • Include a TED-style recorded talk and a startup pitch presentation • Add blog posts, professional content samples and a podcast episode • Include an elevator pitch video and professional video resume • Document interview performance recordings and networking interaction summaries • Present the portfolio with a reflective commentary on communication growth and workplace readiness

Text and Reference Books

Textbooks:

1. Carnegie, D. (1936). How to Win Friends and Influence People. Simon & Schuster.
2. Gallo, C. (2014). Talk Like TED. St. Martin's Press.

Recommended Reference Books:

1. Handley, A. (2014). Everybody Writes. Wiley.
2. Voss, C. & Raz, T. (2016). Never Split the Difference. HarperBusiness.
3. Zinsser, W. (2006). On Writing Well. Harper Perennial.

Recommended Online Resources:

- TED Talks — https://www.ted.com/talks
- LinkedIn Learning — https://www.linkedin.com/learning
- Coursera Business Communication Courses — https://www.coursera.org
- Harvard Business Review Communication Resources — https://hbr.org
- Toastmasters International — https://www.toastmasters.org/

K.R. MANGALAM UNIVERSITY

COMPETITIVE CODING-II

Program Name	B.Tech CSE (Core)			
Course Name: Competitive Coding II	Course Code	L-T-P	Credits	Contact Hours
	SEET004	2-0-0	2	30
Type of Course:	SEC			
Semester:	IV (Fourth Semester)			
Pre-requisite(s):				

Course Description

This course introduces students to advanced competitive programming concepts and problem-solving techniques using trees, hashmaps, heaps, sliding window algorithms, prefix sums, and bit manipulation. The course focuses on improving students' coding efficiency, logical reasoning, and ability to solve complex algorithmic challenges commonly asked in coding interviews and technical assessment. The course emphasizes hands-on coding practice through advanced data structure implementation and optimization techniques. Students learn tree traversals, binary search trees, heap operations, hashing techniques, and bitwise problem solving strategies used in competitive programming platforms and real-world coding scenarios.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply advanced tree and binary search tree algorithms to solve coding problems.	L3 – Apply
CO2	Develop efficient solutions using hashmaps, heaps, sliding window and prefix sum techniques.	L3 – Apply
CO3	Analyze and optimize coding approaches using priority queues and bit manipulation concepts.	L4 – Analyze
CO4	Evaluate and select appropriate data structures and algorithmic techniques to develop efficient solutions for competitive coding and interview-oriented problems.	L5 – Evaluate

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11

CO1	3	3	2	–	2	–	–	–	–	–	2
CO2	3	3	3	1	3	–	–	1	–	–	2
CO3	2	3	2	2	3	–	1	1	1	1	3
CO4	2	3	2	2	3	–	1	1	1	1	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	2	–	1
CO2	3	3	3	1	2
CO3	3	3	2	2	3
CO4	2	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Trees & Binary Search Trees	No. of Hours: 12

- Introduction to trees: fundamentals of tree data structures, types of trees, and implementation of binary trees.
- Tree traversal techniques: preorder, inorder, postorder, and level order traversal with practical problem-solving approaches.
- Tree-based problem solving: height of tree, checking identical trees, symmetric tree problems, and basic tree properties.
- Advanced binary tree problems: diameter of tree, path sum problems, and left/right view of trees.
- Binary Search Tree concepts: BST implementation, validation, and deletion operations.
- BST applications: converting sorted arrays to BST and finding lowest common ancestor.
- Algorithmic understanding: recursive tree processing, traversal optimization, and structured problem-solving techniques.

Unit Number: 2	HashMaps, Sliding Window & Prefix Sum	No. of Hours: 12

- HashMap fundamentals: introduction to hashing, implementation, and operations such as put, get, and containsKey.

- Hashing-based problem solving: highest frequency character, valid anagram, and finding all anagrams in a string.
- Array and matrix problems using hashing: intersection of arrays and set matrix zeroes.
- Sliding window techniques: longest substring without repeating characters and optimization of substring problems.
- Advanced problem solving: longest consecutive sequence and bulls and cows problem.
- Prefix sum concepts: largest zero sum subarray and counting contiguous subarrays.
- Optimization techniques: reducing time complexity using hashing, sliding window, and prefix sum approaches.

Unit Number: 3	Priority Queues & Heap Algorithms	No. of Hours: 11

- Introduction to priority queues: fundamentals and implementation using heap data structures.
- Heap concepts: understanding min heap and max heap operations and their applications.
- Sorting techniques: in-place heap sort implementation and analysis.
- Selection problems: finding kth largest and kth smallest elements efficiently.
- Heap validation: checking max heap properties and maintaining heap structure.
- Advanced heap problems: top k frequent elements and sliding window maximum.
- Algorithmic focus: efficient data handling, priority-based processing, and optimization techniques.

Unit Number: 4	Bit Manipulation & Competitive Coding Practice	No. of Hours: 10

- Bit manipulation fundamentals: introduction to AND, OR, and XOR operations and their applications.
- Bitwise problem solving: counting set and unset bits using efficient techniques.
- Bit operations: toggling kth bit and checking whether nth bit is set or unset.
- Mathematical applications: checking power of two and power of four using bitwise logic.
- Advanced bit manipulation: counting bits, single number problems, and subset generation using bit masking.
- Application-based problems: finding missing and duplicate numbers using bit manipulation techniques. Competitive coding practice: implementation of optimized solutions, debugging strategies, and solving interview-level coding problems.

Recommended Reference Books

- Introduction to Algorithms by Cormen, Leiserson, Rivest, and Stein
- Cracking the Coding Interview by Gayle Laakmann McDowell
- Elements of Programming Interviews by Adnan Aziz, Tsung-Hsien Lee, and Amit Prakash

Recommended Online Resources

- LeetCode

- CodeChef
- Codeforces
- GeeksforGeeks
- HackerRank
- AtCoder
- TopCoder
- CS50 Problem Solving Resources

MINOR PROJECT – I

Program Name	B. Tech CSE Core			
Course Name: Minor Project – I	Course Code	L-T-P	Credits	Contact Hours
	PRCSE002	0-0-4	2	—
Type of Course:	PROJ			
Semester:	IV (Fourth Semester)			
Pre-requisite(s):	—			

Course Description

The course is designed to provide students with an opportunity to apply theoretical knowledge, technical competencies, and problem-solving abilities to the development of a comprehensive solution for a real-world problem. The course emphasizes innovation, research orientation, design thinking, and interdisciplinary learning through the execution of individual or team-based projects under faculty mentorship. Carrying academic credit as prescribed in the curriculum, the course serves as an essential component for enhancing practical knowledge, technical proficiency, teamwork, and professional preparedness.

The course focuses on experiential and outcome-based learning by engaging students in activities such as problem identification, requirement analysis, literature review, system design, implementation, testing, documentation, and presentation. Students are encouraged to explore emerging technologies, industry-oriented challenges, research problems, entrepreneurial ideas, or socially relevant applications while developing their projects.

Through the project work, students gain hands-on exposure to project planning, technical documentation, collaborative development, research methodologies, and professional ethics. The course also strengthens communication skills, analytical thinking, creativity, and decision-making abilities while fostering independent learning and innovation. By the end of the project, students are expected to demonstrate the successful integration of domain knowledge and technical skills through the development and presentation of a functional prototype, software solution, research outcome, or engineering model aligned with real-world applications and professional standards.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Concise Course Outcome	Bloom's Level
CO1	Analyze and define a real-world problem statement by applying domain knowledge, research understanding, and requirement	L4 – Analyze

	analysis techniques.	
CO2	Design and develop an appropriate project solution using suitable tools, technologies, methodologies, and engineering practices.	L3 – Apply
CO3	Evaluate the performance, functionality, and effectiveness of the developed project through testing, validation, and result analysis.	L5 – Evaluate
CO4	Prepare technical documentation and present the project outcomes effectively while demonstrating teamwork, communication skills, and professional ethics.	L3 – Apply

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	2	3	2	3	1	2	1	2	2	2	2
CO2	3	3	3	2	3	2	2	2	2	3	2
CO3	3	2	3	2	3	1	1	2	2	2	2
CO4	1	2	2	2	2	2	2	3	3	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	2	3	2	1
CO2	3	3	2	2	3
CO3	2	2	3	3	2
CO4	2	2	1	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

1. Objectives

All student projects must:

- Employ industry-relevant technologies and tools.

- Reflect a well-defined development or research process from ideation to final deployment or publication.
- Promote innovation and real-world problem-solving.
- Deliver meaningful outcomes such as a functional application, published research, or validated prototype.

2. Project Scope and Categories

Projects must fall under one of the following approved categories:

S.N	Category	Description
A	**Research-Oriented Projects**	Must address futuristic domains like Generative AI (GANs, Transformers, LLMs), cybersecurity, quantum computing, etc. Multidisciplinary Research Integration is encouraged. Requires mandatory publication in a reputed journal or international conference before final submission. Supervised by internal faculty mentor.
B	**Industry-Based Projects**	Based on real-world industry challenges with practical applications. Must demonstrate full implementation, deployment, and usability. Students work under external and faculty mentorship. External Mentorship Confirmation certificate and a completion certificate from the associated organization are mandatory.
C	**University-Focused Interdisciplinary / Community Solution Projects**	Identify and address real-world challenges or operational gaps within K.R. Mangalam University. Collaboration across disciplines to develop innovative, practical, and sustainable campus solutions. Projects must propose scalable or implementable models.
D	**Start-Up Projects**	Focus on real-world problems and technology-driven solutions with commercialization potential. Must include market research, user validation, business model design, and go-to-market strategy. Outcome should demonstrate technical feasibility and market relevance.
E	**Deep-Tech & Systems (Learning-by-Replication)**	Emphasizes deep technical mastery by replicating foundational technologies. Examples: build your own database engine, compiler (lexer + parser + AST + code-gen), search engine, ML framework, virtual machine, or runtime environment. Must have a minimum viable core plus extensions or optimizations.
F	**Hardware & Software Integration Using IoT /**	End-to-end solutions combining physical hardware (sensors, actuators, microcontrollers) with software systems. A fully

	Embedded Systems	functional hardware prototype is mandatory, including working sensors/actuators, embedded programming, connectivity, and a software interface.

3. Technology Stack and Industry Tools

- Students must select a technology stack aligned with current industry standards, such as MERN, MEAN, Django, Flutter, or cloud-native architectures (AWS, Azure, GCP).
- Projects must incorporate industry-relevant tools and frameworks for version control (Git/GitHub), containerization (Docker), CI/CD (GitHub Actions, Jenkins), and agile development practices.
- Use of modern development environments, APIs, and deployment platforms is strongly encouraged to ensure scalability, maintainability, and real-world applicability of the solution.

4. Formation of Groups

4.1 Group Size & Composition

- Optimal Group Size: Groups will consist of 2 students to allow effective collaboration while maintaining manageable group dynamics.
- Skill Diversity: Form groups with a mix of skill sets (e.g., coding, design, research, communication, etc.) to enhance collaboration. Students should be encouraged to select group members based on complementary skills.

4.2 Group Formation Process

- Self-selection: Allow students to form their own groups, encouraging them to choose teammates based on project interests.
- Pre-formed Groups: Alternatively, groups can be assigned by the coordinator to ensure diversity of skills and ideas across all teams.

4.3 Team Finalization Rule

Once teams are formed and registered, no changes in team composition will be allowed under any circumstances.

4.4 Timeline

Week	Activity / Stage
Week 0	Onboarding on Projexa (Project Management Tool); Team formation; Faculty Mentor allotment
Week 1–2	Problem Statement Finalization; Initial Mentor Interactions and discussion
Week 3–4	Synopsis Submission; Project Proposal Evaluation and Approval
Week 5–7	Implementation Sprint 1: Core Modules, MVP Build, Mentor Interaction

Week 8	Development Phase – Part 1: Module-wise implementation begins
Week 9–10	Development Phase – Part 2: Module-wise implementation continues
Week 11–12	Mid Term Evaluation
Week 13–14	Testing and Debugging; Performance evaluation
Week 15–16	Final Report Submission; Project Video (1–2 mins) demonstrating working project; Presentation & Viva Preparation

5. Guidelines for Project Selection

For any project category, students may select their project through either of the following approaches:

1. Faculty-Suggested Problem Statements: Faculty members will provide curated problem statements based on their domain expertise, current research trends, or institutional needs. Students can choose from these problems and discuss with the faculty mentor before finalizing.
2. Student-Proposed Problem Statements: Student teams are encouraged to identify real-world problems based on personal interests, industry trends, or societal challenges. The proposed problem must be original, relevant, and feasible within the given timeline and resources. Final approval is subject to evaluation by the project review committee.

6. Faculty Allocation Process

Faculty-Suggested Problem Statement: If a team selects a problem provided by a faculty member, the team will be allocated to that faculty after peer review and approval by the Project Coordinator.

1. Student-Proposed Problem Statement: If the team proposes its own problem statement, an appropriate faculty mentor will be assigned based on domain expertise after peer review and validation by the Project Coordinator.
2. Inter School Problem Statement: If a team chooses a problem statement proposed by a faculty member from another school/department, the team will be assigned to that faculty member after peer review and approval by the Project Coordinator.

7. Roles and Responsibilities of Faculty Mentors

1. Regular Interactions: Conduct regular meetings with the assigned team(s) through Projexa in online/offline mode. All meeting records, including agenda, minutes, and attendance, must be documented and uploaded on Projexa. A minimum of five (5) interactions is mandatory, with at least two (2) interactions per month.
2. Task Monitoring: Assign tasks with clear deadlines and track the progress and completion status for each team through the project cycle.
3. Evaluation After Interaction: After every interaction, the faculty mentor must evaluate the team's progress and update the evaluation/comments directly on Projexa as part of the meeting record.

4. Meeting Confirmation & Rescheduling: Faculty must respond to meeting requests initiated by student teams. If unable to conduct a scheduled meeting, it should be rescheduled within one week.
5. Justification for Meeting Cancellation: Any cancellation of scheduled team meetings by the mentor must be supported with a valid and documented reason.
6. GitHub Activity Monitoring: Regularly monitor each team's GitHub repository to verify individual contributions and maintain performance records for all team members.

8. Evaluation Metrics

Projects will be evaluated in 3 phases (Total 100 Marks):

- Synopsis Presentation (15 Marks)
- Mid Term Presentation (20 Marks)
- Final Term Presentation (40 Marks)
- Project Mentor Interaction (25 Marks)

Evaluation Stage	Project Mentor (Interaction)	Project Evaluation Committee
Synopsis Presentation (20 marks)	5	15
Mid Term Presentation (30 marks)	10	20
Final Term Presentation (40 marks)	10	40
Total	**25**	**75**
Grand Total: (25 + 75) = 100 Marks		

Project Evaluation Committee Marking Scheme

Evaluation Stage	Criteria	Marks
Synopsis Presentation	Creative / Naive / Real-world Problem	5
	Clarity & Feasibility of Project Objectives	5
	Preparation & Presentation of PPT	5
	Total	**15**
Mid-Term Presentation	Project Efforts – UI / Implementation	10
	Effective Use of Modern Technology Stack	5
	Effective Coding Practices / Use of Git	5
	Total	**20**
Final-Term Presentation	Final Production Demonstration & Completion	20

	w.r.t. Objectives	
	Impact & Use Cases of the Project	10
	Deployment	10
	Total	**40**

9. Policy on Missed Presentation Schedules

Students are strictly required to adhere to the assigned schedules for all project presentations, including the Synopsis, Mid-Term, and Final-Term evaluations.

- Only one rescheduling opportunity will be granted in case a team or individual misses their assigned presentation slot. This rescheduling must be approved in advance (where possible) or justified immediately after the missed session with valid reasons.
- A penalty of 5 marks will be deducted for each missed presentation schedule, irrespective of the evaluation stage.
- Failure to appear even after the rescheduled opportunity will result in zero marks being awarded for that evaluation component.
- This policy ensures fairness, accountability, and professionalism in the evaluation process.

10. Project Documentation Requirements – Stage-Wise

1. Synopsis Stage

To be submitted before the Synopsis Presentation:

- Synopsis Report
- Synopsis PPT
- Mentorship Confirmation Certificate: Must be on industry letterhead, duly signed by the External Mentor.

2. Mid-Term Evaluation Stage

To be submitted during the Mid-Term Presentation:

- Mid Evaluation PPT
- Project Use Case & Deployment Confirmation Certificate: Issued by the Industry Mentor, confirming project relevance and real-world deployment potential.

3. Final-Term Evaluation Stage

To be submitted during or before Final Presentation:

- Final Project Report
- PPT for Final Presentation
- Video Demonstration (1–2 minutes): Should clearly document key features of the project, execution workflow or demonstration of functionality, and real-world impact or use-case. To be submitted in MP4 format or via a shared video link (as per instructions).

- Plagiarism Certificate: Must be compulsorily appended to the Final Project Report. Should be obtained using institutional or approved plagiarism checking tools.

11. Plagiarism & Ethical Guidelines

- Maximum allowable plagiarism: ≤ 15%
- Check plagiarism using open-source tools and proper citations.
- No reuse of previous projects.
- Intellectual property rights (if applicable).
- Compulsory Plagiarism Certificate to be appended in the report file.

12. Project Outcome Requirements for Each Project Category

Project Category	Desired Outcome
Industry-Based Project	Project completion certificate from Industry Mentor; Working prototype/application demonstration
Research-Based Project	Research paper submission in conference/journal; Technical report with findings and implementation
University-Focused / Community Solution Projects	Faculty Mentor approval certificate; Functional solution addressing university/community problem
Start-Up Project	Basic business idea validation or mentorship proof; Functional MVP or prototype demonstration
Deep-Tech & Systems (Learning-by-Replication)	Simplified working model/prototype demonstrating core functionality of the selected system
Hardware & Software Integration (IoT / Embedded Systems)	Functional hardware prototype with sensor/actuator integration; Basic software interface/dashboard demonstrating system operation

Semester: V

SOFTWARE ENGINEERING & SYSTEM DESIGN

Program Name	B.Tech CSE (Core)			
Course Name: Software Engineering & System Design	Course Code	L-T-P	Credits	Contact Hours
	CSE301	3-0-2	4	45
Type of Course:	Major			
Semester:	V (Fifth Semester)			
Pre-requisite(s):	Knowledge of Programming Fundamentals, Object-Oriented Programming, and Data Structures.			

Course Description

This course introduces modern software engineering principles and system design practices used in the software industry for developing scalable, maintainable, reliable, and collaborative software systems. The course focuses on the complete software development lifecycle (SDLC), agile methodologies, requirement engineering, software architecture, API design, database design, testing, DevOps basics, and scalable system design concepts.

Students will learn how modern software products are planned, designed, developed, tested, deployed, and maintained using industry-standard workflows and tools. The course emphasizes practical implementation through Git-based collaboration, UML modelling, REST API design, low-level and high-level system design, testing automation, CI/CD workflows, cloud-native thinking, and real-world case studies inspired by industry interview and placement scenarios.

The laboratory component focuses on collaborative software development, UML modelling, architecture visualization, API development, Git workflows, software testing, scalable system design exercises, and deployment-oriented mini projects using modern open-source tools and platforms.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply agile software engineering practices.	L3 – Apply
CO2	Analyze software designs and architecture decisions.	L4 – Analyze
CO3	Develop and evaluate tested software systems.	L5 – Evaluate
CO4	Design scalable backend systems and APIs.	L6 – Create

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	2	2	2	1	2	1	1	3	2	2	2
CO2	3	3	3	2	2	1	1	1	1	1	2
CO3	3	3	3	3	3	1	2	2	2	2	2
CO4	3	3	3	2	3	1	1	2	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	3	2	1	2
CO2	3	3	3	2	3
CO3	3	3	3	2	3
CO4	2	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Agile Software Engineering & Collaborative Development	No. of Hours: 12

- **Introduction to Software Engineering:** software lifecycle, software quality and modern development practices
- **Agile & Scrum Fundamentals:** sprint planning, backlog management, stand-up meetings and iterative development
- **Requirement Engineering:** functional requirements, non-functional requirements, user stories and acceptance criteria
- **Version Control & Collaboration:** Git basics, branching, merging, pull requests and GitHub workflows
- **Software Documentation:** README files, API documentation and issue tracking basics
- **Software Ethics & Open-Source Practices:** software licensing, responsible development and collaborative coding ethics
- **Project Planning Basics:** task estimation, milestones and software project tracking

Unit Number: 2	Software Design, APIs & Database Design	No. of Hours: 12

- **Software Design Principles:** modularity, abstraction, cohesion, coupling and SOLID principles
- **UML Modelling:** use case, class, sequence and activity diagrams
- **Object-Oriented Design:** reusable components and class relationships
- **REST API Design:** HTTP methods, status codes, request-response design, validation and error handling
- **API Security Basics:** authentication, authorization and JWT concepts
- **Database Design:** ER diagrams, normalization and relational schema design
- **Design Patterns:** Singleton, Factory and MVC patterns through practical implementation
- **Introduction to Software Architecture:** layered architecture and microservices overview

Unit Number: 3	Software Testing, Debugging & Deployment	No. of Hours: 11

- **Software Testing Fundamentals:** unit testing, integration testing and regression testing
- **Test Case Design:** boundary value analysis and equivalence partitioning
- **Automated Testing:** PyTest/JUnit basics and API testing using Postman
- **Debugging & Logging:** debugging workflows, exception handling and logging practices
- **Code Quality & Refactoring:** identifying code smells and improving maintainability
- **CI/CD Basics:** automated build and deployment workflows using GitHub Actions
- **Containerization Basics:** Docker containers and deployment concepts
- **Software Maintenance & Technical Debt Basics**

Unit Number: 4	System Design & Scalable Applications	No. of Hours: 10

- **Introduction to System Design:** scalability, reliability, availability and fault tolerance
- **Client-Server Architecture & Backend Systems**
- **Caching Concepts:** caching basics and Redis overview
- **Database Scalability Basics:** indexing, replication and simple sharding concepts
- **Asynchronous Systems:** message queues and event-driven workflow basics
- **Performance Optimization:** API performance, bottlenecks and scalability analysis
- **System Design Case Studies:** URL shortener, notification system and online booking system
- **Security & Responsible System Design:** secure APIs, privacy and reliability considerations

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Agile Team Collaboration & Git Workflow	Git, GitHub, Trello	L3
2	UML & Software Design Analyzer	Draw.io, StarUML	L4
3	REST API Development with Validation & Security	FastAPI/Node.js, Postman	L4
4	Database Design & Optimization System	MySQL, PostgreSQL	L3
5	Automated Testing & Bug Fixing Lab	PyTest/JUnit, Postman	L5
6	Code Refactoring & Design Improvement Lab	VS Code, SonarLint	L5
7	Docker-Based Deployment & Performance Analysis	Docker, GitHub Actions	L4
8	Capstone: Smart Service Booking System	FastAPI/Node.js, MySQL, Docker	L5

Lab Assignments

Unit 1 Lab Assignment: Agile Team Collaboration & Git Workflow System	**CO1**
Problem Statement	Simulate a collaborative software development workflow using agile practices, version control systems and project management tools.
Sub-Problems	• Create user stories and software requirements for a small application • Plan sprint tasks and maintain backlog items using Trello • Create Git repositories and implement branching workflows • Perform pull requests, merging and conflict resolution • Track project progress using issue tracking and Kanban boards • Prepare project README and software documentation files
Unit 2 Lab Assignment: Requirement Analysis & UML Design System	**CO2**
Problem Statement	Design software models and architecture diagrams for a real-world software application.
Sub-Problems	• Identify functional and non-functional requirements

	• Create use case diagrams for system workflows • Design class diagrams for software modules • Create sequence diagrams for API interactions • Analyze software design using cohesion and coupling concepts • Compare modular and tightly coupled software designs
Unit 3 Lab Assignment: Code Refactoring & Software Quality Analysis	**CO3**
Problem Statement	Analyze and improve software quality by refactoring inefficient and poorly structured codebases.
Sub-Problems	• Identify code smells and redundant code segments • Refactor code using modular programming principles • Improve readability and maintainability of code • Analyze software quality using static analysis tools • Compare original and improved code structures • Evaluate maintainability and software quality improvements
Unit 4 Lab Assignment: Docker Deployment & Performance Benchmarking	**CO4**
Problem Statement	Containerize and benchmark software applications for deployment and performance analysis.
Sub-Problems	• Create Dockerfiles for backend applications • Build and run application containers • Configure automated build workflows using GitHub Actions • Analyze API response time and application performance • Identify performance bottlenecks during execution • Compare deployment performance under different workloads
Capstone Lab Assignment: Smart Service Booking Platform	**CO1, CO2, CO3, CO4**
Problem Statement	Design and develop a scalable service booking platform for campus services such as appointments, complaints or resource booking using modern software engineering and system design practices.
Sub-Problems	• Gather requirements and define user stories • Design UML diagrams and relational database schema

	• Develop REST APIs with validation and authentication • Implement role-based access workflows • Integrate backend APIs with database systems • Test APIs and debug faulty workflows • Containerize and deploy the application using Docker • Analyze API performance and scalability bottlenecks • Refactor inefficient code modules and improve maintainability • Present final architecture, deployment workflow and testing reports

Text and Reference Books

Textbooks:

1. Software Engineering, Ian Sommerville, 10th Edition, Pearson Education, 2015. ISBN: 9789332582699
2. Clean Code, Robert C. Martin, 1st Edition, Prentice Hall, 2008. ISBN: 9780132350884
3. Head First Design Patterns, Eric Freeman & Elisabeth Robson, 2nd Edition, O'Reilly Media, 2020. ISBN: 9781492078005

Recommended Reference Books:

1. Fundamentals of Software Architecture, Mark Richards & Neal Ford, 1st Edition, O'Reilly Media, 2020. ISBN: 9781492043454
2. System Design Interview – An Insider's Guide, Alex Xu, 2nd Edition, ByteByteGo Publications, 2022. ISBN: 9798981339929
3. The Pragmatic Programmer, David Thomas & Andrew Hunt, 20th Anniversary Edition, Addison-Wesley Professional, 2019. ISBN: 9780135957059
4. Software Engineering: A Practitioner's Approach, Roger S. Pressman & Bruce R. Maxim, 9th Edition, McGraw-Hill Education, 2019. ISBN: 9781259872976
5. API Design Patterns, JJ Geewax, 1st Edition, Manning Publications, 2021. ISBN: 9781617295850

Recommended Online Resources:

- GitHub Skills — https://skills.github.com/
- FastAPI Documentation — https://fastapi.tiangolo.com/
- Postman Learning Center — https://learning.postman.com/
- Docker Documentation — https://docs.docker.com/
- NPTEL Software Engineering — https://nptel.ac.in/courses/106105182
- Refactoring Guru — https://refactoring.guru/
- Roadmap.sh System Design — https://roadmap.sh/system-design
- GeeksforGeeks Software Engineering — https://www.geeksforgeeks.org/software-engineering/
- ByteByteGo System Design Resources — https://bytebytego.com/
- freeCodeCamp API Development Course — https://www.freecodecamp.org/learn/back-end-development-and-apis/

K.R. MANGALAM UNIVERSITY

COMPUTER NETWORKS

Program Name	B.Tech. CSE (Core)			
Course Name: Computer Networks	Course Code	L-T-P	Credits	Contact Hours
	CSE302	3-0-2	4	45
Type of Course:	Major			
Semester:	V (Fifth Semester)			
Pre-requisite(s):	Computer fundamentals, operating systems, programming concepts and basic Linux usage.			

Course Description

This course introduces the concepts, architectures and technologies used in modern computer networks and communication systems. The course focuses on network models, routing, switching, wireless communication, network security, network monitoring and modern internet technologies.

Students will learn how modern communication systems enable reliable, scalable and secure data transmission across enterprise and internet-scale infrastructures. The course emphasizes practical understanding of network protocols, packet analysis, network configuration, troubleshooting, wireless communication and network security aligned with current industry requirements.

The laboratory component focuses on hands-on implementation and analysis of networking concepts using Linux networking tools, Cisco Packet Tracer, Wireshark and socket programming environments. Students will gain practical exposure to routing, subnetting, traffic analysis, network security and real-world communication systems.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Explain networking concepts and protocols.	L2 – Understand
CO2	Configure routing and switching networks.	L3 – Apply
CO3	Analyze network traffic and performance.	L4 – Analyze
CO4	Develop secure communication systems.	L3 – Apply
CO5	Evaluate modern networking technologies and security practices.	L5 – Evaluate

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	1	2	-	1	1	1	-	2
CO2	3	3	3	2	3	1	1	1	1	1	2

CO3	3	3	2	3	3	1	2	1	1	1	2
CO4	3	3	3	2	3	2	2	2	2	2	2
CO5	3	3	3	3	3	2	2	2	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	2	1	–	1
CO2	3	3	2	1	2
CO3	3	3	3	2	2
CO4	3	3	2	2	3
CO5	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Unit-wise Syllabus

Unit 1: Fundamentals of Computer Networks & Data Communication

- Introduction to Computer Networks: network types, topology and communication models
- OSI & TCP/IP Models: layered architecture and protocol stack concepts
- Data Communication Fundamentals: signals, bandwidth, latency and transmission media
- Network Devices: hubs, switches, routers, gateways and access points
- IP Addressing & Subnetting: IPv4, IPv6 and CIDR concepts
- ARP, ICMP and DHCP Protocols
- Transport Layer Protocols: TCP and UDP
- Linux Networking Commands: ping, traceroute, netstat and ip/ifconfig

Unit 2: Routing, Switching & Wireless Communication

- Switching Concepts: circuit switching and packet switching
- MAC Addressing & Ethernet Technologies
- VLANs & Network Segmentation
- Routing Concepts & Routing Tables
- Static & Dynamic Routing Basics
- Routing Protocols: RIP, OSPF and BGP overview
- Wireless Networking Fundamentals: Wi-Fi architecture and wireless communication
- Mobile & Wireless Security Basics

- Socket Programming Fundamentals

Unit 3: Network Security, Monitoring & Performance Analysis

- Network Security Fundamentals: firewalls, IDS and VPN concepts
- Authentication & Secure Communication Protocols
- Cryptography Basics in Networking: SSL/TLS and HTTPS
- Packet Sniffing & Traffic Analysis using Wireshark
- Network Monitoring & Troubleshooting Techniques
- Quality of Service (QoS) & Network Performance Metrics
- DNS, HTTP, HTTPS, FTP and SMTP Protocol Analysis
- Introduction to Network Automation Basics

Unit 4: Modern Networking Technologies & Internet Infrastructure

- Software-Defined Networking (SDN) Fundamentals
- Introduction to Network Virtualization
- 5G & Modern Wireless Communication Systems
- Internet of Things (IoT) Communication Basics
- Content Delivery Networks (CDN) & Load Balancing
- Data Center Networking Concepts
- Distributed Communication Systems & Scalability Concepts
- Case Studies: enterprise networks, ISP infrastructure and modern internet architecture

Lab Experiments – CO Mapping Table

S. No.	Experiment Title	Tools / Technologies Used	Mapped CO(s)	Bloom's Level
1	IP Addressing & Packet Communication Analyzer	Cisco Packet Tracer, Linux Commands	CO1, CO2	L3 – Apply
2	Routing, Switching & Socket Communication System	Cisco Packet Tracer, Python Socket Programming	CO2	L3 – Apply
3	Network Security & Traffic Monitoring Toolkit	Wireshark, Linux Firewall Tools	CO3, CO4	L4 – Analyze
4	Wireless Network Analysis & Performance Evaluation	Wireshark, Linux Networking Tools	CO3, CO5	L4 – Analyze
5	Capstone: Intelligent Network Monitoring & Secure Communication System	Python, Wireshark, Linux	CO1, CO2, CO3, CO4, CO5	L5 – Evaluate

Lab Assignments

Unit 1 Lab Assignment: IP Addressing & Packet Communication Analyzer	**CO1**
Tools / Technologies Used	Cisco Packet Tracer, Linux Commands
Problem Statement	Design and analyze IP-based communication systems demonstrating addressing, subnetting and packet transmission techniques.
Sub-Problems	1. Configure IPv4 and IPv6 addressing schemes 2. Perform subnetting and CIDR calculations 3. Simulate packet transmission between hosts 4. Analyze ARP, ICMP and DHCP operations 5. Use Linux networking commands for connectivity testing 6. Visualize packet flow across network devices 7. Generate network communication reports
Concepts Covered	IP addressing, subnetting, Ethernet communication and packet analysis

Unit 2 Lab Assignment: Routing, Switching & Socket Communication System	**CO2**
Tools / Technologies Used	Cisco Packet Tracer, Python Socket Programming
Problem Statement	Develop routing and communication systems demonstrating switching, routing and socket-based networking concepts.
Sub-Problems	1. Configure static and dynamic routing 2. Implement VLANs and network segmentation 3. Simulate routing protocols and packet forwarding 4. Configure wireless network devices and communication 5. Develop TCP and UDP socket communication programs 6. Analyze network latency and routing efficiency 7. Generate routing and communication performance reports
Concepts Covered	Routing, switching, wireless networking and socket programming

Unit 3 Lab Assignment: Network Security & Traffic Monitoring	

Toolkit	**CO3, CO4**
Tools / Technologies Used	Wireshark, Linux Firewall Tools, OpenSSL
Problem Statement	Develop network security and traffic monitoring systems for analyzing communication behavior and secure data transmission.
Sub-Problems	1. Capture and analyze packets using Wireshark 2. Configure firewall rules and access control 3. Analyze HTTP, HTTPS and DNS traffic 4. Implement encrypted communication using SSL/TLS basics 5. Monitor network throughput and latency 6. Detect suspicious traffic and network anomalies 7. Generate traffic analysis and security reports
Concepts Covered	Network security, packet analysis and secure communication

Unit 4 Lab Assignment: Wireless Network Analysis & Performance Evaluation	**CO4, CO5**
Tools / Technologies Used	Wireshark, Linux Networking Tools
Problem Statement	Analyze wireless communication systems and evaluate performance of modern networking technologies.
Sub-Problems	1. Monitor wireless network traffic and signal behavior 2. Analyze latency, throughput and packet loss 3. Evaluate QoS and bandwidth utilization 4. Simulate SDN and virtual network concepts 5. Analyze IoT communication workflows 6. Compare wired and wireless network performance 7. Generate network performance and scalability reports
Concepts Covered	Wireless communication, SDN, QoS and performance evaluation

Capstone Lab Assignment: Intelligent Network Monitoring & Secure Communication System	**CO1, CO2, CO3, CO4, CO5**
Tools / Technologies Used	Python, Wireshark, Linux, Grafana

Problem Statement	Design and implement an intelligent network monitoring and secure communication platform for analyzing network traffic, system performance and secure communication.
Sub-Problems	1. Monitor real-time network traffic and device communication 2. Analyze bandwidth, latency and throughput metrics 3. Detect abnormal traffic and security threats 4. Configure secure communication channels 5. Develop automated network monitoring scripts 6. Visualize network analytics using dashboards and graphs 7. Generate optimization and security recommendations 8. Develop modular and reusable monitoring components
Concepts Covered	Network monitoring, security analysis and scalable communication systems

Text Books

1. Computer Networking: A Top-Down Approach — James F. Kurose & Keith W. Ross, 8th Edition, Pearson, 2021. ISBN: 9780136681557
2. Computer Networks — Andrew S. Tanenbaum & David J. Wetherall, 6th Edition, Pearson, 2021. ISBN: 9780136764052
3. Network Warrior — Gary A. Donahue, 3rd Edition, O'Reilly Media, 2017. ISBN: 9781449387860

Recommended Reference Books

1. TCP/IP Illustrated, Volume 1 — W. Richard Stevens & Kevin R. Fall, 2nd Edition, Addison-Wesley Professional, 2011. ISBN: 9780321336316
2. Practical Packet Analysis — Chris Sanders, 3rd Edition, No Starch Press, 2017. ISBN: 9781593278021
3. Computer Networking Problems and Solutions — Russ White & Ethan Banks, Addison-Wesley Professional, 2018. ISBN: 9780134766072
4. Network Programmability and Automation — Jason Edelman, Scott Lowe & Matt Oswalt, O'Reilly Media, 2018. ISBN: 9781491931258

Recommended Online Resources:

- Cisco Networking Academy — https://www.netacad.com
- Wireshark Documentation — https://www.wireshark.org/docs
- NPTEL – Computer Networks — https://nptel.ac.in/courses/106105183
- Kurose & Ross Resources — https://gaia.cs.umass.edu/kurose_ross/index.php
- GeeksforGeeks – Computer Networks — https://www.geeksforgeeks.org/computer-networks/
- Linux Networking Documentation — https://tldp.org/HOWTO/NET3-4-HOWTO.html

ADVANCED COMMUNICATION AND INTERVIEW MASTERY

Program Name	**B.Tech CSE (Core)**			
Course Name: Advanced Communication and Interview Mastery	**Course Code**	**L-T-P**	**Credits**	**Contact Hours**
		3-0-0	3	45
Type of Course:	AEC			
Semester:	V (Fifth Semester)			
Pre-requisite(s):				

Course Description

This course is designed as an advanced, activity-driven communication and employability program that prepares students for contemporary professional environments and placement processes. The course adopts a practice-first pedagogy in which learners engage in real workplace simulations, group discussions, mock interviews, professional writing tasks, and problem-solving challenges before examining the underlying communication concepts. Emphasis is placed on Indian professional contexts, digital communication practices, cross-cultural interaction, leadership communication, and interview preparedness. Through collaboration with Placement Cells, recruiters, alumni, and industry professionals, learners develop authentic workplace communication competence, critical thinking abilities, and recruiter-ready professional portfolios.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome
CO1	Demonstrate confident self-expression, active listening, and emotional intelligence in professional and workplace communication contexts.
CO2	Produce effective professional writing including emails, reports, applications, cover letters, and digital communication artefacts using audience-centred strategies.
CO3	Perform effectively in presentations, group discussions, interviews, and workplace conversations using structured and persuasive communication techniques.
CO4	Build a credible professional identity through digital presence, cross-cultural communication competence, and workplace professionalism.
CO5	Apply analytical, creative, and problem-solving skills to real workplace and organisational challenges and communicate solutions effectively.

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	2	1	2	1	1	-	1	-	2
CO2	3	2	3	1	3	1	1	1	-	1	2
CO3	2	3	1	3	3	1	2	1	2	-	2
CO4	2	3	3	2	3	2	2	1	1	-	2
CO5	2	3	2	3	2	2	2	2	3	1	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	1	–	1
CO2	3	3	3	2	2
CO3	3	3	2	2	2
CO4	3	3	3	2	3
CO5	2	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Confident Self-Expression and Professional Identity	No. of Hours: 11

Communication in Indian Professional Contexts

- Communication styles in Indian workplaces
- Formal and informal communication patterns
- Hierarchy, tone, and relational communication

Self-Expression and Professional Identity

- Building confidence in workplace interactions
- Introducing oneself in professional contexts
- Authenticity and personal communication style

Communication Barriers

- Language hierarchies and regional influences
- Code-switching and generational communication gaps
- Miscommunication in workplace environments

Active Listening

- Listening for understanding and interpretation
- Clarifying, summarising, and questioning
- Listening in interviews and group tasks

Emotional Intelligence in the Workplace

- Self-awareness and emotional regulation
- Empathy and workplace relationships
- Handling conflict and difficult conversations

Suggested Activities

- Recorded professional introductions
- Workplace communication simulations
- Listening and summarising exercises
- Role-plays on workplace conflict scenarios

Unit Number: 2	Professional Writing in Workplace Contexts	No. of Hours: 12

Email Writing and Professional Etiquette

- Writing clear and concise emails
- Subject lines, tone, and structure
- Professional greetings and follow-up communication

Business Writing

- Principles of effective workplace writing
- Writing memos, notices, and formal correspondence
- Audience awareness and purpose-driven writing

Report Writing

- Structure and organisation of reports
- Writing observations, findings, and recommendations
- Executive summaries and workplace documentation

Cover Letters and Applications

- Writing job and internship applications
- Presenting authentic professional experiences
- Resume and cover letter integration

AI and Professional Writing

- Ethical use of AI writing tools
- Editing and proofreading strategies
- Maintaining authenticity and contextual relevance

Suggested Activities

- Email rewriting workshops
- Incident and workplace report writing
- Job application and cover letter drafting
- AI-assisted editing and peer review sessions

Unit Number: 3	Digital Presence and Cross-Cultural Communication	No. of Hours: 11

Digital Professional Identity
- LinkedIn profiles and networking
- Professional email communication
- Building an online professional image

Online Communication
- Virtual meeting etiquette
- Communication through Teams, Slack, and WhatsApp
- Managing professional communication online

Cross-Cultural Communication
- Communication across Indian cultural diversity
- High-context and low-context communication
- Global workplace communication practices

Professional Networking
- Building professional relationships
- Writing networking messages and follow-ups
- Communication in professional communities

Portfolio Development
- Curating professional communication artefacts
- Reflective communication statements
- Preparing recruiter-ready portfolios

Suggested Activities
- LinkedIn profile audits
- Virtual meeting simulations
- Cross-cultural communication case studies
- Portfolio presentation workshops

Unit Number: 4	Problem-Solving and Creative Thinking at Work	No. of Hours: 10

Problem Identification and Analysis
- Defining workplace problems
- Root cause analysis techniques

- Problem framing and analytical thinking

Structured Problem-Solving

- PDCA and design thinking approaches
- Applying frameworks to workplace scenarios
- Decision-making and solution development

Creative Thinking and Innovation

- Creativity in professional environments
- Constraint-based innovation and frugal thinking
- Generating and evaluating ideas

Solution Communication

- Presenting solutions persuasively
- Data-supported communication strategies
- Workplace pitching and recommendation delivery

Workplace Readiness and Collaboration

- Team-based problem-solving
- Managing ambiguity and pressure
- Communication in innovation and project teams

Suggested Activities

- Case-study analysis and solution design
- Community and campus problem-solving projects
- Innovation and idea-generation challenges
- Recruiter-evaluated solution presentations

Text and Reference Books

Textbooks:

- Carnegie, Dale. *How to Win Friends and Influence People*. Simon & Schuster, 1936.
- Gallo, Carmine. *Talk Like TED: The 9 Public-Speaking Secrets of the World's Top Minds*. St. Martin's Press, 2014.
- Patterson, Kerry, et al. *Crucial Conversations: Tools for Talking When Stakes Are High*. McGraw-Hill Education, 2012.
- Rosenberg, Marshall B. *Nonviolent Communication: A Language of Life*. Puddle Dancer Press, 2015.
- Voss, Chris, and Tahl Raz. *Never Split the Difference: Negotiating as If Your Life Depended on It*. Harper Business, 2016.
- Zinsser, William. *On Writing Well: The Classic Guide to Writing Nonfiction*. Harper Perennial, 2006.

Recommended Reference Books:

- Cuddy, Amy. *Presence: Bringing Your Boldest Self to Your Biggest Challenges*. Little, Brown and Company, 2015.
- Duarte, Nancy. *HBR Guide to Persuasive Presentations*. Harvard Business Review Press, 2012.
- Heath, Chip, and Dan Heath. *Made to Stick*. Random House, 2007.

- Reynolds, Garr. *Presentation Zen*. New Riders, 2011.
- Turkle, Sherry. *Reclaiming Conversation: The Power of Talk in a Digital Age*. Penguin Press, 2015.
- Brown, Brené. *Dare to Lead*. Random House, 2018.

Open Educational Resources:

- TED Talks: Public speaking, storytelling, and leadership communication resources.
- LinkedIn Learning: Professional communication and employability courses.
- Coursera: Business Communication and Interview Skills courses.
- Harvard Business Review Communication Collection.
- BBC Learning English: Workplace communication and speaking activities.
- NPTEL Soft Skills and Personality Development Modules.

Learning Experience:

By the end of the course, students will be able to communicate effectively and confidently in placement processes, workplace interactions, and professional networking environments. They will demonstrate competence in professional writing, group discussions, interviews, presentations, and digital communication while also developing analytical thinking, creativity, and problem-solving abilities. Learners will create a recruiter-ready professional portfolio containing written communication samples, recorded presentations, interview performances, LinkedIn profiles, and workplace problem-solving projects that showcase authentic professional growth and employability readiness.

K.R. MANGALAM UNIVERSITY

COMPETITIVE CODING III

Program Name	B.Tech CSE (Core)			
Course Name: Competitive Coding III	Course Code	L-T-P	Credits	Contact Hours
	SEET005	2-0-0	2	30
Type of Course:	SEC			
Semester:	V (Fifth Semester)			
Pre-requisite(s):				

Course Description

This course introduces students to advanced competitive programming concepts and algorithmic problem-solving techniques involving recursion, backtracking, greedy algorithms, graph theory and dynamic programming. The course focuses on enhancing coding efficiency, optimization strategies and logical reasoning for technical interviews and coding contests.

The course emphasizes hands-on learning through advanced coding problems, graph traversal algorithms, shortest path techniques and dynamic programming optimization methods. Students develop efficient problem-solving approaches commonly used in competitive programming platforms.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply recursion, backtracking and greedy algorithms to solve computational problems.	L3 – Apply
CO2	Develop efficient graph-based solutions using traversal and shortest path algorithms.	L3 – Apply
CO3	Analyze and optimize dynamic programming approaches for advanced coding problems.	L4 – Analyze
CO4	Evaluate competitive programming strategies using advanced algorithmic techniques.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	1	1	1	1	1	2	2	3	3	1	2
CO2	1	1	2	1	2	1	2	2	3	1	2

CO3	1	1	1	1	2	2	2	3	3	1	2
CO4	1	2	2	1	2	2	2	2	3	1	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	3	2	2	–	1
CO2	3	3	3	1	2
CO3	3	3	2	2	3
CO4	2	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	**Advanced Recursion & Backtracking**	**No. of Hours: 12**

- Introduction to advanced recursion techniques and problem-solving approaches.
- Recursive generation problems: printing all subsets and permutations using recursion.
- Subset variations: finding all unique subsets using recursive approaches and handling duplicates.
- Combination-based problems: combination, combination sum, and combination sum-II.
- Backtracking problems: generate parentheses, subset-II, and Sudoku solver.
- Maze and board problems: rat in a maze, all maze paths, and N-Queens problem.
- Optimization techniques: improving recursive solutions using pruning and backtracking strategies.

Unit Number: 2	**Greedy Algorithms & Optimization**	**No. of Hours: 12**

- Introduction to greedy algorithms: concept of local optimization and decision-making strategies
- Basic greedy problems: assign cookies, array partition, and can place flowers.
- Real-world greedy applications: lemonade change and activity selection problem.
- Optimization problems: minimum platform and coin change using greedy techniques.
- Advanced greedy problems: maximum chunks to make sorted and maximum chunks to make sorted-II.
- Introduction to optimization problems: basic understanding of 0/1 knapsack problem.
- Algorithmic focus: selecting optimal solutions using greedy strategies and analyzing efficiency.

Unit Number: 3	Graph Theory & Graph Algorithms	No. of Hours: 11

- Introduction to graphs: fundamentals and real-world applications of graph data structures.
- Graph representation: adjacency matrix and adjacency list implementation.
- Graph traversal techniques: Breadth First Search (BFS) and Depth First Search (DFS).
- Graph problem solving: connected components and cycle detection in graphs.
- Path-based problems: path existence, all paths from source to destination, and number of islands.
- Grid-based graph problems: number of provinces, flood fill, and number of closed islands.
- Minimum Spanning Tree algorithms: Prim's and Kruskal's algorithms.
- Shortest path algorithms: Dijkstra and Bellman-Ford algorithms.
- Algorithmic understanding: traversal optimization, graph modeling, and efficient pathfinding techniques.

Unit Number: 4	Dynamic Programming & Advanced Coding	No. of Hours: 10

- Introduction to dynamic programming: optimization concepts and overlapping subproblems.
- DP techniques: memoization and tabulation methods for efficient computation.
- Basic DP problems: staircase, minimum cost climbing stairs, and counting bits.
- Advanced DP problems: house robber, goldmine, and path with maximum gold.
- Grid and path problems: unique path, minimum path sum, and target sum subset problems.
- Optimization problems: coin change, jump game, and jump game-II.
- String-based DP: longest common subsequence using recursion, memoization, and tabulation.
- Advanced string problems: longest palindromic substring and longest palindromic subsequence.
- Coding practice: implementing optimized dynamic programming solutions for interview and competitive coding scenarios.

Text and Reference Books

Textbooks:

1. Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C., *Introduction to Algorithms*, MIT Press, 4th Edition, 2022.
2. Karumanchi, N., *Data Structures and Algorithms Made Easy*, CareerMonk Publications, 5th Edition, 2017.
3. Halim, S., Halim, F., and Effendy, S., *Competitive Programming 4*, Lulu Publication, 2022.
4. Skiena, S.S., *The Algorithm Design Manual*, Springer, 3rd Edition, 2020.

Recommended Reference Books:

- Competitive Programming by Steven Halim and Felix Halim
- Cracking the Coding Interview by Gayle Laakmann McDowell
- Elements of Programming Interviews by Adnan Aziz, Tsung-Hsien Lee and Amit Prakash

Recommended Online Resources:

- LeetCode – https://leetcode.com
- CodeChef – https://www.codechef.com
- Codeforces – https://codeforces.com
- GeeksforGeeks – https://www.geeksforgeeks.org
- HackerRank – https://www.hackerrank.com
- InterviewBit – https://www.interviewbit.com
- Coding Ninjas Studio – https://www.codingninjas.com/studio
- AtCoder – https://atcoder.jp

SUMMER INTERNSHIP-II (ASSESSMENT)

Program Name	B.Tech. CSE (Core)			
Course Name: Summer Internship-II	Course Code	L-T-P	Credits	Contact Hours
	SICSE003	0-0-4	2	30
Type of Course:	INT			
Semester:	V (Fifth Semester)			
Pre-requisite(s):				

Course Description

The Summer Internship course is designed to provide students with practical exposure to real-world professional environments by integrating academic learning with industry-oriented experiences. Conducted during the summer break, the internship enables students to apply theoretical concepts, technical knowledge, and problem-solving skills in practical situations across industry, research, entrepreneurial, or organizational settings. Carrying a weightage of 2 academic credits, the course serves as a mandatory component of the curriculum and focuses on enhancing professional competence, workplace ethics, communication abilities, and industry readiness.

The course emphasizes experiential learning through active participation in technical, managerial, research, or innovation-based tasks under professional supervision. Students gain exposure to real-world practices, organizational workflows, and collaborative work environments, helping them develop self-learning abilities, critical thinking, adaptability, and professional responsibility. The internship duration of 6–8 weeks allows students to strengthen the connection between academic knowledge and practical applications while preparing them for future career opportunities and higher professional engagement.

The internship also encourages students to explore emerging technologies, modern development practices, organizational culture, and collaborative problem-solving techniques. Through continuous mentor interactions, documentation, and presentations, students develop professional ethics, leadership qualities, communication abilities, and industry-oriented technical competencies. By the end of the internship, students are expected to demonstrate practical understanding, professional growth, and the ability to connect academic concepts with real-world applications effectively.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply academic knowledge and technical skills to perform assigned tasks in a professional work environment.	L3 – Apply
CO2	Demonstrate professional ethics, communication skills, and teamwork during internship activities.	L3 – Apply
CO3	Analyze real-world industry practices, organizational workflows, and	L4 – Analyze

	problem-solving approaches observed during the internship.	
CO4	Prepare and present a comprehensive internship report reflecting learning experiences, outcomes, and professional growth.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	2	-	3	-	1	2	1	2	2
CO2	-	-	1	-	1	1	3	3	3	2	2
CO3	2	3	2	2	2	1	1	2	1	2	3
CO4	1	1	-	1	1	-	1	2	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Software Solutions
CO1	2	3	1	1	1
CO2	1	2	2	1	1
CO3	2	2	3	2	1
CO4	1	2	2	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Objectives

The Summer Internship aims to:

- Provide students with hands-on industry exposure and practical learning experiences.
- Bridge the gap between academic concepts and real-world applications.
- Enhance technical competency, communication skills, teamwork, and professional ethics.
- Develop analytical thinking, adaptability, and workplace readiness.
- Familiarize students with organizational workflows, project execution, and modern tools/technologies.
- Encourage experiential learning through research, development, innovation, and problem-solving activities.
- Strengthen documentation, reporting, and presentation skills through structured evaluation processes.

Internship Scope and Categories

Students may opt for any one of the following approved internship categories:

External (Outside KRMU) Tracks

- Track 1: Industrial Internship
- Track 2: Government / Research Institution
- Track 3: Start-ups

Internal (Inside KRMU) Tracks

- Track 4: Bootcamp
- Track 5: University Specific Projects
- Track 6: Ideas 4.0
- Track 7: Global Certifications

Track 1: Industrial Internship

- Internship in reputed companies, IT firms, software industries, consulting organizations, or technical enterprises.
- Students will work on real-world industry problems, software development, testing, analytics, cloud computing, cybersecurity, AI/ML, or other domain-specific tasks.
- Mandatory submission of internship completion certificate and mentor verification.

Track 2: Government / Research Institution Internship

- Internship in government departments, public sector organizations, research labs, or academic/research institutions.
- Students may contribute to research activities, data analysis, technical documentation, simulations, or development projects.
- Exposure to research methodologies and institutional workflows is encouraged.

Track 3: Start-Up Internship

- Internship in start-ups or entrepreneurial organizations focusing on innovation and product development.
- Students may participate in prototype development, product design, business analytics, marketing technology, or software implementation.
- Encourages innovation, adaptability, and entrepreneurial thinking.

Track 4: Bootcamp (Internal – KRMU)

- Intensive skill-based training programs organized by the university.
- Focus on modern technologies such as Data Science, AI/ML, Web Development, Cloud Computing, DevOps, Cybersecurity, IoT, etc.
- Includes hands-on assignments, assessments, and mini projects.

Track 5: University Specific Projects

- Students work on projects addressing university-specific requirements or operational enhancements.
- Encourages interdisciplinary collaboration and solution development for institutional improvement.
- Projects may include automation systems, dashboards, portals, analytics platforms, or smart campus solutions.

Track 6: Ideas 4.0

- Innovation and idea development program focused on problem-solving and entrepreneurial creativity.
- Students develop proof-of-concept models, prototypes, or innovative technical solutions.
- Encourages design thinking, creativity, and innovation-driven learning.

Track 7: Global Certifications

- Students may complete internationally recognized certifications aligned with their specialization.
- Certifications may include cloud platforms, programming technologies, AI/ML tools, cybersecurity, data analytics, or software development frameworks.
- Practical implementation and assessment-based learning are mandatory.

Technology Exposure and Professional Tools

Students are encouraged to gain exposure to:

- Industry-standard software tools and development environments.
- Version control systems such as Git/GitHub.
- Cloud platforms such as AWS, Azure, or GCP.
- Data analytics, AI/ML frameworks, or software engineering tools.
- Collaboration and project management platforms.
- Documentation, reporting, and presentation tools.

Use of emerging technologies and industry practices is strongly encouraged to improve professional readiness and technical competency.

Internship Duration and Timeline

Week	Activity / Stage
Week 0	Internship Registration on Projexa and Mentor Allocation
Week 1	Joining Organization / Orientation / Goal Setting
Week 2–3	Task Understanding and Initial Learning
Week 4–5	Practical Work / Technical Assignments / Development Activities
Week 6	Mid Internship Review and Mentor Interaction
Week 7	Advanced Task Execution / Documentation
Week 8	Final Submission, Report Preparation, Presentation & Viva

Internship Guidelines

Internship Approval

- Students must obtain prior approval from the department before joining any internship organization/program.
- Internship domain must align with the student's specialization or technical learning objectives.

Attendance Requirement

- Minimum attendance and active participation throughout the internship duration are mandatory.
- Attendance records must be certified by the organization/mentor.

Mentor Allocation

- Each student will be assigned a faculty mentor for monitoring and evaluation.
- External internships will additionally require an industry mentor/supervisor.

Progress Monitoring

- Students must participate in regular mentor interactions through Projexa.
- Faculty mentors will monitor progress, documentation, and learning outcomes.

Professional Conduct

Students are expected to:

- Follow workplace ethics and discipline.
- Maintain confidentiality of organizational data.
- Demonstrate professionalism, punctuality, and responsibility.
- Respect organizational policies and reporting structures.

Roles and Responsibilities of Faculty Mentors

Regular Interactions

- Conduct regular mentor interactions through Projexa in online/offline mode.
- Monitor student progress and provide academic guidance.
- A minimum of four (4) mentor interactions is mandatory.

Progress Evaluation

- Review assigned tasks, internship activities, and technical learning outcomes.
- Maintain interaction records and evaluation comments on Projexa.

Report Verification

- Verify internship report authenticity and learning outcomes.
- Ensure proper documentation and completion of all required submissions.

Presentation Evaluation

- Assess the student's presentation, technical understanding, communication skills, and professional growth.

Evaluation Criteria

Each student must be evaluated on the following documents submitted after completing their internship/certification:

Deliverable	Description	Marks
Summer Internship Report File	A detailed Summer Internship report in the prescribed format including internship objectives, work completed, learning outcomes, and a signed one-page logbook with progress details endorsed by the industry mentor.	20
Attendance Record	Signed and stamped attendance record certified by the industry mentor/organization.	10
Hard Bound Completion	Submission of hard-bound copy of the internship	10

Certificate	completion certificate issued by the organization/industry.	
Mentor Interaction through Projexa	Participation in four scheduled mentor interaction meetings conducted through Projexa for progress monitoring and evaluation.	40
Viva / Final Presentation	Final presentation and viva demonstrating internship work, technical learning, outcomes, and professional experience gained during the internship.	20
Total		**100**

Rubrics

Assessment Component	Excellent (9–10)	Good (7–8)	Satisfactory (5–6)	Needs Improvement (0–4)
Summer Internship Report File (20 Marks)	Report is highly organized, comprehensive, professionally formatted, and includes detailed learning outcomes, activities, and signed logbook with complete progress details.	Report is well-structured with relevant content and proper documentation; minor details may be missing.	Report covers basic internship details but lacks depth, clarity, or proper formatting.	Report is incomplete, poorly organized, or missing significant details/logbook.
Attendance Record (10 Marks)	Attendance is fully signed and stamped with complete authenticity and timely submission.	Attendance record is complete with minor formatting/documentation issues.	Attendance record is partially complete or submitted late.	Attendance proof is incomplete, unclear, or not authenticated.
Hard Bound Completion Certificate (10 Marks)	Properly submitted hard-bound certificate with all required details and professional presentation.	Certificate submitted with minor issues in formatting or presentation.	Certificate submitted but lacks proper binding or some required details.	Certificate not submitted or significantly incomplete.
Mentor Interaction through Projexa (40 Marks)	Attended all mentor meetings actively with excellent	Participated regularly with good interaction and satisfactory progress updates.	Limited participation or inconsistent progress	Poor participation or absence in mentor

	progress updates, responsiveness, and task completion.		reporting during meetings.	interactions.
Viva / Final Presentation (20 Marks)	Presentation is clear, confident, technically strong, and demonstrates excellent understanding of internship work and outcomes.	Presentation is well-delivered with good understanding and communication skills.	Presentation communicates basic work done but lacks clarity or confidence.	Presentation is unclear, incomplete, or demonstrates insufficient understanding.

Internship Documentation Requirements

To be Submitted After Completion of Internship

Mandatory Documents:

- Summer Internship Report File
- Signed One-Page Logbook
- Attendance Record
- Internship Completion Certificate
- Mentor Evaluation/Verification (if applicable)
- Presentation PPT
- Any supporting documents/certifications relevant to the internship

Plagiarism & Ethical Guidelines

- Internship reports must be original and based on actual work carried out during the internship.
- Proper acknowledgment and citations must be included wherever required.
- Submission of false or fabricated internship records/documents will lead to disciplinary action.
- Maximum allowable plagiarism should not exceed institutional norms (≤15%).
- Professional ethics, confidentiality, and organizational policies must be strictly followed.

Expected Learning Outcomes

By the end of the Summer Internship, students should be able to:

- Demonstrate practical understanding of industry practices and professional workflows.
- Apply technical and analytical skills in real-world scenarios.
- Communicate effectively in professional environments.
- Develop teamwork, adaptability, and problem-solving abilities.
- Present structured documentation of internship experiences and learning outcomes.
- Enhance employability, professional confidence, and industry readiness.

Text and Reference Books

Textbooks:

- Sommerville, I., Software Engineering, Pearson Education, 10th Edition.2015, ISBN- 978-0133943030
- Hunt, A. and Thomas, D., The Pragmatic Programmer, Addison-Wesley.2019, ISBN- 978-0135957059
- Wulf, W.A. and Barrett, G.C., Engineering Your Future, Oxford University Press.2015, ISBN-978-0199374557
- Nicholas, J.M. and Steyn, H., Project Management for Engineering, Business and Technology, Pearson.2017, ISBN- 978-1138048386.
- Bovee, C.L. and Thill, J.V., Business Communication Today, Pearson.2019, ISBN- 978-0134729398.

Recommended Reference Books:

- Orosz, G., The Software Engineer's Guidebook, 2023, ISBN: 978-9083381824.
- Richards, M. and Ford, N., Fundamentals of Software Architecture, O'Reilly, 2020, ISBN: 978-1492043454.
- Winters, T., Manshreck, T., and Wright, H., Software Engineering at Google, O'Reilly, 2020, ISBN: 978-1492082798.
- Kerzner, H., Project Management: A Systems Approach to Planning, Scheduling, and Controlling, Wiley, 2022, ISBN: 978-1119805373.
- Project Management Institute, PMBOK® Guide – Seventh Edition, 2021, ISBN: 978-1628256642.

Recommended Online Resources:

- GitHub Documentation — https://docs.github.com
- Microsoft Learn — https://learn.microsoft.com
- AWS Training and Certification — https://aws.amazon.com/training
- NASSCOM FutureSkills Prime — https://futureskillsprime.in
- LinkedIn Learning — https://www.linkedin.com/learning

Discipline Specific Electives-I						
1	**CSE001**	Next-Generation Programming Languages	**3**	**0**	**2**	**4**
2	**CSE002**	Cyber Security	**3**	**0**	**2**	**4**
3	**CSE003**	Deep Learning & Neural Networks	**3**	**0**	**2**	**4**
4	**CSE004**	iOS Application Development with Swift & SwiftUI	**3**	**0**	**2**	**4**

NEXT-GENERATION PROGRAMMING LANGUAGES

Program Name	B.Tech. CSE (Core)			
Course Name: Next-Generation Programming Languages	Course Code	L-T-P	Credits	Contact Hours
	CSE001	3-0-2	4	45
Type of Course:	Discipline Specific Elective (DSE)			
Semester:	V (Fifth Semester)			
Pre-requisite(s):	Basic programming knowledge, object-oriented programming and data structures fundamentals			

Course Description

This course introduces modern programming languages that represent the future of software development across systems programming, concurrent computing, functional programming and scalable web application development. The course focuses on modern programming paradigms including memory-safe systems programming, concurrency, functional programming and type-safe web development using contemporary languages such as Rust, Go, Scala and TypeScript.

Students will learn how modern programming languages solve current industry challenges such as safety, scalability, performance, concurrency and developer productivity. The course emphasizes hands-on implementation of real-world applications, concurrent backend services, functional data processing systems and scalable web applications using modern development tools and frameworks.

The laboratory component focuses on systems programming, concurrent applications, REST APIs, functional data pipelines and type-safe frontend applications aligned with modern software industry requirements.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply modern programming constructs using emerging languages.	L3 – Apply
CO2	Analyze modern programming paradigms and concurrency models.	L4 – Analyze
CO3	Develop scalable applications using modern programming tools.	L3 – Apply
CO4	Evaluate programming languages for different computing problems.	L5 – Evaluate

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	2	1	3	-	1	1	1	-	2
CO2	3	3	2	3	3	1	1	1	1	-	2
CO3	3	2	3	2	3	1	1	2	1	1	3
CO4	2	3	2	2	2	1	2	1	2	1	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts	PSO2 Applying Computing Techniques	PSO3 Analyzing Complex Problems	PSO4 Evaluating Solutions	PSO5 Developing Innovative Applications
CO1	3	3	2	1	2
CO2	3	3	3	2	2
CO3	3	3	3	2	3
CO4	2	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Safe Systems Programming using Rust	No. of Hours: 12

- Introduction to Rust & Modern Systems Programming
- Rust Development Environment & Cargo Package Manager
- Variables, Data Types & Control Structures
- Ownership, Borrowing & Lifetimes
- Stack vs Heap Memory Management
- Collections: vectors, strings and hash maps
- Error Handling: Result and Option types
- File Handling & Command-Line Applications
- Building Safe & Efficient System-Level Programs

Unit Number: 2	Concurrent & Backend Programming using Go	No. of Hours: 11

- Introduction to Go Programming Language
- Functions, Structs & Packages
- Concurrency using Goroutines
- Channels & Select Statement
- Synchronization & Concurrent Task Processing
- HTTP Servers & REST API Fundamentals
- JSON Handling & API Communication

- Backend Service Development & Routing
- Building Scalable Backend Applications

Unit Number: 3	Functional Programming using Scala	No. of Hours: 11

- Introduction to Functional Programming Concepts
- Immutability & Higher-Order Functions
- Lambda Expressions & Functional Collections
- Scala Data Types & Pattern Matching
- Option, Either & Functional Error Handling
- Collection Processing & Functional Pipelines
- Introduction to Data Processing Workflows
- Scala & Big Data Ecosystem Overview
- Building Functional Data Processing Applications

Unit Number: 4	Modern Web & Type-Safe Programming using TypeScript	No. of Hours: 11

- Introduction to TypeScript & Modern Web Development
- TypeScript vs JavaScript
- Types, Interfaces & Enums
- Generics & Type Inference
- DOM Manipulation & Event Handling
- API Integration & Asynchronous Programming
- Frontend Application Development Basics
- Type-Safe Web Application Design
- Building Scalable Web Applications using TypeScript

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Tools / Technologies Used	Mapped CO(s)	Bloom's Level
1	Rust Programming & CLI Application Development	Rust Compiler, Cargo, VS Code	CO1	L3 – Apply
2	Concurrent Backend Programming using Go	Go Compiler, VS Code, Postman	CO2	L4 – Analyze
3	Functional Data Processing using Scala	Scala, IntelliJ IDEA	CO3	L3 – Apply
4	Type-Safe Web Development	Node.js, TypeScript,	CO3	L3 – Apply

	using TypeScript	VS Code		
5	Capstone: Modern Multi-Paradigm Application Development	Rust, Go, Scala, TypeScript	CO1, CO2, CO3, CO4	L5 – Evaluate

Lab Assignments

Unit 1 Lab Assignment: Rust Programming & CLI Application Development	**CO1**
Tools / Technologies Used	Rust Compiler, Cargo, VS Code
Problem Statement	Develop safe and efficient system-level applications using Rust programming language and modern memory management concepts.
Sub-Problems	• Write Rust programs using variables and control structures • Implement ownership and borrowing examples • Demonstrate stack and heap memory operations • Implement vectors, strings and hash maps • Perform file handling and data processing operations • Build command-line utilities using Rust • Develop a mini system-level application such as a file search tool
Concepts Covered	Ownership, memory safety, collections and systems programming

Unit 2 Lab Assignment: Concurrent Backend Programming using Go	**CO2**
Tools / Technologies Used	Go Compiler, VS Code, Postman
Problem Statement	Develop concurrent backend applications using Go programming language and REST API development concepts.
Sub-Problems	• Write programs using functions, structs and packages • Implement concurrency using goroutines • Use channels and select statements for communication • Develop concurrent task processing systems • Build REST APIs and HTTP servers • Process JSON requests and API responses • Develop scalable backend service applications

Concepts Covered	Concurrency, goroutines, REST APIs and backend programming

Unit 3 Lab Assignment: Functional Data Processing using Scala	**CO3**
Tools / Technologies Used	Scala, IntelliJ IDEA
Problem Statement	Develop functional applications and data processing systems using Scala programming language and functional programming paradigms.
Sub-Problems	• Implement higher-order functions and lambda expressions • Process collections using functional operations • Implement pattern matching techniques • Apply functional error handling using Option and Either • Build functional data transformation pipelines • Process structured datasets using Scala collections • Develop mini functional data processing applications
Concepts Covered	Functional programming, pattern matching and data pipelines

Unit 4 Lab Assignment: Type-Safe Web Development using TypeScript	**CO3**
Tools / Technologies Used	Node.js, TypeScript, VS Code
Problem Statement	Build scalable and type-safe web applications using TypeScript and modern frontend development practices.
Sub-Problems	• Write TypeScript programs using types and interfaces • Implement generics and type inference examples • Perform DOM manipulation and event handling • Integrate APIs and asynchronous workflows • Build frontend application modules using TypeScript • Develop scalable type-safe web interfaces • Create interactive web applications with API integration
Concepts Covered	TypeScript, frontend development and type-safe programming

Capstone Lab Assignment: Modern Multi-Paradigm Application Development	CO1, CO2, CO3, CO4
Tools / Technologies Used	Rust, Go, Scala, TypeScript, VS Code
Problem Statement	Design and develop a modern multi-paradigm software application integrating systems programming, concurrency, functional programming and scalable web development concepts using next-generation programming languages.
Sub-Problems	• Design application architecture using appropriate programming paradigms • Develop system-level modules using Rust • Implement concurrent backend services using Go • Build functional data processing workflows using Scala • Develop frontend web interfaces using TypeScript • Integrate APIs and backend communication systems • Analyze scalability, performance and safety characteristics • Compare suitability of programming languages for different modules • Generate technical documentation and benchmarking reports • Present a fully functional modern application system
Concepts Covered	Systems programming, concurrency, functional programming and scalable web development

Text and Reference Books

Textbooks:

1. Programming Rust, Jim Blandy, Jason Orendorff & Leonora Tindall, 2nd Edition, O'Reilly Media, 2021. ISBN: 9781492052593
2. The Go Programming Language, Alan A. A. Donovan & Brian W. Kernighan, Addison-Wesley Professional, 2015. ISBN: 9780134190440
3. Programming in Scala, Martin Odersky, Lex Spoon & Bill Venners, 5th Edition, Artima Press, 2021. ISBN: 9780981531687
4. Effective TypeScript, Dan Vanderkam, O'Reilly Media, 2019. ISBN: 9781492053743

Recommended Reference Books:

1. Rust for Rustaceans, Jon Gjengset, No Starch Press, 2021. ISBN: 9781718501855
2. Concurrency in Go, Katherine Cox-Buday, O'Reilly Media, 2017. ISBN: 9781491941195

3. Scala for the Impatient, Cay S. Horstmann, 2nd Edition, Addison-Wesley Professional, 2016. ISBN: 9780134540568
4. Programming TypeScript, Boris Cherny, O'Reilly Media, 2019. ISBN: 9781492037651

Recommended Online Resources:

- Rust Official Documentation — https://www.rust-lang.org/learn
- Go Documentation — https://go.dev/doc/
- Scala Documentation — https://docs.scala-lang.org/
- TypeScript Documentation — https://www.typescriptlang.org/docs/
- MDN Web Docs — https://developer.mozilla.org/
- FreeCodeCamp Modern Programming Courses — https://www.freecodecamp.org/
- Coursera Programming Courses — https://www.coursera.org/browse/computer-science/software-development
- GitHub Open Source Projects — https://github.com/explore

CYBER SECURITY

Program Name	B.Tech. CSE (Core)			
Course Name: Cyber Security	Course Code	L-T-P	Credits	Contact Hours
	CSE002	3-0-2	4	45
Type of Course:	DSE			
Semester:	V (Fifth Semester)			
Pre-requisite(s):	Basic knowledge of computer networks, operating systems and programming fundamentals			

Course Description

This course focuses on practical cyber security concepts including network security, cryptography fundamentals, system security, authentication mechanisms, secure communication, web security, ethical hacking basics, malware awareness, and cyber defense strategies. Students gain exposure to modern security tools, vulnerability assessment techniques, and secure software and network practices widely used in industry and enterprise environments.

The course also emphasizes hands-on learning through security analysis exercises, network monitoring tasks, vulnerability assessment activities, secure coding practices, and mini security projects. Exposure to cyber laws, ethical issues, and security governance concepts helps students understand the broader impact of cyber security in organizational and societal contexts.

By the end of the course, students will be able to identify common cyber threats, apply basic security mechanisms, analyze system vulnerabilities, implement secure computing practices, and understand modern cyber defense methodologies for real-world applications.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply fundamental cyber security concepts to identify and mitigate security threats.	L3 – Apply
CO2	Analyze network and system vulnerabilities using security tools and techniques.	L4 – Analyze
CO3	Implement secure communication, authentication, and data protection mechanisms.	L3 – Apply
CO4	Evaluate modern cyber security practices, ethical issues, and cyber defense strategies.	L5 – Evaluate

CO–PO Articulation Matrix

COs \	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11

POs											
CO1	3	2	2	1	3	1	2	1	1	-	2
CO2	3	3	2	3	3	1	2	1	1	-	2
CO3	3	2	3	2	3	1	2	1	1	1	2
CO4	2	3	2	3	2	2	3	1	2	1	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Programming & Computing Fundamentals	PSO2 Software & System Development	PSO3 Problem Solving & Security Analysis	PSO4 Modern Computing Tools	PSO5 Innovative Technology Solutions
CO1	3	2	2	2	1
CO2	3	3	3	3	2
CO3	3	3	3	3	2
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Introduction to Cyber Security & Threat Landscape	No. of Hours: 12

- Introduction to cyber security, information security principles and CIA triad
- Cyber threats, attacks and vulnerabilities
- Types of malware: viruses, worms, trojans, ransomware and spyware
- Social engineering attacks, phishing and identity theft
- Cyber security awareness and safe computing practices
- Authentication, authorization and access control basics
- Introduction to ethical hacking and penetration testing concepts
- Cyber security tools and security monitoring basics

Unit Number: 2	Network Security & Cryptography Fundamentals	No. of Hours: 11

- Computer network security concepts and common network attacks
- Firewalls, intrusion detection systems and intrusion prevention systems
- Virtual Private Networks (VPNs) and secure communication concepts
- Basics of cryptography and encryption techniques
- Symmetric and asymmetric encryption methods
- Hashing, digital signatures and certificates

- Secure communication protocols: HTTPS, SSL/TLS basics
- Wireless and Wi-Fi security fundamentals

Unit Number: 3	System Security, Web Security & Secure Practices	No. of Hours: 11

- Operating system security and user privilege management
- Password security, multi-factor authentication and access management
- Web application security basics and common web attacks
- SQL injection, cross-site scripting and session hijacking concepts
- Secure coding practices and input validation techniques
- Vulnerability assessment and security scanning tools
- Backup, disaster recovery and incident response basics
- Cyber laws, digital privacy and ethical security practices

Unit Number: 4	Emerging Trends & Modern Cyber Defense Techniques	No. of Hours: 11

- Cloud security fundamentals and cloud-based threats
- Mobile and IoT security basics
- Introduction to cyber forensics and digital investigation
- Security monitoring and log analysis concepts
- Artificial intelligence and machine learning in cyber security
- Blockchain security fundamentals
- Security governance, policies and risk management
- Recent trends in cyber security and cyber defense strategies

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Security Awareness & Threat Analysis Toolkit	CO1	L3 – Apply
2	Network Security & Cryptography System	CO2	L4 – Analyze
3	Secure Web & System Protection Application	CO3	L3 – Apply
4	Cyber Defense & Security Monitoring System	CO4	L5 – Evaluate
5	Capstone: Integrated Cyber Security Management Platform	CO1, CO2, CO3, CO4	L5 – Evaluate

Lab Assignments

Unit 1 Lab Assignment: Security Awareness & Threat Analysis Toolkit	**CO1**
Problem Statement	Develop a basic cyber security analysis system to identify common threats, improve authentication security, and promote safe computing practices.
Sub-Problems	• Analyze phishing emails and malicious websites using security tools • Implement password strength checking and secure authentication mechanisms • Identify malware types and suspicious system activities • Generate basic security awareness reports and safe browsing guidelines
Tools Used	• Kali Linux • VirusTotal • Password Strength Checker • VirtualBox

Unit 2 Lab Assignment: Network Security & Cryptography System	**CO2**
Problem Statement	Build a secure communication and network analysis system using cryptography and network security concepts.
Sub-Problems	• Perform network scanning and packet analysis using security tools • Configure firewall and network filtering rules for traffic protection • Implement encryption, decryption and hashing techniques • Analyze secure communication protocols and wireless security mechanisms
Tools Used	• Nmap • Wireshark • OpenSSL • Windows/Linux Firewall

Unit 3 Lab Assignment: Secure Web & System Protection Application	**CO3**
Problem Statement	Develop a secure web and system protection framework to prevent common vulnerabilities and unauthorized access.
Sub-Problems	• Configure user privilege management and multi-factor authentication • Detect and analyze SQL Injection and Cross-Site Scripting attacks • Perform vulnerability assessment and web security testing • Implement secure coding, backup and recovery mechanisms
Tools Used	• DVWA • OWASP Juice Shop

	• OpenVAS • Burp Suite

Unit 4 Lab Assignment: Cyber Defense & Security Monitoring System	**CO4**
Problem Statement	Develop a cyber defense and security monitoring system for detecting, analyzing and responding to modern cyber threats.
Sub-Problems	• Monitor and analyze security logs and suspicious activities • Perform disk imaging and evidence acquisition using FTK Imager • Conduct basic digital forensic investigation using Autopsy • Generate security alerts, dashboards and incident response reports
Tools Used	• Splunk • ELK Stack • FTK Imager • Autopsy

Capstone Lab Assignment: Integrated Cyber Security Management Platform	**CO1, CO2, CO3, CO4**
Problem Statement	Design an integrated cyber security platform that combines authentication, monitoring, network protection, and secure communication techniques for organizational security management.
Sub-Problems	• Develop secure authentication and access modules • Implement encryption and secure communication systems • Analyze network and web vulnerabilities • Generate security monitoring dashboards • Perform incident response and recovery tasks • Apply secure coding and cyber defense techniques • Develop modular and scalable security utilities • Demonstrate integrated cyber security management practices

Text and Reference Books

Textbooks:

1. William Stallings, Cryptography and Network Security: Principles and Practice, 8th Edition, Pearson Education, 2020.
2. Charles P. Pfleeger and Shari Lawrence Pfleeger, Security in Computing, 5th Edition, Pearson Education, 2015.
3. Nina Godbole and Sunit Belapure, Cyber Security: Understanding Cyber Crimes, Computer Forensics and Legal Perspectives, Wiley India, 2021.

Recommended Reference Books:

1. Michael E. Whitman and Herbert J. Mattord, Principles of Information Security, 7th Edition, Cengage Learning, 2021.
2. Kevin Mitnick, The Art of Deception: Controlling the Human Element of Security, Wiley Publishing, 2011.
3. Jon Erickson, Hacking: The Art of Exploitation, 2nd Edition, No Starch Press, 2008.
4. Joseph Migga Kizza, Guide to Computer Network Security, 5th Edition, Springer, 2020.

Recommended Online Resources:

- **PortSwigger Web Security Academy**
- Best for web security, SQL Injection, XSS, Burp Suite practice, and hands-on labs.
- **TryHackMe Cyber Security Labs**
- Best for beginners and structured cyber security learning paths.
- **Hack The Box Training Platform**
- Best for advanced penetration testing and real-world hacking challenges.
- **LetsDefend SOC Training Platform**
- Best for SOC analyst training, SIEM, blue teaming, and incident response.
- **OWASP Official Documentation**
- Best reference for web application security standards and OWASP Top 10.
- **Cisco Networking Academy Cyber Security Courses**
- Strong foundation for networking and security basics.
- **Kali Linux Documentation**
- Essential for penetration testing tools and Linux security environment.
- **Wireshark Documentation**
- Best resource for packet analysis and network traffic monitoring.
- **Nmap Official Documentation**
- Excellent for network scanning and reconnaissance learning.

DEEP LEARNING & NEURAL NETWORKS

Program Name	B.Tech. CSE Core			
Course Name: Deep Learning & Neural Networks	Course Code	L-T-P	Credits	Contact Hours
	CSE003	3-0-2	04	45
Type of Course:	DSE			
Semester:	V (Fifth Semester)			
Pre-requisite(s):	Probability & Statistics, Linear Algebra, Python Programming, Essentials of Machine Learning			

Course Description

This course introduces the principles, mathematical foundations, architectures, and applications of Deep Learning and Neural Networks for intelligent computational systems. The course focuses on artificial neurons, multilayer perceptrons, activation functions, loss functions, backpropagation, optimization methods, convolutional neural networks, recurrent neural networks, autoencoders, generative models, and deep learning deployment practices.

Students will learn how machines learn complex representations from structured, image, sequential, and textual data using deep neural architectures. The course also introduces regularization, hyperparameter tuning, transfer learning, model evaluation, explainable AI, and responsible AI practices used in modern academic research and industry-oriented deep learning applications.

The laboratory component emphasizes practical implementation of neural network models, CNN-based image classification systems, RNN/LSTM-based sequence models, autoencoders, transfer learning workflows, and complete deep learning pipelines using modern frameworks. By the end of the course, students will be able to design, train, evaluate, optimize, and deploy deep learning systems for real-world problems.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Concise Course Outcome	Bloom's Level
CO1	Explain the fundamentals of artificial neural networks, deep learning architectures, activation functions, and learning mechanisms.	L2 – Understand
CO2	Analyze optimization methods, regularization techniques, CNNs, RNNs, LSTMs and deep learning model behavior.	L4 – Analyze
CO3	Develop deep learning applications using modern neural network frameworks and model training pipelines.	L3 – Apply
CO4	Evaluate deep learning systems for accuracy, generalization, scalability, explainability, ethics and performance.	L5 – Evaluate

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	–	2	–	–	1	1	–	2
CO2	3	3	2	2	3	1	–	1	2	1	2
CO3	3	3	3	2	3	1	1	2	2	1	2
CO4	3	3	3	3	3	1	1	2	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	2	1	–	1
CO2	3	3	3	2	2
CO3	3	3	2	2	3
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Foundations of Neural Networks and Deep Learning	No. of Hours: 12

- Introduction to Deep Learning: evolution, applications, learning paradigms and deep learning pipeline architecture
- Artificial Neuron and Perceptron: biological motivation, McCulloch-Pitts neuron, perceptron learning and limitations
- Multilayer Neural Networks: feedforward networks, hidden layers, representation learning and universal approximation
- Activation Functions: sigmoid, tanh, ReLU, Leaky ReLU, ELU, softmax and their role in gradient flow
- Loss Functions and Learning: regression loss, classification loss, cross-entropy and empirical risk minimization
- Backpropagation and Gradient Descent: chain rule, weight updates, learning rate and convergence behavior

Unit Number: 2	Training, Optimization and Regularization of Deep Networks	No. of Hours: 12

- Optimization Algorithms: batch gradient descent, stochastic gradient descent, momentum, RMSprop, Adam, AdamW and adaptive learning
- Weight Initialization and Normalization: Xavier initialization, He initialization, batch normalization and layer normalization
- Regularization Techniques: L1 and L2 regularization, dropout, early stopping, data augmentation and model generalization
- Hyperparameter Tuning: learning rate scheduling, batch size, epochs, model depth, model width and validation strategy
- Model Evaluation: accuracy, precision, recall, F1-score, loss curves, confusion matrix and error analysis
- Overfitting and Generalization: bias-variance trade-off, cross-validation, robustness and performance interpretation

Unit Number: 3	Convolutional and Sequence Deep Learning Architectures	No. of Hours: 11

- Convolutional Neural Networks: convolution operation, filters, feature maps, padding, stride and pooling operations
- CNN Architectures: LeNet, AlexNet, VGG, ResNet and modern image classification workflows
- Transfer Learning: pretrained models, feature extraction, fine-tuning and domain adaptation
- Recurrent Neural Networks: sequence modeling, temporal dependency learning and vanishing gradient problem
- LSTM and GRU Networks: gates, memory cells, sequence classification, time-series analysis and language applications
- Attention Mechanisms and Sequence Models: encoder-decoder learning, attention basics and transformer introduction

Unit Number: 4	Advanced Deep Learning Models, Deployment and Responsible AI	No. of Hours: 10

- Autoencoders: undercomplete, sparse, denoising and variational autoencoders for representation learning
- Generative Deep Learning: GAN fundamentals, generator-discriminator training and generative applications
- Deep Reinforcement Learning Basics: agents, environment, policy, reward and neural function approximation
- Deep Learning Deployment and Optimization: model saving, inference, compression, quantization and API-based serving
- Explainability and Responsible Deep Learning: bias, fairness, interpretability, privacy, security and trustworthy AI

- Emerging Trends: foundation models, self-supervised learning, multimodal learning, edge AI and neural architecture search
- Case Studies: healthcare diagnosis, speech recognition, recommendation systems, autonomous systems and intelligent vision platforms

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Neural Network Fundamentals and Training Toolkit	CO1	L2 – Understand
2	Optimization, Regularization and Model Evaluation Framework	CO2	L4 – Analyze
3	CNN and Transfer Learning-Based Vision Application	CO3	L3 – Apply
4	Sequence Modeling and Generative Deep Learning System	CO4	L4 – Analyze
5	Capstone: End-to-End Deep Learning Application Project	CO1, CO2, CO3, CO4	L5 – Evaluate

Lab Assignments

Unit 1 Lab Assignment: Neural Network Fundamentals and Training Toolkit	CO1

Problem Statement

Design a neural network toolkit that implements basic deep learning components and demonstrates feedforward learning for classification and regression problems.

Sub-Problems

- Implement artificial neuron and perceptron models
- Build a multilayer perceptron using forward propagation
- Apply different activation functions and compare their behavior
- Implement loss computation for regression and classification tasks
- Perform backpropagation-based weight updates
- Analyze training loss and validation performance
- Generate neural network training and learning reports

Concepts Covered: neural networks, activation functions, loss functions and backpropagation

Unit 2 Lab Assignment: Optimization, Regularization and Model Evaluation Framework	CO2

Problem Statement

Develop a deep learning training framework that compares optimization algorithms, regularization strategies and evaluation metrics.

Sub-Problems

- Implement SGD, momentum, RMSprop, Adam and AdamW optimizers
- Apply dropout, L1/L2 regularization and early stopping
- Perform batch normalization or layer normalization experiments
- Tune learning rate, batch size, epochs and network depth
- Evaluate models using accuracy, precision, recall and F1-score
- Plot training and validation loss curves
- Generate optimization and regularization analysis reports

Concepts Covered: optimization algorithms, regularization, hyperparameter tuning and model evaluation

Unit 3 Lab Assignment: CNN and Transfer Learning-Based Vision Application	**CO3**

Problem Statement

Create a convolutional deep learning application for image classification using CNN architectures and transfer learning.

Sub-Problems

- Implement convolution, pooling and fully connected layers
- Train a CNN model on an image dataset
- Apply data augmentation to improve generalization
- Use pretrained CNN models for transfer learning
- Fine-tune selected layers for domain-specific classification
- Analyze confusion matrix and class-wise performance
- Generate CNN architecture and evaluation reports

Concepts Covered: CNNs, image classification, transfer learning and feature extraction

Unit 4 Lab Assignment: Sequence Modeling and Generative Deep Learning System	**CO4**

Problem Statement

Develop a sequence learning and generative deep learning system for temporal data analysis and representation learning.

Sub-Problems

- Implement RNN, LSTM or GRU models for sequence classification
- Analyze vanishing gradient behavior and memory-based learning
- Build an autoencoder for feature compression or reconstruction
- Implement a basic generative model such as VAE or GAN workflow
- Compare sequence and generative model performance
- Apply model saving and inference optimization techniques
- Generate responsible AI and deployment evaluation reports

Concepts Covered: RNN, LSTM, GRU, autoencoders, generative models and deployment

Capstone Lab Assignment: End-to-End Deep Learning Application Project	CO1, CO2, CO3, CO4

Problem Statement

Design and implement a complete deep learning application that includes data preprocessing, model development, optimization, evaluation, deployment and documentation.

Sub-Problems

- Prepare and preprocess a real-world dataset
- Select a suitable deep learning architecture for the problem
- Train and optimize the model using appropriate optimizers
- Apply regularization and hyperparameter tuning strategies
- Evaluate model performance using suitable metrics
- Analyze model explainability, fairness and robustness
- Deploy or demonstrate inference using a simple interface or API
- Prepare dashboards, technical documentation and project report

Concepts Covered: deep learning pipeline, model optimization, deployment, explainability and responsible AI

Text and Reference Books

Textbooks:

1. Ian Goodfellow, Yoshua Bengio and Aaron Courville, “Deep Learning,” 1st Edition, MIT Press, 2016. ISBN: 978-0262035613
2. Charu C. Aggarwal, “Neural Networks and Deep Learning: A Textbook,” 1st Edition, Springer, 2018. ISBN: 978-3319944623

Recommended Reference Books:

1. Michael Nielsen, “Neural Networks and Deep Learning,” Determination Press, 2015. Online book
2. François Chollet, “Deep Learning with Python,” 2nd Edition, Manning Publications, 2021. ISBN: 978-1617296864
3. Aurélien Géron, “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow,” 3rd Edition, O’Reilly Media, 2022. ISBN: 978-1098125974
4. Aston Zhang, Zachary C. Lipton, Mu Li and Alexander J. Smola, “Dive into Deep Learning,” 1st Edition, Cambridge University Press, 2023. ISBN: 978-1009389433

Recommended Online Resources:

- TensorFlow Tutorials — https://www.tensorflow.org/tutorials
- PyTorch Tutorials — https://docs.pytorch.org/tutorials
- Keras Documentation — https://keras.io/guides
- DeepLearning.AI Resources — https://www.deeplearning.ai/resources
- NPTEL – Deep Learning — https://nptel.ac.in
- Dive into Deep Learning — https://d2l.ai

iOS Application Development with Swift & SwiftUI

Program Name	B.Tech. CSE Core			
Course Name: iOS Application Development with Swift & SwiftUI	Course Code	L-T-P	Credits	Contact Hours
	CSE004	3-0-2	04	45
Type of Course:	Discipline Specific Elective (DSE I)			
Semester:	V (Fifth Semester)			
Pre-requisite(s):	Basic programming knowledge, object-oriented programming and mobile application development fundamentals.			

Course Description

This course introduces students to modern iOS application development using Swift and SwiftUI for building interactive, scalable and user-centric mobile applications for Apple platforms. The course focuses on Swift programming fundamentals, declarative UI design using SwiftUI, state management, networking, data persistence, animations, cloud integration and deployment workflows.

Students will learn how to design responsive interfaces, manage application states, integrate APIs, work with device capabilities and develop cross-device applications for iPhone and iPad ecosystems. The course also introduces modern mobile architectures, concurrency, security, accessibility and application optimization techniques.

The laboratory component emphasizes hands-on mobile application development through practical assignments, UI design exercises, networking tasks and mini projects using Xcode, SwiftUI and Apple development frameworks. By the end of the course, students will be able to design responsive mobile interfaces, develop data-driven iOS applications, integrate cloud services, and deploy scalable applications on Apple platforms.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Concise Course Outcome	Bloom's Level
CO1	Apply Swift and SwiftUI for iOS application development.	L3 – Apply
CO2	Design responsive and interactive mobile interfaces.	L4 – Analyze
CO3	Develop data-driven and network-enabled iOS applications.	L3 – Apply
CO4	Analyze iOS applications for performance, security and scalability.	L4 – Analyze

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	–	2	–	–	1	1	–	2

CO2	3	3	2	2	3	1	–	1	2	1	2
CO3	3	3	3	2	3	1	1	2	2	1	2
CO4	3	3	3	3	3	1	1	2	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	2	1	–	1
CO2	3	3	3	2	2
CO3	3	3	2	2	3
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Swift Programming Fundamentals & iOS Development Environment	No. of Hours: 12

- Introduction to iOS Development: Apple ecosystem, Xcode IDE, Swift language overview and app lifecycle
- Swift Programming Basics: variables, constants, data types, operators, control statements and functions
- Collections & Data Structures: arrays, dictionaries, sets and tuples
- Object-Oriented Programming in Swift: classes, structures, inheritance, protocols and extensions
- Error Handling & Optionals: optional binding and safe programming practices
- Introduction to SwiftUI: declarative UI design principles and view hierarchy
- Practical Activities: development of basic Swift applications and SwiftUI components

Unit Number: 2	SwiftUI User Interfaces & State Management	No. of Hours: 12

- SwiftUI Components: text, buttons, images, lists, forms and navigation views
- Layout Management: stacks, grids, spacers and responsive UI design
- State Management: @State, @Binding, @ObservedObject and environment objects
- Navigation & Routing: navigation stacks, tab views and modal presentations
- Animations & Transitions: implicit and explicit animations in SwiftUI
- Dark Mode & Accessibility: adaptive design and accessibility guidelines
- Practical Activities: development of interactive and responsive mobile interfaces

Unit Number: 3	Data Persistence, Networking & Device Integration	No. of Hours: 11

- Data Persistence: UserDefaults, Core Data and file handling
- Networking in iOS: REST APIs, JSON parsing and asynchronous data fetching
- Concurrency in Swift: async-await, tasks and background processing
- Location & Device Features: GPS, camera, notifications and sensor integration
- Application Security: secure storage, authentication and privacy considerations
- MVVM Architecture: scalable application design and separation of concerns

Unit Number: 4	Advanced iOS Development & Application Deployment	No. of Hours: 10

- Advanced SwiftUI Concepts: reusable components, custom views and advanced modifiers
- Cloud Integration: Firebase and iCloud integration basics
- Testing & Debugging: unit testing, UI testing and debugging tools
- Performance Optimization: memory management and battery optimization techniques
- App Store Deployment: certificates, provisioning profiles and publishing process
- Cross-Platform Development Concepts: SwiftUI for iPadOS, watchOS and macOS
- Mini Project Development: complete iOS application design and deployment

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Swift Fundamentals & UI Development Toolkit	CO1	L3 – Apply
2	Interactive SwiftUI Application Framework	CO2	L4 – Analyze
3	Networked & Data-Driven iOS Application System	CO3	L3 – Apply
4	Advanced SwiftUI & Cloud Integration Platform	CO4	L4 – Analyze
5	Capstone: Smart iOS Productivity & Analytics Application	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Swift Fundamentals & UI Development Toolkit	CO1

Problem Statement

Design a Swift-based toolkit demonstrating programming fundamentals and user interface development using SwiftUI.

Sub-Problems

- Implement arithmetic and logical operations using Swift
- Develop menu-driven applications using control statements

- Create reusable classes and structures
- Design SwiftUI interfaces using views and controls
- Implement error handling and optional binding
- Analyze application execution and UI behavior
- Generate structured outputs and reports

Concepts Covered: Swift programming, SwiftUI basics, OOP and error handling

Unit 2 Lab Assignment: Interactive SwiftUI Application Framework	**CO2**

Problem Statement

Develop an interactive mobile application using SwiftUI with dynamic interfaces and state management.

Sub-Problems

- Design responsive interfaces using stacks and navigation views
- Implement dynamic state management using SwiftUI property wrappers
- Add animations and transitions for user interaction
- Create reusable SwiftUI components and forms
- Implement accessibility and adaptive design features
- Compare navigation and routing strategies
- Generate UI interaction and usability reports

Concepts Covered: SwiftUI, state management, animations and accessibility

Unit 3 Lab Assignment: Networked & Data-Driven iOS Application System	**CO3**

Problem Statement

Create a data-driven iOS application integrating APIs, local storage and device capabilities.

Sub-Problems

- Fetch and display remote data using REST APIs
- Parse JSON data and update SwiftUI views dynamically
- Implement Core Data for local persistence
- Use async-await for concurrent operations
- Integrate device features such as GPS and notifications
- Apply authentication and secure storage techniques
- Generate synchronization and analytics reports

Concepts Covered: Networking, persistence, concurrency and device integration

Unit 4 Lab Assignment: Advanced SwiftUI & Cloud Integration Platform	**CO4**

Problem Statement

Develop an advanced iOS application integrated with cloud services and optimized for deployment.

Sub-Problems

- Develop reusable and scalable SwiftUI architectures

- Integrate Firebase or iCloud services
- Implement unit testing and UI testing strategies
- Optimize application performance and memory usage
- Configure App Store deployment settings
- Analyze cloud synchronization and scalability
- Generate deployment and optimization reports

Concepts Covered: Cloud integration, testing, optimization and deployment

Capstone Lab Assignment: Smart iOS Productivity & Analytics Application	**CO1, CO2, CO3, CO4**

Problem Statement

Design and implement an intelligent iOS productivity application integrating SwiftUI, cloud services, analytics and device capabilities.

Sub-Problems

- Develop scalable and modular application components
- Implement secure authentication and cloud synchronization
- Integrate analytics dashboards and visualizations
- Use advanced animations and responsive UI design
- Analyze application performance and usability metrics
- Deploy and test applications across Apple devices
- Implement accessibility and privacy features
- Generate technical documentation and deployment reports

Concepts Covered: Complete iOS development, SwiftUI, analytics and cloud integration

Text and Reference Books

Textbooks:

1. Hacking with Swift, Paul Hudson, SwiftUI Edition, 5th Edition, Hacking with Swift Publications, 2023. ISBN: 9781916177420
2. Programming iOS, Matt Neuburg, 14th Edition, O'Reilly Media, 2023. ISBN: 9781098150617

Recommended Reference Books:

1. Thinking in SwiftUI, Chris Eidhof & Florian Kugler, 2nd Edition, objc.io, 2021. ISBN: 9783945490068
2. SwiftUI Apprentice, Hwee-Boon Yar, 3rd Edition, Kodeco Publications, 2023. ISBN: 9781950325566
3. Swift Concurrency by Example, Antoine van der Lee, 1st Edition, SwiftLee Publications, 2022. ISBN: 9789083207834
4. SwiftUI Cookbook, Vandad Nahavandipoor, 1st Edition, O'Reilly Media, 2021. ISBN: 9781838981860

Recommended Online Resources:

- Apple Swift Documentation — https://developer.apple.com/swift

- SwiftUI Tutorials by Apple — https://developer.apple.com/tutorials/swiftui
- Hacking with Swift — https://www.hackingwithswift.com/
- Kodeco iOS Tutorials — https://www.kodeco.com/ios
- Apple Human Interface Guidelines — https://developer.apple.com/design/human-interface-guidelines
- Stanford CS193p iOS Development Course — https://cs193p.sites.stanford.edu/

SEMESTER: VI

ESSENTIALS OF MACHINE LEARNING

Program Name	B.Tech. CSE Core			
Course Name: Essentials of Machine Learning	Course Code	L-T-P	Credits	Contact Hours
	CSE303	3-0-2	04	45
Type of Course:	Major			
Semester:	VI (Sixth Semester)			
Pre-requisite(s):	Probability & Statistics, Data Analysis using Python			

Course Description

This course introduces the foundational concepts, workflows and practical applications of Machine Learning required for solving real-world business and engineering problems. The course focuses on industry-oriented machine learning workflows including data preprocessing, feature engineering, supervised learning, unsupervised learning, model evaluation and intelligent decision-making systems.

The course is designed with a strong practical orientation and emphasizes implementation using modern open-source tools and frameworks widely adopted in the industry. Students will work on real-world datasets, predictive analytics problems, recommendation systems and intelligent automation use-cases relevant to domains such as healthcare, finance, e-commerce, cybersecurity and business analytics.

The laboratory component focuses on hands-on implementation using Python, Scikit-learn, Pandas, NumPy, XGBoost and visualization libraries. Students will gain practical exposure to model building, evaluation, optimization and deployment-ready machine learning workflows aligned with current industry standards.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Concise Course Outcome	Bloom's Level
CO1	Explain machine learning workflows and concepts.	L2 – Understand
CO2	Implement machine learning models using modern tools.	L3 – Apply
CO3	Analyze model performance and optimization techniques.	L4 – Analyze
CO4	Develop intelligent ML applications for real-world problems.	L3 – Apply

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	2	2	3	1	1	1	1	1	2
CO2	3	3	3	2	3	1	1	2	1	1	3
CO3	3	3	2	3	3	1	1	1	1	1	3
CO4	3	3	3	3	3	2	1	2	1	1	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	2	2	1	1
CO2	3	3	2	2	2
CO3	3	3	3	2	2
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Foundations of Machine Learning & Data Preparation	No. of Hours: 12

- Introduction to Machine Learning: AI vs ML, ML workflows, real-world applications and industry use-cases
- Types of Machine Learning: supervised learning, unsupervised learning and reinforcement learning basics
- ML Problem Formulation: classification, regression and clustering problems
- Data Collection & Dataset Understanding: structured datasets, feature-target identification and dataset splitting
- Data Preprocessing: handling missing values, categorical encoding and feature scaling
- Feature Engineering Basics: feature selection, transformation and dimensionality reduction intuition
- ML Workflow Design: training, validation and testing pipelines
- Practical Activities: preprocessing and preparing real-world datasets for ML workflows

Unit Number: 2	Supervised Machine Learning Techniques	No. of Hours: 12

- Regression Models: linear regression and regression-based prediction systems
- Classification Models: logistic regression, decision trees and Naïve Bayes classifiers
- Ensemble Learning Basics: random forest and boosting concepts
- K-Nearest Neighbors (KNN): distance-based learning workflows
- Support Vector Machines (SVM): classification boundaries and hyperplane intuition
- Model Training & Validation: train-test split, cross-validation and bias-variance concepts
- Business Applications of Supervised Learning: fraud detection, churn prediction and recommendation systems
- Practical Activities: predictive modeling using structured datasets and ML libraries

Unit Number: 3	Unsupervised Learning & Model Evaluation	No. of Hours: 11

- Clustering Fundamentals: segmentation and pattern discovery workflows
- K-Means Clustering: centroid-based clustering and customer segmentation
- Hierarchical Clustering: dendrograms and grouping workflows
- Association Rule Mining: support, confidence and recommendation patterns
- Model Evaluation Metrics: accuracy, precision, recall, F1-score and ROC-AUC
- Confusion Matrix & Error Analysis: classification performance interpretation
- Model Optimization Basics: hyperparameter tuning and feature importance analysis
- Practical Activities: clustering, segmentation and model evaluation workflows

Unit Number: 4	Applied Machine Learning Systems & Intelligent Applications	No. of Hours: 10

- Recommendation System Basics: collaborative filtering and content-based recommendation concepts
- Time-Series Forecasting Basics: trend analysis and forecasting intuition
- ML Pipelines & Automation: reusable workflows and ML lifecycle basics
- Explainable AI Basics: feature importance and interpretable ML concepts
- Ethics in Machine Learning: fairness, bias and responsible AI practices
- Industry Use-Cases: healthcare analytics, cybersecurity analytics, business intelligence and smart automation systems
- Introduction to Deployment Concepts: APIs, model serialization and deployment intuition
- Practical Activities: end-to-end machine learning case studies and intelligent applications

Lab Experiments – CO Mapping Table

S. No.	Experiment Title	Tools / Technologies Used	Mapped CO(s)	Bloom's Level
1	Data Preprocessing & Feature Engineering Toolkit	Python, Pandas, NumPy	CO1	L2 – Understand

2	Supervised Learning & Predictive Analytics System	Scikit-learn, XGBoost, Jupyter Notebook	CO2	L3 – Apply
3	Clustering & Model Evaluation Platform	Python, Scikit-learn, Seaborn	CO3	L4 – Analyze
4	Intelligent ML Applications & Recommendation System	Python, Scikit-learn, Plotly	CO4	L3 – Apply
5	Capstone: End-to-End Machine Learning Solution	Python, Scikit-learn, XGBoost, Pandas	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Data Preprocessing & Feature Engineering Toolkit	**CO1**

Tools / Technologies Used: Python, Pandas, NumPy, Jupyter Notebook

Problem Statement

Develop preprocessing and feature engineering workflows for preparing structured datasets for machine learning applications.

Sub-Problems

- Import and clean structured datasets
- Handle missing values and inconsistent records
- Perform categorical encoding and feature scaling
- Split datasets into training and testing sets
- Apply feature transformation and normalization techniques
- Visualize feature distributions and correlations
- Generate ML-ready datasets and preprocessing reports

Concepts Covered: Data preprocessing, feature engineering and ML workflow preparation

Unit 2 Lab Assignment: Supervised Learning & Predictive Analytics System	**CO2**

Tools / Technologies Used: Scikit-learn, XGBoost, Python

Problem Statement

Develop supervised machine learning models for solving classification and regression problems using real-world datasets.

Sub-Problems

- Train regression and classification models
- Implement decision trees and random forests
- Apply KNN and SVM algorithms for prediction tasks
- Perform model validation and cross-validation
- Analyze prediction accuracy and performance metrics

- Compare multiple supervised learning models
- Generate predictive analytics dashboards and reports

Concepts Covered: Supervised learning and predictive analytics

Unit 3 Lab Assignment: Clustering & Model Evaluation Platform	**CO3**

Tools / Technologies Used: Python, Scikit-learn, Matplotlib, Seaborn

Problem Statement

Develop unsupervised learning systems for segmentation and model evaluation using clustering and analytics techniques.

Sub-Problems

- Implement K-Means and hierarchical clustering algorithms
- Perform customer segmentation workflows
- Visualize cluster distributions and grouping patterns
- Compute model evaluation metrics and confusion matrices
- Analyze classification and clustering performance
- Perform feature importance and tuning analysis
- Generate evaluation reports and visual summaries

Concepts Covered: Unsupervised learning, clustering and model evaluation

Unit 4 Lab Assignment: Intelligent ML Applications & Recommendation System	**CO4**

Tools / Technologies Used: Python, Scikit-learn, Plotly, Pandas

Problem Statement

Develop intelligent machine learning applications and recommendation systems for solving practical business problems.

Sub-Problems

- Build recommendation system workflows
- Perform trend analysis and forecasting basics
- Develop explainable AI visualizations and reports
- Create ML pipelines for intelligent automation systems
- Analyze business-oriented ML use-cases
- Visualize predictions and intelligent recommendations
- Generate intelligent analytics dashboards and summaries

Concepts Covered: Recommendation systems, explainable AI and intelligent applications

Capstone Lab Assignment: End-to-End Machine Learning Solution	**CO1, CO2, CO3, CO4**

Tools / Technologies Used: Python, Scikit-learn, XGBoost, Pandas, Plotly

Problem Statement

Design and implement a complete machine learning solution for solving a real-world business or engineering problem using predictive analytics and intelligent workflows.

Sub-Problems

- Collect and preprocess real-world datasets
- Perform feature engineering and transformation workflows
- Train supervised and unsupervised ML models
- Evaluate models using performance metrics and validation techniques
- Build recommendation or predictive analytics workflows
- Generate intelligent dashboards and prediction visualizations
- Interpret model outputs and explainability results
- Present business recommendations and intelligent insights

Concepts Covered: Machine learning workflows, predictive analytics and intelligent applications

Text and Reference Books

Textbooks:

1. Aurélien Géron, "Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow," 3rd Edition, O'Reilly Media, 2022.
2. Andreas C. Müller & Sarah Guido, "Introduction to Machine Learning with Python," O'Reilly Media, 2016.
3. Sebastian Raschka & Vahid Mirjalili, "Python Machine Learning," 3rd Edition, Packt Publishing, 2019.

Recommended Reference Books:

1. Oliver Theobald, "Machine Learning for Absolute Beginners," 2nd Edition, Scatterplot Press, 2020.
2. Alice Zheng & Amanda Casari, "Feature Engineering for Machine Learning," O'Reilly Media, 2018.
3. Max Kuhn & Kjell Johnson, "Applied Predictive Modeling," Springer, 2013.
4. Christoph Molnar, "Interpretable Machine Learning," Lulu.com, 2022.

Recommended Online Resources:

- Scikit-learn Documentation — https://scikit-learn.org/stable/
- Kaggle Learn Machine Learning — https://www.kaggle.com/learn
- Google Machine Learning Crash Course — https://developers.google.com/machine-learning/crash-course
- TensorFlow Tutorials — https://www.tensorflow.org/tutorials
- IBM Cognitive Class ML Courses — https://cognitiveclass.ai/

CLOUD COMPUTING & DEVOPS

Program Name	B.Tech. CSE (Core)			
Course Name: Cloud Computing & DevOps	Course Code	L-T-P	Credits	Contact Hours
	CSE304	3-0-2	04	45
Type of Course:	Major			
Semester:	VI (Sixth Semester)			
Pre-requisite(s):	Operating Systems, Computer Networks, Software Engineering and basic Linux knowledge			

Course Description

This course introduces modern Cloud Computing and DevOps practices used in today's software industry for scalable application deployment, automation and infrastructure management. The course focuses on practical cloud services, virtualization, containerization, CI/CD workflows, cloud deployment and DevOps automation using industry-standard tools and platforms.

Students will learn how modern software systems are developed, deployed and managed using cloud-native technologies and DevOps practices. The course emphasizes hands-on implementation of cloud services, containerized applications, deployment pipelines, infrastructure automation and collaborative software delivery workflows aligned with current industry requirements.

The laboratory component focuses on practical implementation using Linux, Docker, GitHub Actions, Jenkins, Kubernetes, AWS/Azure cloud services and Terraform basics. Students will gain practical exposure to cloud deployment, DevOps pipelines, container orchestration and scalable application management.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply cloud computing services and deployment workflows.	L3 – Apply
CO2	Develop containerized and automated DevOps pipelines.	L3 – Apply
CO3	Analyze scalable cloud infrastructure and monitoring systems.	L4 – Analyze
CO4	Build deployable cloud-native applications.	L3 – Apply

CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	2	1	3	1	1	–	1	1	2
CO2	2	3	3	2	3	–	–	1	2	1	2
CO3	3	3	2	3	3	1	1	–	2	1	3
CO4	2	2	3	2	3	1	1	2	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts	PSO2 Applying Computing Techniques	PSO3 Analyzing Complex Problems	PSO4 Evaluating Solutions	PSO5 Developing Innovative Applications
CO1	3	3	2	1	2
CO2	3	3	3	2	3
CO3	3	3	3	3	3
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Foundations of Cloud Computing & Virtualization	No. of Hours: 12

- Introduction to Cloud Computing: evolution of cloud computing, benefits of cloud platforms, characteristics of cloud systems and industry applications
- Cloud Service Models: IaaS, PaaS and SaaS models, service provider examples and practical business use cases
- Cloud Deployment Models: public cloud, private cloud, hybrid cloud and multi-cloud environments
- Virtualization Concepts: hypervisors, virtual machines, resource virtualization and virtual machine lifecycle
- Linux Fundamentals for Cloud Systems: Linux file system, shell commands, user management, file permissions and process monitoring
- Cloud Compute Services: virtual servers, compute instances, cloud resource provisioning and auto-scaling basics
- Cloud Storage Services: object storage, block storage, cloud backup and storage management basics
- Introduction to AWS/Azure/GCP: cloud dashboards, compute services, storage services and cloud account setup basics

- Cloud Networking Basics: IP addressing, security groups, firewall configuration and virtual private cloud concepts
- Cloud Resource Management: cloud instance creation, monitoring, billing basics and resource optimization

Unit Number: 2	DevOps Fundamentals & CI/CD Workflows	No. of Hours: 11

- Introduction to DevOps: DevOps lifecycle, Agile integration, collaboration workflows and software delivery pipelines
- Version Control using Git: repositories, commits, branching, merging and conflict resolution basics
- GitHub Collaboration Workflows: pull requests, repository management, issue tracking and team collaboration
- Continuous Integration (CI): automated build systems, testing workflows and integration pipelines
- Continuous Deployment (CD): deployment automation, release workflows and rollback concepts
- Jenkins Fundamentals: Jenkins setup, jobs, pipelines, plugins and build automation workflows
- GitHub Actions Automation: workflow files, runners, automation triggers and deployment workflows
- Build & Deployment Pipelines: build stages, testing stages, deployment stages and deployment monitoring
- CI/CD Monitoring & Debugging: workflow logs, deployment troubleshooting, failure handling and debugging workflows
- DevOps Best Practices: collaboration practices, deployment reliability, automation strategies and version management
- Practical Activities: GitHub repository workflows, Jenkins pipelines, CI/CD automation and deployment workflows

Unit Number: 3	Containerization & Cloud-Native Deployment	No. of Hours: 11

- Introduction to Docker: containerization concepts, Docker architecture, container lifecycle and Docker installation
- Docker Images & Containers: image creation, container execution, image repositories and Docker Hub usage
- Dockerfile Concepts: creating Dockerfiles, environment variables, port mapping and application packaging
- Multi-Container Applications: Docker Compose configuration, service communication and multi-container deployment
- Container Storage & Networking: persistent volumes, container ports, bridge networking and communication workflows
- Introduction to Kubernetes: container orchestration concepts, Kubernetes architecture and cluster management basics
- Kubernetes Components: pods, nodes, deployments, namespaces and service configuration
- Kubernetes Services & Scaling: load balancing, replica scaling and deployment scaling basics
- Cloud-Native Applications: microservices basics, stateless applications and cloud-native deployment workflows

- Container Monitoring Basics: monitoring container resources, application health checks and logging basics

Unit Number: 4	Infrastructure Automation, Monitoring & Cloud Applications	No. of Hours: 11

- Infrastructure as Code (IaC): infrastructure automation concepts, declarative configuration and provisioning workflows
- Terraform Fundamentals: providers, configuration files, infrastructure deployment and Terraform workflow basics
- Cloud Monitoring Concepts: infrastructure monitoring, application logging, alert management and performance tracking
- Prometheus Monitoring Basics: metrics collection, monitoring targets and alert configuration basics
- Grafana Dashboards: visualization of infrastructure metrics, monitoring dashboards and analytics panels
- Cloud Security Basics: authentication, authorization, Identity & Access Management (IAM) and access control
- Secure Cloud Deployment Practices: secrets management, deployment security, secure APIs and configuration management
- Cloud Reliability Concepts: backup systems, disaster recovery, fault tolerance and high availability basics
- Cloud Cost Optimization: resource optimization, monitoring cloud usage costs and scaling efficiency

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Tools / Technologies Used	Mapped CO(s)	Bloom's Level
1	Cloud Services & Virtualization Toolkit	AWS/Azure, Linux, VirtualBox	CO1	L3 – Apply
2	DevOps CI/CD Automation Platform	GitHub, Jenkins, GitHub Actions	CO2	L3 – Apply
3	Containerized Cloud-Native Deployment System	Docker, Kubernetes, Docker Compose	CO3	L4 – Analyze
4	Cloud Monitoring & Infrastructure Automation Platform	Terraform, Grafana, Prometheus	CO4	L3 – Apply
5	Capstone: Cloud-Native DevOps Application Deployment	Docker, Jenkins, Kubernetes	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Cloud Services & Virtualization Toolkit	CO1
Tools / Technologies Used	AWS/Azure Free Tier, Linux, VirtualBox

Problem Statement	Develop basic cloud computing and virtualization workflows for scalable application environments.
Sub-Problems	• Create and configure virtual machines • Perform Linux-based cloud administration tasks • Deploy cloud storage and compute resources • Explore AWS/Azure cloud service dashboards • Configure user access and cloud resources • Analyze cloud deployment performance basics • Generate cloud service deployment reports
Concepts Covered	Cloud computing, virtualization and cloud services

Unit 2 Lab Assignment: DevOps CI/CD Automation Platform	**CO2**
Tools / Technologies Used	GitHub, Jenkins, GitHub Actions
Problem Statement	Develop automated DevOps workflows using CI/CD pipelines and collaborative development practices.
Sub-Problems	• Configure Git and GitHub repositories • Implement branching and merging workflows • Create CI/CD pipelines using Jenkins • Automate workflows using GitHub Actions • Build and deploy sample applications automatically • Monitor CI/CD execution logs and workflows • Generate DevOps automation reports
Concepts Covered	DevOps, CI/CD and workflow automation

Unit 3 Lab Assignment: Containerized Cloud-Native Deployment System	**CO3**
Tools / Technologies Used	Docker, Docker Compose, Kubernetes
Problem Statement	Develop containerized and scalable cloud-native applications using modern deployment workflows.
Sub-Problems	• Create Docker images and containers • Write Dockerfiles for applications • Configure multi-container applications using Docker Compose • Deploy applications using Kubernetes basics • Scale applications using deployment configurations • Analyze container performance and resource usage • Generate cloud-native deployment reports
Concepts Covered	Containerization, orchestration and cloud-native

	deployment

Unit 4 Lab Assignment: Cloud Monitoring & Infrastructure Automation Platform	**CO4**
Tools / Technologies Used	Terraform, Prometheus, Grafana
Problem Statement	Develop automated infrastructure and monitoring systems for cloud-based applications.
Sub-Problems	• Configure infrastructure using Terraform basics • Monitor cloud services using Prometheus • Visualize metrics using Grafana dashboards • Analyze reliability and scalability concepts • Configure basic cloud security workflows • Monitor application performance and logs • Generate infrastructure automation and monitoring reports
Concepts Covered	Infrastructure automation, monitoring and reliability

Capstone Lab Assignment: Cloud-Native DevOps Application Deployment	**CO1, CO2, CO3, CO4**
Tools / Technologies Used	Docker, Jenkins, Kubernetes, GitHub Actions
Problem Statement	Design and implement a complete cloud-native DevOps workflow integrating CI/CD automation, containerized deployment and monitoring systems.
Sub-Problems	• Develop a deployable web application • Create CI/CD pipelines for automated deployment • Containerize applications using Docker • Deploy applications using Kubernetes basics • Monitor deployment performance and logs • Configure scalable deployment workflows • Generate deployment documentation and reports • Demonstrate end-to-end DevOps workflows
Concepts Covered	Cloud-native deployment, DevOps automation and scalable systems

Text and Reference Books

Textbooks:

1. Cloud Computing, McGraw Hill Education, 2019. ISBN: 9789353168155
2. Docker Deep Dive, Nigel Poulton, 2023 Edition, Independently Published, 2023. ISBN: 9781521822807

3. The DevOps Handbook, IT Revolution Press, 2nd Edition, 2021. ISBN: 9781950508402

Recommended Reference Books:

1. Terraform: Up and Running, O'Reilly Media, 3rd Edition, 2022. ISBN: 9781098116743
2. Learn Kubernetes in a Month of Lunches, Manning Publications, 2022. ISBN: 9781617297984
3. AWS for Solutions Architects, Packt Publishing, 2020. ISBN: 9781789534467
4. Jenkins: The Definitive Guide, O'Reilly Media, 2011. ISBN: 9781449305352

Recommended Online Resources:

- Docker Documentation — https://docs.docker.com/
- Kubernetes Documentation — https://kubernetes.io/docs/
- Jenkins Documentation — https://www.jenkins.io/doc/
- GitHub Actions Documentation — https://docs.github.com/en/actions
- Terraform Documentation — https://developer.hashicorp.com/terraform/docs
- AWS Training & Documentation — https://docs.aws.amazon.com/
- Azure Documentation — https://learn.microsoft.com/en-us/azure/
- Prometheus Documentation — https://prometheus.io/docs/

PROFESSIONAL PROBLEM SOLVING & CAREER READINESS – I

Program Name	B.Tech. CSE (Core)			
Course Name:	**Course Code**	**L-T-P**	**Credits**	**Contact Hours**
Course Name: Professional Problem Solving & Career Readiness -I	**CSE013**	**2-0-4**	**04**	**30**
Type of Course:	Major			
Semester:	VI (Sixth Semester)			
Pre-requisite(s):	Operating Systems, Computer Networks, Software Engineering and basic Linux knowledge			

1. Course Perspective

The course "Professional Problem Solving & Career Readiness – I" is designed to prepare students for modern technical recruitment processes through rigorous practice in problem-solving, object-oriented programming, core computer science fundamentals, coding interviews and professional communication.

The course follows a highly practical and industry-focused approach where students continuously solve coding problems, work on mini-projects, revise interview-oriented technical concepts and participate in mock technical and HR interviews. The course integrates aptitude building, debugging skills, DSA-based problem solving, resume preparation and placement readiness activities into a unified professional development framework.

The course emphasizes:

- Industry-oriented coding and debugging practice
- Interview-focused revision of OOPs, OS and DBMS
- DSA-based problem-solving patterns
- Technical communication and mock interview preparation
- Continuous assessment and performance monitoring
- Placement-oriented skill enhancement

The course is designed to improve employability readiness and technical confidence for campus placements, internships and professional software engineering roles.

2. Course Objectives

The course aims to:

- Develop professional problem-solving and coding skills required for technical recruitment processes.
- Strengthen students' understanding of core computer science concepts from placement and interview perspectives.

- Enhance debugging, logical reasoning and algorithmic thinking abilities through practical coding exercises.
- Build confidence in technical communication, resume presentation and interview performance.
- Prepare students for industry-level coding assessments and placement workflows.

3. Course Outcomes (COs)

CO	Course Outcome	Bloom's Level
CO1	Apply object-oriented programming concepts for software problem solving.	L3 – Apply
CO2	Solve coding and DSA-based interview problems.	L3 – Apply
CO3	Analyze OS and DBMS concepts for technical interviews.	L4 – Analyze
CO4	Demonstrate professional readiness in coding and interview environments.	L3 – Apply

4.CO–PO Articulation Matrix

Cos \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	2	1	2	–	–	1	1	1	1
CO2	3	3	2	2	2	–	–	1	1	1	1
CO3	3	2	1	2	1	–	–	–	–	1	1
CO4	2	2	1	–	1	1	1	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

5. Course Structure & Execution Model

Component	Focus Area	Hours	Marks
Module 1	OOPs & Software Development Fundamentals	10 Hours	30
Module 2	Coding & Problem Solving	20 Hours	50
Module 3	Operating Systems & DBMS for Interviews	20 Hours	50
Module 4	Mock Interviews & Career Readiness	10 Hours	40
Module 5	Final Placement Readiness Assessment	—	30
Total		**60 Hours**	**200**

6. Unit-wise Structure

Unit 1: Object-Oriented Programming & Software Development Fundamentals

Topics Covered

- OOP Paradigm & Object-Oriented Design
- Classes, Objects & Constructors
- Encapsulation & Data Hiding
- Inheritance & Polymorphism
- Constructor Chaining & Method Overriding
- Composition vs Aggregation
- Static Members & Memory Management
- Console-Based Java Mini Projects
- Code Reusability & Modular Programming

Mini Projects

Project 1	**Employee Payroll Management System**
Description	Develop a console-based payroll management system using Java OOP concepts including inheritance, abstraction, polymorphism and encapsulation.
Project 2	**Car Rental Management System**
Description	Develop a console-based rental management system for managing cars, customers and rental operations using collection-based data handling and OOP principles.

Unit 2: Coding, DSA & Problem Solving Patterns

Topics Covered

- Mathematical & Number-Based Problems
- Recursion & Pattern-Based Problems
- Array Traversal & Prefix Sum Problems
- String Manipulation & Hashing
- Searching & Sorting Techniques
- Sliding Window & Two Pointer Techniques
- Stack & Queue Problem Solving
- Binary Search on Answer
- BFS & DFS Basics
- Backtracking & Subset Problems
- Debugging & Optimization Techniques

Coding Platforms

- HackerRank
- LeetCode
- CodeChef
- Internal Coding LMS

Practical Focus

- Daily coding practice

- Timed coding assessments
- Debugging exercises
- Interview-style coding rounds

Unit 3: Technical Fundamentals for Interviews (OS & DBMS)

Operating Systems

- Process & Thread Management
- CPU Scheduling Algorithms
- Process Synchronization
- Deadlocks & Banker's Algorithm
- Memory Management & Paging
- Virtual Memory & Page Replacement
- File Systems & Disk Scheduling
- I/O Systems & Interrupts

Database Management Systems

- ER Model & Relational Model
- SQL Queries & Joins
- Aggregate Functions & Subqueries
- Functional Dependencies & Normalization
- Transactions & ACID Properties
- Concurrency Control & Locks
- Recovery Techniques

Practical Focus

- SQL query writing practice
- OS interview question discussions
- Scenario-based technical quizzes
- Interview-oriented MCQs and debugging exercises

Unit 4: Career Readiness & Mock Interview Preparation

Topics Covered

- Resume Building & LinkedIn Optimization
- Technical Mock Interviews
- HR & Behavioral Interview Preparation
- Self-Introduction & Professional Communication
- Group Discussion & Extempore Practice
- Email Writing & Corporate Etiquette
- Personality Grooming & Professional Conduct
- Placement Assessment Preparation

Practical Activities

- Technical Interview Simulations
- HR Mock Interviews
- Resume Evaluation Sessions
- Communication Practice Sessions
- Industry Assessment Simulations

7. Lab Activities & Practical Assignments

Lab No.	Experiment / Activity	Tools / Platforms Used	CO Mapping
1	OOP Mini Project Development	Java, IntelliJ IDEA / Eclipse	CO1
2	Coding Problem Solving Practice	HackerRank, LeetCode	CO2
3	DSA Pattern-Based Coding	Java / Python / C++	CO2
4	OS & DBMS Technical Quiz Practice	Google Forms, LMS	CO3
5	SQL Query & Normalization Exercises	MySQL / PostgreSQL	CO3
6	Resume & LinkedIn Preparation Workshop	Canva, LinkedIn	CO4
7	Mock Technical Interview	Interview Panels	CO4
8	HR & Communication Assessment	GD / Mock HR	CO4

8. Assessment & Evaluation Scheme

Component	Weightage
Coding Assessments	50 Marks
Technical Assessments (OS / DBMS / OOPs)	50 Marks
Mock Interviews & Communication	40 Marks
Mini Projects & Lab Activities	30 Marks
Final Placement Readiness Assessment	30 Marks
Total	**200 Marks**

9. Evaluation Guidelines

- Continuous internal evaluation shall be followed.
- Students must actively participate in coding labs and mock interviews.
- Coding assessments shall be conducted on coding platforms or LMS systems.
- Mock interviews shall be rubric-based and evaluated by faculty/trainers.
- Minimum 60% overall marks are required for successful completion of the course.
- Performance dashboards may be maintained for tracking coding progress, interview readiness and technical assessments.

10. Recommended Tools & Platforms

- HackerRank
- LeetCode
- CodeChef
- IntelliJ IDEA / Eclipse
- MySQL / PostgreSQL
- Google Forms
- Canva
- LinkedIn
- Internal LMS

11. Suggested Text Books

- Cracking the Coding Interview — Gayle Laakmann McDowell, 6th Edition, CareerCup, 2015.
- Head First Object-Oriented Analysis and Design — O'Reilly Media, 2006.
- Operating System Concepts — 10th Edition, Wiley, 2018.
- Database System Concepts — 7th Edition, McGraw-Hill, 2019.

12. Recommended Online Resources

- HackerRank — https://www.hackerrank.com/
- LeetCode — https://leetcode.com/
- CodeChef — https://www.codechef.com/
- GeeksforGeeks Interview Preparation — https://www.geeksforgeeks.org/interview-preparation-for-software-developer/
- PrepInsta Placement Preparation — https://prepinsta.com/
- IndiaBix Aptitude Preparation — https://www.indiabix.com/

ARITHMETIC & REASONING SKILLS

Program Name	B.Tech. CSE (Core)			
Course Name:	**Course Code**	**L-T-P**	**Credits**	**Contact Hours**
Course Name: Arithmetic & Reasoning Skills		**3-0-0**	**03**	**45**
Type of Course:	AEC			
Semester:	VI (Sixth Semester)			
Pre-requisite(s):	Basic school-level mathematics and logical reasoning skills			

Course Description

This course is designed to strengthen students' arithmetic aptitude, logical reasoning ability, analytical thinking and quantitative problem-solving skills required for campus placements, competitive examinations and professional assessments. The course focuses on improving speed, accuracy and logical decision-making using placement-oriented aptitude and reasoning techniques.

Students will learn shortcut approaches, quantitative reasoning methods, analytical thinking strategies and aptitude-solving frameworks commonly used in placement tests conducted by service-based and product-based companies. The course emphasizes practical problem-solving, test-taking strategies and timed assessment practice rather than theoretical mathematics.

The course also develops critical thinking, numerical fluency and reasoning efficiency necessary for aptitude rounds, online assessments and interview processes.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Understand arithmetic algorithms required for solving mathematical problems.	L2 – Understand
CO2	Apply arithmetic algorithms to improve proficiency in calculations.	L3 – Apply
CO3	Analyze cases, scenarios and variables in reasoning problems.	L4 – Analyze
CO4	Evaluate suitable approaches for solving aptitude and reasoning problems.	L5 – Evaluate

CO–PO Articulation Matrix

Cos \	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11

POs											
CO1	3	2	2	1	2	–	–	1	1	1	1
CO2	3	3	2	2	2	–	–	1	1	1	1
CO3	3	2	1	2	1	–	–	–	–	1	1
CO4	2	2	1	–	1	1	1	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

COs \ PSOs	PSO1	PSO2	PSO3	PSO4	PSO5
CO1	3	2	1	-	-
CO2	2	3	2	1	-
CO3	2	2	3	2	1
CO4	1	2	2	3	1

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Mathematical Essentials	No. of Hours: 12

- Vedic Mathematics: rapid calculations, multiplication shortcuts, divisibility shortcuts and approximation techniques
- Classification of Numbers: natural numbers, whole numbers, integers, rational numbers and irrational numbers
- Divisibility Rules: divisibility tests, factors, multiples, HCF and LCM concepts
- Percentage: percentage calculations, percentage increase/decrease, profit-loss applications and discount-based problems
- Ratio & Proportion: direct proportion, inverse proportion, ratio comparison and mixture-based applications
- Practical Focus: speed calculations, shortcut methods and placement-oriented arithmetic practice

Unit Number: 2	Fundamentals of Logical Reasoning	No. of Hours: 11

- Blood Relations: family-tree analysis, coded relations and generation-based reasoning
- Direction Sense: directional movement, distance calculation and orientation-based problems
- Coding-Decoding: alphabet coding, number coding, pattern-based coding and logical decoding techniques
- Logical Analysis: reasoning-based interpretation and analytical deduction basics
- Practical Focus: puzzle-solving speed, reasoning assessments and interview-oriented logical reasoning practice

Unit Number: 3	Elementary Quantitative Skills	No. of Hours: 11

- Simple & Compound Interest: interest calculations, compound growth and EMI-based basics
- Average: weighted average, group average and age-based average problems
- Partnership: investment sharing, profit distribution and ratio-based partnership problems
- Time & Work: work efficiency, pipes & cisterns and collaborative work problems
- Time, Speed & Distance: relative speed, trains, boats & streams and circular track problems
- Quantitative Problem Solving: arithmetic optimization, approximation methods and timed aptitude practice
- Practical Focus: company-level aptitude questions and speed improvement exercises

Unit Number: 4	Advanced Quantitative Skills	No. of Hours: 11

- Permutation & Combination: arrangements, selections, counting principles and grouping problems
- Probability: probability basics, conditional probability and event-based reasoning problems
- Analytical Quantitative Problems: mixed aptitude sets and logical quantitative analysis
- Placement Assessment Strategies: question selection, time management and aptitude-solving approaches
- Practical Focus: mock aptitude assessments, placement test simulations and problem-solving strategies

Suggested Assessment Strategy

Course Outcomes	Quiz	Test	Presentation	Assignments & Worksheets	Participation & Attendance	Mid-Term Exam	End-Term Exam
CO1	✓	✓		✓	✓	✓	✓
CO2	✓	✓	✓	✓	✓	✓	✓
CO3	✓	✓		✓	✓	✓	✓
CO4	✓	✓	✓	✓	✓	✓	✓

Suggested Learning Activities

- Worksheet-based aptitude practice
- Timed arithmetic assessments
- Group-based reasoning discussions
- Mock placement aptitude tests
- Case-based logical reasoning activities
- Competitive aptitude quizzes
- Shortcut calculation exercises
- Peer discussion and analytical problem solving

Suggested Mini Project / Analytical Assignment

E-Commerce Sales Analysis Project	
Problem Statement	Analyze e-commerce sales data using arithmetic reasoning and quantitative analysis techniques to identify trends, patterns and business insights.
Activities	• Identify key metrics such as total sales, average order value and growth rate • Organize and clean the sales dataset • Perform percentage, ratio and average-based analysis • Interpret trends using quantitative reasoning techniques • Prepare analytical reports and visual summaries • Present recommendations based on data insights

Text and Reference Books

Suggested Text Books:

- Quantitative Aptitude for Competitive Examinations — R. S. Aggarwal, Revised Edition, S. Chand Publishing, 2023.
- A Modern Approach to Logical Reasoning — R. S. Aggarwal, Revised Edition, S. Chand Publishing, 2022.
- How to Prepare for Quantitative Aptitude for CAT — Arun Sharma, 11th Edition, McGraw Hill Education, 2024.

Recommended Reference Books:

- Fast Track Objective Arithmetic — Rajesh Verma, Arihant Publications, 2023.
- Quantitative Aptitude Quantum CAT — Sarvesh K. Verma, Arihant Publications, 2023.
- Data Interpretation and Logical Reasoning — Pearson Education, 2022.

Recommended Online Resources:

- IndiaBix Aptitude Portal — https://www.indiabix.com/online-test/aptitude-test/
- GeeksforGeeks Aptitude Resources — https://www.geeksforgeeks.org/aptitude-questions-and-answers/
- Hitbullseye Aptitude Preparation — https://www.hitbullseye.com/

Discipline Specific Electives-II						
1	**CSE005**	Distributed Systems	**3**	**0**	**2**	**4**
2	**CSE006**	Compiler Design	**3**	**0**	**2**	**4**
3	**CSE007**	Computer Vision & Image Processing	**3**	**0**	**2**	**4**
4	**CSE008**	Game Development with Unity	**3**	**0**	**2**	**4**

DISTRIBUTED SYSTEMS

Program Name	B.Tech. CSE Core			
Course Name: Distributed Systems	Course Code	L-T-P	Credits	Contact Hours
	CSE005	3-0-2	04	45
Type of Course:	DSE-II			
Semester:	VI (Sixth Semester)			
Pre-requisite(s):	Computer Networks, Operating Systems, Databases and Basic Linux Programming Knowledge			

Course Description

This course introduces the concepts and technologies used in modern distributed systems for developing scalable and reliable computing applications. The course focuses on distributed communication, synchronization, distributed coordination, cloud-native systems, microservices and fault tolerance using practical industry-oriented tools and frameworks.

Students will learn how modern distributed systems coordinate multiple computing nodes, exchange data, manage scalability and support reliable applications used in cloud platforms, web systems and enterprise applications. The course emphasizes practical implementation of distributed applications, APIs, containers, distributed databases and monitoring systems rather than deep theoretical algorithms.

The laboratory component focuses on hands-on implementation of distributed applications using Docker, Kubernetes, REST APIs, Redis, MongoDB, Kafka and Python-based distributed communication frameworks. Students will gain practical exposure to microservices, container deployment, distributed communication and scalable application development aligned with current industry requirements.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Course Outcome	Bloom's Level
CO1	Explain distributed system concepts and communication models.	L2 – Understand
CO2	Apply distributed communication and coordination techniques.	L3 – Apply
CO3	Develop cloud-native distributed applications.	L3 – Apply
CO4	Analyze scalability, reliability and fault tolerance in distributed systems.	L4 – Analyze

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	–	2	–	–	1	1	–	2
CO2	3	3	2	2	3	1	1	2	2	1	2
CO3	3	3	3	2	3	1	1	2	2	2	2

CO4	3	3	3	3	3	2	2	2	2	2	3

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	2	1	–	1
CO2	3	3	2	2	2
CO3	3	3	2	2	3
CO4	3	3	3	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Fundamentals of Distributed Systems & Communication	No. of Hours: 12

- Introduction to Distributed Systems: characteristics, architecture and real-world applications
- Client-Server & Peer-to-Peer Systems
- Communication Models: sockets, REST APIs and message-based communication
- Remote Communication Concepts: RPC and gRPC basics
- Process Synchronization & Distributed Coordination Basics
- Logical Clocks & Event Ordering Concepts
- Distributed Naming & DNS Basics
- Introduction to Distributed Application Development using Python
- Practical Activities: distributed communication using APIs and sockets

Unit Number: 2	Distributed Coordination, Databases & Fault Tolerance	No. of Hours: 12

- Distributed Coordination Concepts
- Leader Election & Consensus Basics
- Consistency Models: strong consistency and eventual consistency
- Replication & Distributed Caching Basics
- Fault Tolerance & Recovery Mechanisms
- Distributed Databases: MongoDB and NoSQL fundamentals
- Distributed File Storage Concepts
- Redis for Distributed Caching
- Practical Activities: distributed database communication and caching systems

Unit Number: 3	Cloud-Native Systems & Microservices	No. of Hours:

		11

- Cloud-Native Application Fundamentals
- Containerization using Docker
- Docker Compose & Multi-Container Applications
- Introduction to Kubernetes & Container Orchestration
- Microservices Architecture Basics
- API Gateway & Service Communication Concepts
- Message Queues & Kafka Basics
- Distributed Monitoring & Logging Basics
- Practical Activities: microservices deployment and container orchestration

Unit Number: 4	Scalable Distributed Applications & Modern Trends	No. of Hours: 10

- Scalability Concepts & Load Balancing
- Distributed Monitoring & Observability
- Secure Distributed Communication Basics
- Stream Processing & Real-Time Communication Concepts
- Edge Computing & IoT Communication Basics
- Serverless Computing Fundamentals
- Case Studies: Netflix, Google, Kubernetes and modern distributed systems
- Mini Project Development: scalable distributed application or monitoring system
- Practical Activities: scalable deployment and distributed system analysis

Lab Experiments – CO Mapping Table

S. No.	Experiment Title	Tools / Technologies Used	Mapped CO(s)	Bloom's Level
1	Distributed Communication using APIs & Sockets	Python, Flask, Socket Programming	CO1, CO2	L3 – Apply
2	Distributed Database & Caching System	MongoDB, Redis, Python	CO2	L3 – Apply
3	Microservices & Container Deployment Platform	Docker, Docker Compose, Flask	CO3	L3 – Apply
4	Scalable Distributed Monitoring & Messaging System	Kafka, Prometheus, Grafana	CO4	L4 – Analyze
5	Capstone: Distributed Enterprise Application Platform	Docker, MongoDB, Redis, Kubernetes	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Distributed Communication using APIs & Sockets	CO1, CO2

Tools / Technologies Used

Python, Flask, Socket Programming

Problem Statement

Develop distributed communication systems using APIs and socket-based communication techniques.

Sub-Problems

- Implement client-server communication using sockets
- Create REST APIs using Flask
- Exchange messages between distributed applications
- Simulate distributed communication workflows
- Analyze communication latency and response times
- Implement simple distributed synchronization logic
- Generate distributed communication reports

Concepts Covered

Distributed communication, sockets and REST APIs

Unit 2 Lab Assignment: Distributed Database & Caching System	**CO2**

Tools / Technologies Used

MongoDB, Redis, Python

Problem Statement

Develop distributed storage and caching systems using NoSQL databases and distributed caching tools.

Sub-Problems

- Configure MongoDB distributed database operations
- Store and retrieve distributed application data
- Implement distributed caching using Redis
- Simulate replication and data synchronization
- Analyze consistency and response times
- Implement fault recovery using backup datasets
- Generate distributed database performance reports

Concepts Covered

Distributed databases, caching and replication

Unit 3 Lab Assignment: Microservices & Container Deployment Platform	**CO3**

Tools / Technologies Used

Docker, Docker Compose, Flask

Problem Statement

Develop and deploy microservices-based distributed applications using container technologies.

Sub-Problems

- Create containerized applications using Docker
- Configure multi-container applications using Docker Compose
- Develop simple microservices using Flask APIs
- Enable communication between microservices
- Deploy distributed applications in containers
- Monitor container resource usage
- Generate deployment and orchestration reports

Concepts Covered

Microservices, Docker and cloud-native deployment

Unit 4 Lab Assignment: Scalable Distributed Monitoring & Messaging System	**CO4**

Tools / Technologies Used

Kafka, Prometheus, Grafana

Problem Statement

Develop scalable distributed monitoring and messaging systems for analyzing distributed application performance.

Sub-Problems

- Configure Kafka message communication workflows
- Implement producer-consumer messaging systems
- Monitor distributed applications using Prometheus
- Visualize system metrics using Grafana dashboards
- Analyze scalability and resource utilization
- Simulate distributed system failures and recovery
- Generate monitoring and analytics reports

Concepts Covered

Distributed messaging, monitoring and scalability

Capstone Lab Assignment: Distributed Enterprise Application Platform	**CO1, CO2, CO3, CO4**

Tools / Technologies Used

Docker, Kubernetes, MongoDB, Redis, Flask

Problem Statement

Design and implement a scalable distributed enterprise application using microservices, distributed databases and container-based deployment.

Sub-Problems

- Develop distributed application modules using Flask APIs
- Configure distributed communication between services
- Integrate MongoDB and Redis for storage and caching
- Deploy applications using Docker containers
- Configure Kubernetes-based scalable deployment

- Monitor distributed applications and resource usage
- Analyze scalability, reliability and system performance
- Generate technical documentation and deployment reports

Concepts Covered

Distributed systems, microservices, scalability and fault tolerance

Text and Reference Books

Textbooks:

1. Maarten van Steen & Andrew S. Tanenbaum, "Distributed Systems," 4th Edition, distributed-systems.net, 2023. ISBN: 9789081540612
2. Brendan Burns, "Designing Distributed Systems," O'Reilly Media, 2018. ISBN: 9781491983645

Recommended Reference Books:

1. Sam Newman, "Building Microservices," 2nd Edition, O'Reilly Media, 2021. ISBN: 9781492034025
2. Kelsey Hightower, Brendan Burns & Joe Beda, "Kubernetes Up and Running," 3rd Edition, O'Reilly Media, 2022. ISBN: 9781098110208
3. John Arundel & Justin Domingus, "Cloud Native DevOps with Kubernetes," O'Reilly Media, 2019. ISBN: 9781492040767
4. Chris Richardson, "Microservices Patterns," Manning Publications, 2018. ISBN: 9781617294549
5. Pethuru Raj & Jeeva S. Chelladhurai, "Learning Docker," 2nd Edition, Packt Publishing, 2020. ISBN: 9781839211898

Recommended Online Resources:

- Docker Documentation — https://docs.docker.com/
- Kubernetes Documentation — https://kubernetes.io/docs/
- Apache Kafka Documentation — https://kafka.apache.org/documentation/
- MongoDB Documentation — https://www.mongodb.com/docs/
- Redis Documentation — https://redis.io/docs/
- Prometheus Documentation — https://prometheus.io/docs/
- Grafana Documentation — https://grafana.com/docs/
- NPTEL Distributed Systems — https://nptel.ac.in/courses/106106168

COMPILER DESIGN

Program Name	B.Tech. CSE Core			
Course Name: Compiler Design	Course Code	L-T-P	Credits	Contact Hours
	CSE006	3-0-2	04	45
Type of Course:	DSE II			
Semester:	VI (Sixth Semester)			
Pre-requisite(s):	Discrete Mathematics, Formal Languages and Automata Theory, Programming Languages			

Course Description

This course introduces the principles, techniques, and tools used in the design and implementation of compilers and language processing systems. The course focuses on lexical analysis, syntax analysis, semantic analysis, intermediate code generation, optimization techniques, runtime environments, and code generation for modern programming languages.

Students will learn how source programs are translated into executable machine code using compiler phases and language processing algorithms. The course also introduces parser generation tools, syntax-directed translation, error handling, code optimization, and modern compiler infrastructures used in software systems, interpreters, and programming language environments.

The laboratory component emphasizes practical implementation of lexical analyzers, parsers, syntax trees, intermediate code generators, and optimization techniques using compiler construction tools and programming languages. By the end of the course, students will be able to analyze language grammars, design compiler components, optimize program execution, and develop language processing systems for modern computing platforms.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Concise Course Outcome	Bloom's Level
CO1	Explain compiler phases and language processing concepts.	L2 – Understand
CO2	Analyze lexical, syntax, and semantic structures of programming languages.	L4 – Analyze
CO3	Develop parsers, intermediate code generators, and optimization techniques.	L3 – Apply
CO4	Evaluate compiler systems for correctness, efficiency, and optimization.	L5 – Evaluate

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	–	2	–	–	1	1	–	2

CO2	3	3	2	2	3	1	–	1	2	1	2
CO3	3	3	3	2	3	1	1	2	2	1	2
CO4	3	3	3	3	3	1	1	2	2	2	3

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	2	1	–	1
CO2	3	3	3	2	2
CO3	3	3	2	2	3
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Introduction to Compilers and Lexical Analysis	No. of Hours: 12

- Introduction to Compiler Design: compiler phases, interpreters, assemblers and language processing systems
- Programming Languages and Formal Grammars: tokens, alphabets, strings and finite automata
- Lexical Analysis: token recognition, lexical errors and scanner generation
- Finite Automata: DFA, NFA and conversion techniques
- Regular Expressions: pattern matching and lexical specification
- Lexical Analyzer Generation Tools: Lex/Flex fundamentals
- Error Handling in Lexical Analysis: recovery mechanisms and diagnostics

Unit Number: 2	Syntax Analysis and Semantic Analysis	No. of Hours: 12

- Context-Free Grammars: derivations, parse trees and ambiguity
- Top-Down Parsing: recursive descent parsing and predictive parsing
- Bottom-Up Parsing: shift-reduce parsing, LR parsing and operator precedence parsing
- Parser Generation Tools: Yacc/Bison and parser construction
- Syntax-Directed Translation: attributed grammars and semantic rules
- Semantic Analysis: type checking, scope handling and symbol table management
- Error Recovery in Parsing: panic mode and phrase-level recovery

Unit Number: 3	Intermediate Code Generation and Runtime Environments	No. of Hours: 11

- Intermediate Representations: syntax trees, DAGs and three-address code
- Intermediate Code Generation: postfix notation and quadruples
- Runtime Environments: memory layout, activation records and parameter passing
- Symbol Table Management: hashing and scope resolution techniques
- Type Systems and Type Checking: static and dynamic typing
- Code Optimization Basics: common subexpression elimination and dead code elimination

Unit Number: 4	Code Optimization and Code Generation	No. of Hours: 10

- Machine-Independent Optimization: loop optimization and data flow analysis
- Machine-Dependent Optimization: instruction scheduling and register allocation
- Code Generation Techniques: target code generation and assembly code translation
- Compiler Back-End Design: runtime support and memory management
- Modern Compiler Infrastructures: LLVM and JIT compilation concepts
- Parallel and Distributed Compilation: multicore compilation techniques
- Case Studies: GCC, LLVM and modern language compilers

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Lexical Analyzer & Token Processing Toolkit	CO1	L2 – Understand
2	Syntax Parsing & Grammar Analysis System	CO2	L4 – Analyze
3	Intermediate Code Generation & Optimization Framework	CO3	L3 – Apply
4	Compiler Optimization & Code Generation Framework	CO4	L4 – Analyze
5	Capstone: Mini Compiler & Language Processing System	CO1, CO2, CO3, CO4	L5 – Evaluate

Lab Assignments

Unit 1 Lab Assignment: Lexical Analyzer & Token Processing Toolkit	CO1

Problem Statement

Design a lexical analysis toolkit that recognizes tokens and processes programming language constructs using automata-based techniques.

Sub-Problems

- Implement token recognition using regular expressions

- Design DFA and NFA for lexical patterns
- Develop lexical analyzers using Lex/Flex tools
- Identify keywords, identifiers, operators and literals
- Handle lexical errors and invalid tokens
- Analyze token frequency and lexical statistics
- Generate lexical analysis reports and token streams

Concepts Covered: lexical analysis, finite automata, regular expressions and scanner generation

Unit 2 Lab Assignment: Syntax Parsing & Grammar Analysis System	**CO2**

Problem Statement

Develop a grammar analysis and parsing system that validates source code syntax using compiler parsing techniques.

Sub-Problems

- Construct context-free grammars for programming languages
- Implement recursive descent and predictive parsers
- Generate parse trees and syntax trees
- Implement LR parsing using parser generation tools
- Handle syntax errors and recovery mechanisms
- Analyze grammar ambiguity and precedence rules
- Generate parser performance and grammar reports

Concepts Covered: parsing, CFGs, syntax trees and parser generation

Unit 3 Lab Assignment: Intermediate Code Generation & Semantic Analyzer	**CO3**

Problem Statement

Create a semantic analysis and intermediate code generation framework for source code translation.

Sub-Problems

- Build symbol tables and scope management systems
- Perform semantic analysis and type checking
- Generate intermediate representations and three-address code
- Implement syntax-directed translation schemes
- Optimize intermediate code using basic optimization techniques
- Analyze runtime memory allocation mechanisms
- Generate semantic and code translation reports

Concepts Covered: semantic analysis, symbol tables, intermediate code generation and optimization

Unit 4 Lab Assignment: Compiler Optimization & Code Generation Framework	**CO4**

Problem Statement

Develop a compiler optimization and code generation framework that improves execution efficiency

and generates target code.

Sub-Problems

- Implement machine-independent optimization techniques
- Perform register allocation and instruction scheduling
- Generate assembly-level target code
- Analyze runtime performance and memory utilization
- Implement basic LLVM or JIT compilation concepts
- Compare optimized and non-optimized code performance
- Generate optimization and target code generation reports

Concepts Covered: code optimization, target code generation, LLVM and runtime environments

Capstone Lab Assignment: Mini Compiler & Language Processing System	**CO1, CO2, CO3, CO4**

Problem Statement

Design and implement a mini compiler or interpreter system for a custom programming language using compiler construction principles.

Sub-Problems

- Design lexical and syntax specifications for a language
- Implement lexical analyzers and parsers
- Perform semantic analysis and symbol table management
- Generate intermediate and target code representations
- Apply optimization and runtime support techniques
- Analyze compiler correctness and execution efficiency
- Generate technical documentation and compiler design reports
- Evaluate language processing system scalability and extensibility

Concepts Covered: compiler construction, parsing, semantic analysis, optimization and language processing

Text and Reference Books

Textbooks:

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ullman, “Compilers: Principles, Techniques, and Tools,” 2nd Edition, Pearson, 2006. ISBN: 978-0321486813
2. Keith D. Cooper and Linda Torczon, “Engineering a Compiler,” 2nd Edition, Morgan Kaufmann, 2011. ISBN: 978-0120884786

Recommended Reference Books:

1. Steven S. Muchnick, “Advanced Compiler Design and Implementation,” 1st Edition, Morgan Kaufmann, 1997. ISBN: 978-1558603208
2. Dick Grune, Henri E. Bal, Ceriel J.H. Jacobs and Koen Langendoen, “Modern Compiler Design,” 2nd Edition, Springer, 2012. ISBN: 978-1461446985
3. Terence Parr, “Language Implementation Patterns,” 1st Edition, Pragmatic Bookshelf, 2009. ISBN: 978-1934356452

4. Allen I. Holub, “Compiler Design in C,” 1st Edition, Prentice Hall, 1990. ISBN: 978-0131550458

Recommended Online Resources:

- LLVM Documentation — https://llvm.org/docs/
- GNU Compiler Collection Documentation — https://gcc.gnu.org/onlinedocs/
- Flex Documentation — https://westes.github.io/flex/manual/
- Bison Documentation — https://www.gnu.org/software/bison/manual/
- NPTEL – Compiler Design — https://nptel.ac.in
- ANTLR Parser Generator — https://www.antlr.org

COMPUTER VISION & IMAGE PROCESSING

Program Name	B.Tech. CSE Core			
Course Name: Computer Vision & Image Processing	Course Code	L-T-P	Credits	Contact Hours
	CSE007	3-0-2	04	45
Type of Course:	Discipline Specific Elective (DSE II)			
Semester:	VI (Sixth Semester)			
Pre-requisite(s):	Basic knowledge of programming, linear algebra, probability and fundamentals of machine learning			

Course Description

This course introduces students to the principles and applications of computer vision and digital image processing with a strong focus on practical implementation using Python, OpenCV and deep learning frameworks. The course covers image enhancement, filtering, feature extraction, segmentation, object detection, video analytics and AI-driven computer vision systems.

Students will learn how modern vision systems are designed for applications such as facial recognition, medical image analysis, autonomous systems, surveillance and industrial automation. The course emphasizes hands-on experimentation, dataset processing, model training, real-time vision analytics and deployment-oriented workflows aligned with current AI and computer vision industry requirements.

The laboratory component focuses on image manipulation, segmentation, object detection, CNN-based image classification, video analytics and real-world computer vision application development using modern open-source tools and frameworks.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Course Outcome	Bloom's Level
CO1	Apply image processing techniques for image enhancement and transformation.	L3 – Apply
CO2	Design computer vision systems using segmentation and feature extraction techniques.	L3 – Apply
CO3	Implement deep learning models for image classification and detection.	L3 – Apply
CO4	Analyze computer vision applications for real-world problem solving.	L4 – Analyze

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	1	3	-	-	-	-	-	2

CO2	3	3	3	2	3	1	-	1	-	-	2
CO3	3	3	3	2	3	1	-	1	1	-	3
CO4	2	3	2	3	2	2	1	1	1	1	3

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts	PSO2 Applying Computing Techniques	PSO3 Analyzing Complex Problems	PSO4 Evaluating Solutions	PSO5 Developing Innovative Applications
CO1	3	3	2	1	2
CO2	3	3	3	2	3
CO3	3	3	3	2	3
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Fundamentals of Digital Image Processing	No. of Hours: 12

- Introduction to Computer Vision & Image Processing: applications and vision system overview
- Digital Image Representation: grayscale images, color images and color models
- Image Acquisition & Sampling Techniques
- Image Enhancement Techniques: contrast stretching, histogram equalization and intensity transformations
- Spatial & Frequency Domain Filtering
- Edge Detection Techniques: Sobel, Canny and Laplacian operators
- Morphological Image Processing: erosion, dilation, opening and closing operations
- Image Transformation & Geometric Operations
- Introduction to OpenCV & Image Manipulation Using Python

Unit Number: 2	Image Segmentation & Feature Extraction	No. of Hours: 12

- Image Segmentation Techniques: thresholding, region growing and clustering-based segmentation
- Contour Detection & Shape Analysis
- Corner & Interest Point Detection
- Feature Extraction Techniques: SIFT, SURF and ORB descriptors
- Texture & Pattern Analysis
- Object Detection Fundamentals
- Template Matching & Feature Matching Techniques
- Feature Descriptors & Image Representation Methods
- Applications in Medical Imaging & Industrial Inspection

Unit Number: 3	Machine Learning & Deep Learning for Vision	No. of Hours: 11

- Introduction to Machine Learning for Vision Systems
- Image Classification Techniques
- Convolutional Neural Networks (CNNs): architecture and working principles
- Transfer Learning & Pretrained Vision Models
- Object Recognition & Image Labelling
- Deep Learning Frameworks for Vision Applications
- Training & Evaluation of Vision Models
- Introduction to YOLO & Real-Time Object Detection
- Image Augmentation & Dataset Preparation Techniques

Unit Number: 4	Advanced Computer Vision Applications	No. of Hours: 10

- Video Processing & Motion Analysis
- Optical Flow & Object Tracking Techniques
- Facial Recognition & Biometric Systems
- Autonomous Vision Systems & Self-Driving Concepts
- Medical Image Analysis & Healthcare Applications
- Surveillance & Security Analytics
- Introduction to AR/VR-Based Vision Applications
- Ethical Issues & Bias in Computer Vision Systems
- Deployment & Optimization of AI-Powered Vision Applications

Lab Experiments – CO Mapping Table

S. No.	Experiment Title	Tools / Technologies Used	Mapped CO(s)	Bloom's Level
1	Image Enhancement & Filtering Techniques	Python, OpenCV, Jupyter Notebook	CO1	L3 – Apply
2	Image Segmentation & Feature Extraction System	OpenCV, Python, NumPy	CO2	L3 – Apply
3	Deep Learning-Based Image Classification & Detection	TensorFlow/PyTorch, OpenCV	CO3	L3 – Apply
4	Real-Time Video Analytics & Vision Applications	OpenCV, YOLO, Python	CO4	L4 – Analyze
5	Capstone: Intelligent Vision-Based Smart Monitoring System	Python, OpenCV, TensorFlow/PyTorch	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Image Enhancement & Filtering Techniques	CO1

Tools / Technologies Used

Python, OpenCV, Jupyter Notebook

Problem Statement

Develop image processing applications to perform enhancement, filtering and transformation operations on digital images.

Sub-Problems

- Read and display grayscale and color images
- Apply image enhancement techniques
- Perform histogram equalization and contrast adjustment
- Implement spatial filtering operations
- Detect edges using Sobel and Canny operators
- Apply morphological image transformations
- Generate transformed and enhanced output images

Concepts Covered

Image enhancement, filtering, edge detection and OpenCV fundamentals

Unit 2 Lab Assignment: Image Segmentation & Feature Extraction System	CO2

Tools / Technologies Used

OpenCV, Python, NumPy

Problem Statement

Design computer vision systems to segment images and extract meaningful visual features for object analysis.

Sub-Problems

- Perform image thresholding and segmentation
- Implement contour and region detection
- Detect corners and feature points
- Apply SIFT, SURF or ORB descriptors
- Perform template and feature matching
- Analyze shapes and textures in images
- Generate segmented and annotated visual outputs

Concepts Covered

Segmentation, contour analysis and feature extraction

Unit 3 Lab Assignment: Deep Learning-Based Image Classification & Detection	CO3

Tools / Technologies Used

TensorFlow/PyTorch, OpenCV, Google Colab

Problem Statement

Build deep learning models for image classification, object recognition and detection tasks.

Sub-Problems

- Prepare image datasets for training and testing
- Implement CNN-based image classification models
- Apply transfer learning using pretrained models
- Train and evaluate image recognition systems
- Perform object detection using YOLO or OpenCV DNN
- Apply image augmentation techniques
- Generate prediction and performance evaluation reports

Concepts Covered

CNNs, transfer learning and object detection

Unit 4 Lab Assignment: Real-Time Video Analytics & Vision Applications	**CO4**

Tools / Technologies Used

OpenCV, YOLO, Python, Webcam/Video Datasets

Problem Statement

Develop real-time computer vision applications for surveillance, tracking and intelligent video analytics.

Sub-Problems

- Capture and process real-time video streams
- Implement motion detection and object tracking
- Perform facial recognition and identification
- Detect objects in live video feeds
- Analyze traffic or crowd movement patterns
- Generate automated alerts and monitoring outputs
- Build basic AR/VR interactive visual modules

Concepts Covered

Video analytics, tracking and intelligent monitoring systems

Capstone Lab Assignment: Intelligent Vision-Based Smart Monitoring System	**CO1, CO2, CO3, CO4**

Tools / Technologies Used

Python, OpenCV, TensorFlow/PyTorch, YOLO, Google Colab

Problem Statement

Design and develop an AI-powered smart monitoring system integrating image processing, deep learning and real-time computer vision analytics for applications such as surveillance, healthcare or smart transportation.

Sub-Problems

- Acquire and preprocess image/video datasets
- Apply image enhancement and segmentation techniques
- Implement object detection and recognition modules
- Develop deep learning-based classification systems
- Integrate real-time video analytics and tracking
- Generate alerts and automated monitoring reports
- Evaluate system accuracy and performance metrics
- Deploy a complete intelligent computer vision solution
- Optimize the system for scalability and real-time processing
- Present a fully functional AI-driven vision application

Concepts Covered

Image processing, deep learning, object detection and real-time vision analytic

Text and Reference Books

Textbooks:

1. Rafael C. Gonzalez & Richard E. Woods, "Digital Image Processing," 4th Edition, Pearson Education, 2018. ISBN: 9780133356724
2. Richard Szeliski, "Computer Vision: Algorithms and Applications," 2nd Edition, Springer, 2022. ISBN: 9783030343729
3. Adrian Kaehler & Gary Bradski, "Learning OpenCV 4 Computer Vision with Python," Packt Publishing, 2019. ISBN: 9781789531619

Recommended Reference Books:

1. Jan Erik Solem, "Programming Computer Vision with Python," O'Reilly Media, 2012. ISBN: 9781449316549
2. Ian Goodfellow, Yoshua Bengio & Aaron Courville, "Deep Learning," MIT Press, 2016. ISBN: 9780262035613
3. Mark Nixon & Alberto Aguado, "Feature Extraction and Image Processing for Computer Vision," 4th Edition, Academic Press, 2019. ISBN: 9780128149768
4. Simon J. D. Prince, "Computer Vision: Models, Learning, and Inference," Cambridge University Press, 2012. ISBN: 9781107011793

Recommended Online Resources:

- OpenCV Documentation — https://docs.opencv.org/
- TensorFlow Documentation — https://www.tensorflow.org/learn
- PyTorch Documentation — https://pytorch.org/docs/
- YOLO (Ultralytics) Documentation — https://docs.ultralytics.com/
- Kaggle Datasets & Competitions — https://www.kaggle.com/
- Google Colab — https://colab.research.google.com/

GAME DEVELOPMENT WITH UNITY

Program Name	B.Tech. CSE (Core)			
Course Name: Game Development with Unity	Course Code	L-T-P	Credits	Contact Hours
	CSE008	3-0-2	4	45
Type of Course:	Discipline Specific Elective (DSE II)			
Semester:	VI (Sixth Semester)			
Pre-requisite(s):	Basic understanding of programming concepts, object-oriented programming and fundamentals of computer graphics			

Course Description

This course introduces students to modern game development using the Unity game engine and C# programming. The course focuses on practical game development workflows including 2D/3D game design, character movement, physics simulation, animation systems, artificial intelligence, UI systems, multiplayer concepts, and cross-platform deployment.

Students will learn how modern games are designed, developed, optimized, tested, and deployed using industry-standard tools and workflows. The course emphasizes hands-on implementation, creativity, problem-solving, and interactive game production aligned with current gaming and immersive technology industry requirements.

The laboratory component focuses on real-world game prototyping, gameplay mechanics, animation systems, AI-driven interactions, multiplayer systems, and deployment-oriented game development using Unity and C#.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Develop interactive 2D and 3D games using Unity.	L3 – Apply
CO2	Design game environments, animations and physics systems.	L3 – Apply
CO3	Implement AI, UI and multiplayer gameplay systems.	L3 – Apply
CO4	Optimize and deploy games across multiple platforms.	L4 – Analyze

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	3	1	3	-	-	1	1	-	2
CO2	3	2	3	2	3	1	-	1	1	-	2

CO3	3	3	3	2	3	1	-	2	1	1	3
CO4	2	3	2	3	3	2	1	1	1	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts	PSO2 Applying Computing Techniques	PSO3 Analyzing Complex Problems	PSO4 Evaluating Solutions	PSO5 Developing Innovative Applications
CO1	3	3	2	1	3
CO2	3	3	3	2	3
CO3	3	3	3	2	3
CO4	2	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Fundamentals of Game Development & Unity	No. of Hours: 12

- Introduction to Game Development: game development lifecycle and modern gaming industry overview
- Unity Interface & Project Structure: Unity editor, scenes, assets and project management
- C# Scripting Fundamentals for Unity: variables, functions, classes and game scripting basics
- Game Objects & Components: prefabs, transforms and reusable game objects
- Scene Management & Player Interaction: scene creation and object interaction systems
- 2D & 3D Coordinate Systems: positioning, scaling and transformations
- Input Handling & Player Controls: keyboard, mouse, touch and controller input handling
- Camera & Lighting Systems: camera movement, cinematic views and lighting fundamentals
- Building Simple 2D & 3D Game Prototypes

Unit Number: 2	Animation, Physics & Environment Design	No. of Hours: 11

- Game Design & Storytelling Basics: gameplay mechanics and player experience design
- Character Design & Environment Creation: terrain, objects and interactive worlds
- Animation Systems: Animator Controller, sprite animation and skeletal animation
- Physics Engine Fundamentals: Rigidbody, forces, gravity and movement systems
- Collision Detection & Trigger Systems
- Particle Systems & Visual Effects: explosions, weather and environment effects
- Audio Systems: background music, sound effects and environmental audio integration
- Level Design & Interactive Gameplay Mechanics

Unit Number: 3	AI, UI Systems & Gameplay Mechanics	No. of Hours: 11

- Artificial Intelligence for Games: NPC behavior and gameplay decision systems
- Finite State Machines (FSM): enemy and character behavior systems
- Pathfinding & Navigation Meshes: movement and navigation systems
- Inventory, Health & Scoring Systems
- UI Development Using Unity Canvas: menus, buttons, scoreboards and HUD systems
- Game Events & Interaction Systems
- Saving & Loading Game Data
- Introduction to AR/VR Game Development Concepts
- Gameplay Optimization & User Experience Enhancement

Unit Number: 4	Multiplayer Systems & Game Deployment	No. of Hours: 11

- Introduction to Multiplayer Game Architecture
- Networking Basics & Synchronization Concepts
- Real-Time Multiplayer Communication
- Backend & Cloud Integration Basics
- Performance Optimization Techniques: frame rate optimization and memory management
- Cross-Platform Game Development: desktop, web and mobile deployment workflows
- Game Testing & Debugging Techniques
- Game Publishing & Deployment Strategies
- Ethical & Legal Considerations in Game Development

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Tools / Technologies Used	Mapped CO(s)	Bloom's Level
1	2D & 3D Game Prototype Development	Unity, C#, Visual Studio	CO1	L3 – Apply
2	Character Animation & Physics-Based Gameplay	Unity Animator, Rigidbody System	CO2	L3 – Apply
3	AI-Driven Gameplay & Interactive UI Systems	Unity NavMesh, FSM, Canvas UI	CO3	L3 – Apply
4	Multiplayer Integration & Cross-Platform Deployment	Unity Multiplayer Tools, Firebase	CO4	L4 – Analyze
5	Capstone: Complete Cross-Platform Unity Game Project	Unity, C#, Firebase, Visual Studio	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: 2D & 3D Game Prototype Development	CO1
Tools / Technologies Used	Unity Engine, C#, Visual Studio
Problem Statement	Develop interactive 2D and 3D game prototypes using Unity game engine and C# scripting fundamentals.
Sub-Problems	• Create Unity projects and organize game assets • Design interactive 2D and 3D game scenes • Implement player movement and controls • Create reusable prefabs and game objects • Handle keyboard, mouse and touch input systems • Configure camera movement and lighting systems • Build a playable 2D or 3D game prototype
Concepts Covered	Unity fundamentals, scene management, input systems and game scripting

Unit 2 Lab Assignment: Character Animation & Physics-Based Gameplay	CO2
Tools / Technologies Used	Unity Animator, Rigidbody Physics, Particle System
Problem Statement	Design animated characters and implement realistic physics interactions in Unity-based games.
Sub-Problems	• Create character animations using Animator Controller • Implement sprite and skeletal animations • Apply Rigidbody physics and collision systems • Design interactive game environments and terrains • Add particle effects and environmental sound systems • Implement physics-based movement and interactions • Develop a small gameplay module with animated characters
Concepts Covered	Animation systems, physics engine and gameplay mechanics

Unit 3 Lab Assignment: AI-Driven Gameplay & Interactive UI Systems	CO3
Tools / Technologies Used	Unity NavMesh, FSM, Canvas UI System, C#
Problem	Develop intelligent gameplay systems and interactive user interfaces for

Statement	modern game applications.
Sub-Problems	• Implement NPC behavior using finite state machines • Create pathfinding systems using NavMesh • Design inventory, health and scoring systems • Build game menus, HUD and UI interfaces • Integrate audio and event-driven gameplay systems • Save and load gameplay progress data • Develop AI-driven enemy or companion systems
Concepts Covered	Game AI, UI systems, pathfinding and gameplay optimization

Unit 4 Lab Assignment: Multiplayer Integration & Cross-Platform Deployment	**CO4**
Tools / Technologies Used	Unity Multiplayer Tools, Firebase, Visual Studio
Problem Statement	Implement multiplayer functionality and deploy optimized Unity games for multiple platforms.
Sub-Problems	• Configure multiplayer networking basics • Synchronize player movement and game states • Integrate cloud/backend services for gameplay data • Optimize game performance for desktop and mobile devices • Perform debugging and gameplay testing • Package and deploy games for multiple platforms • Generate deployment and testing reports
Concepts Covered	Multiplayer systems, networking, optimization and deployment

Capstone Lab Assignment: Complete Cross-Platform Unity Game Project	**CO1, CO2, CO3, CO4**
Tools / Technologies Used	Unity Engine, C#, Firebase, Visual Studio, Unity Multiplayer Services
Problem Statement	Design and develop a complete Unity-based game integrating gameplay mechanics, AI systems, animations, multiplayer functionality and deployment workflows for real-world gaming applications.
Sub-Problems	• Design game concept, storyline and gameplay mechanics • Develop interactive 2D/3D environments and characters • Implement animations and physics-based interactions

	• Integrate AI-driven enemy and gameplay systems • Build UI systems, scoring and progression modules • Implement multiplayer or online gameplay features • Optimize and test the game across platforms • Deploy the game for desktop, web or mobile platforms • Prepare technical documentation and gameplay reports • Present a fully functional and polished game application
Concepts Covered	Game development lifecycle, AI systems, multiplayer gameplay and deployment workflows

Text and Reference Books

Textbooks:

1. Unity in Action: Multiplatform Game Development in C#, Joseph Hocking, 3rd Edition, Manning Publications, 2021. ISBN: 9781617294969
2. Unity 2021 Cookbook, Mike Geig, 5th Edition, Packt Publishing, 2021. ISBN: 9781839217616
3. Learning C# by Developing Games with Unity, Jesse Freeman, Packt Publishing, 2021. ISBN: 9781801813945

Recommended Reference Books:

1. Mastering Unity Game Development with C#, Alan Thorn, Packt Publishing, 2020. ISBN: 9781839217333
2. Programming Game AI by Example, Mat Buckland, Jones & Bartlett Learning, 2004. ISBN: 9781556220784
3. Fundamentals of Game Design, Ernest Adams & Andrew Rollings, 4th Edition, Pearson Education, 2019. ISBN: 9780137442683
4. Game Development with Unity for .NET Developers, David Baron, Apress, 2022. ISBN: 9781484277263

Recommended Online Resources:

- Unity Learn Platform — https://learn.unity.com/
- Unity Official Documentation — https://docs.unity3d.com/
- Unity Asset Store — https://assetstore.unity.com/
- Microsoft C# Documentation — https://learn.microsoft.com/en-us/dotnet/csharp/
- Brackeys Unity Tutorials — https://www.youtube.com/c/Brackeys
- Unity Multiplayer Documentation — https://docs.unity.com/multiplayer/
- Coursera Game Development Courses — https://www.coursera.org/courses?query=game%20development
- GeeksforGeeks Game Development Tutorials — https://www.geeksforgeeks.org/game-development/

SEMESTER: VII

PROFESSIONAL PROBLEM SOLVING & CAREER READINESS – II

Program Name	B.Tech. CSE Core			
Course Name: Professional Problem Solving & Career Readiness – II	Course Code	L-T-P	Credits	Contact Hours
	CSE014	2-0-4	04	60 Hours
Type of Course:	Major			
Semester:	VII (Seventh Semester)			
Pre-requisite(s):	Professional Problem Solving & Career Readiness – I; Basic DSA and Programming knowledge			

Course Description

The course "Professional Problem Solving & Career Readiness – II" is an advanced placement preparation course designed to prepare students for modern product-company recruitment processes, coding interviews, system-level technical discussions and corporate assessment workflows.

This course extends the learning from "Professional Problem Solving & Career Readiness – I" by focusing on advanced problem-solving, product-company coding rounds, low-level system design basics, advanced DSA patterns, CS core interview discussions, online assessment simulations and professional interview readiness.

The course emphasizes:

- Product-company coding preparation
- Advanced DSA and problem-solving patterns
- Interview-oriented CS fundamentals
- Low-level system design and API concepts
- Competitive coding and online assessment practice
- Resume, LinkedIn and portfolio readiness
- Technical and HR mock interview simulations
- Real-world placement workflow preparation

The course follows a rigorous practice-oriented model with continuous coding assessments, mock interviews, leaderboard-based evaluations and placement readiness tracking.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level

CO1	Apply advanced problem-solving and coding techniques.	L3 – Apply
CO2	Analyze system-level and CS core interview problems.	L4 – Analyze
CO3	Develop optimized coding and technical interview solutions.	L3 – Apply
CO4	Demonstrate professional readiness for placement processes.	L3 – Apply

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	3	2	3	1	1	1	1	1	3
CO2	3	3	2	2	3	1	1	1	1	1	3
CO3	3	3	3	2	3	1	1	2	1	1	3
CO4	2	2	2	1	2	1	2	3	3	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Structure & Execution Model

Component	Focus Area	Hours	Marks
Module 1	Advanced DSA & Product Company Coding	24 Hours	60
Module 2	CS Fundamentals & System-Level Interview Preparation	16 Hours	40
Module 3	Low-Level Design, APIs & Development Workflows	08 Hours	20
Module 4	Mock Interviews & Placement Simulations	12 Hours	50
Module 5	Final Coding & Placement Assessment	—	30
Total		**60 Hours**	**200**

Course Outline

Unit Number: 1	Advanced DSA & Product Company Coding Preparation	No. of Hours: 24

- Advanced Sliding Window Problems
- Monotonic Stack & Queue Patterns
- Greedy Problem-Solving Techniques
- Binary Search on Answer
- Backtracking & Recursion Optimization
- Trees & Binary Search Trees
- Heaps & Priority Queues
- Graph Traversal Basics for Interviews
- BFS & DFS Coding Problems
- Dynamic Programming Fundamentals

- String Matching & Hashing Problems
- Competitive Coding Strategies
- Time Complexity Optimization Techniques

Unit Number: 2	CS Fundamentals & System-Level Interview Preparation	No. of Hours: 16

- Operating Systems: Process vs Thread, Synchronization & Deadlock Basics, Memory Management Concepts, Virtual Memory & Paging, OS Interview Scenarios
- Database Management Systems: SQL Optimization Basics, Indexing & Query Optimization, Transactions & Concurrency, DBMS Interview Questions
- Computer Networks: OSI & TCP/IP Models, HTTP/HTTPS Basics, DNS & Client-Server Communication, REST API Fundamentals, Socket & Networking Basics
- OOPs & Software Engineering: SOLID Principles Basics, Exception Handling, Collections Framework, API-Centric Development Concepts

Unit Number: 3	Low-Level Design, APIs & Development Workflows	No. of Hours: 08

- Introduction to Low-Level Design (LLD)
- UML Basics & Class Diagrams
- Designing Small Object-Oriented Systems
- REST API Design Basics
- JSON & API Communication
- Authentication & Authorization Basics
- Git & GitHub Collaboration Workflows
- Debugging & Logging Practices
- Clean Code & Coding Standards
- Mini Design Problems: Parking Lot System, Library Management System, Food Delivery Workflow, URL Shortener Basics

Unit Number: 4	Mock Interviews, Placement Simulations & Career Readiness	No. of Hours: 12

- Advanced Technical Mock Interviews
- Product Company Interview Workflows
- HR & Behavioral Interview Preparation
- Resume & LinkedIn Optimization
- Portfolio & GitHub Profile Review
- Group Discussions & Extempore
- Communication & Professional Etiquette
- Corporate Email Writing
- Salary Discussion & Offer Handling Basics

- Online Assessment Simulation
- Placement Simulations: Coding Round, Technical Interview Round, HR Interview Round, Group Discussion Round, Online Aptitude & MCQ Round

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Tools / Technologies Used	Mapped CO(s)	Bloom's Level
1	Advanced DSA Coding Practice	LeetCode, HackerRank	CO1	L3 – Apply
2	Competitive Coding Contest	CodeChef, LMS	CO1	L3 – Apply
3	SQL Optimization & API Exercises	MySQL, Postman	CO2	L4 – Analyze
4	Computer Networks & REST API Practice	Postman, FastAPI	CO2	L4 – Analyze
5	Low-Level Design Mini Project	Java/Python	CO3	L3 – Apply
6	GitHub Collaboration & Debugging Activity	GitHub, VS Code	CO3	L3 – Apply
7	Technical Mock Interview	Interview Panels	CO4	L3 – Apply
8	HR & Placement Simulation	GD / HR Panels	CO4	L3 – Apply

Assessment & Evaluation Scheme

Component	Weightage
Advanced Coding Assessments	60 Marks
CS Fundamentals & Technical Assessments	40 Marks
Low-Level Design & API Activities	20 Marks
Mock Interviews & Placement Simulations	50 Marks
Final Placement Readiness Assessment	30 Marks
Total	**200 Marks**

Mock Interview Evaluation Structure (50 Marks)

Activity	Marks
Technical Mock Interview – I	15
Technical Mock Interview – II	15
HR & Behavioral Mock Interview	10
Resume, GitHub & LinkedIn Evaluation	10
Total	**50**

Evaluation Guidelines

- Continuous internal evaluation shall be followed.
- Students must participate in coding contests, mock interviews and placement simulations.
- Coding assessments shall be conducted using coding platforms and LMS systems.
- Technical interview performance shall be evaluated using rubric-based assessment methods.
- Students must maintain coding profiles and GitHub repositories.
- Minimum 60% overall marks are required for successful completion of the course.
- Placement readiness dashboards may be maintained for monitoring progress and performance.

Recommended Tools & Platforms

- LeetCode
- HackerRank
- CodeChef
- CodeStudio
- GitHub
- VS Code
- Postman
- MySQL
- FastAPI
- Google Forms
- Internal LMS

Text and Reference Books

Textbooks:

- Cracking the Coding Interview – Gayle Laakmann McDowell, 6th Edition, CareerCup, 2015.
- Elements of Programming Interviews in Java – EPI, 2nd Edition, 2017.
- Clean Code – Pearson, 2008.
- Designing Data-Intensive Applications – O'Reilly Media, 2017.

Recommended Online Resources

Resource	URL
LeetCode	https://leetcode.com/
HackerRank	https://www.hackerrank.com/
CodeChef	https://www.codechef.com/
NeetCode	https://neetcode.io/
GeeksforGeeks Interview Prep	https://www.geeksforgeeks.org/interview-preparation-for-software-developer/
Postman Learning Center	https://learning.postman.com/
GitHub Skills	https://skills.github.com/

CAPSTONE PROJECT I

Programme Name	**B.Tech CSE (Core)**		**Semester**	**VII**
Course Name:	**Course Code**	**L-T-P**	**Credits**	**Contact Hours**
Capstone Project I	PRCSE004	0-0-8	4	120
Type of Course:	PROJ	Pre-requisite(s):	Completion of core and specialization courses up to Semester VI	

Course Description

The course is designed to provide students with an opportunity to apply theoretical knowledge, technical competencies, and problem-solving abilities to the development of a comprehensive solution for a real-world problem. The course emphasizes innovation, research orientation, design thinking, and interdisciplinary learning through the execution of individual or team-based projects under faculty mentorship. Carrying academic credit as prescribed in the curriculum, the course serves as an essential component for enhancing practical knowledge, technical proficiency, teamwork, and professional preparedness.

The course focuses on experiential and outcome-based learning by engaging students in activities such as problem identification, requirement analysis, literature review, system design, implementation, testing, documentation, and presentation. Students are encouraged to explore emerging technologies, industry-oriented challenges, research problems, entrepreneurial ideas, or socially relevant applications while developing their projects.

Through the project work, students gain hands-on exposure to project planning, technical documentation, collaborative development, research methodologies, and professional ethics. The course also strengthens communication skills, analytical thinking, creativity, and decision-making abilities while fostering independent learning and innovation. By the end of the project, students are expected to demonstrate the successful integration of domain knowledge and technical skills through the development and presentation of a functional prototype, software solution, research outcome, or engineering model aligned with real-world applications and professional standards.

The Course Outcomes (COs). On completion of the course the participants will learn to:

CO	Course Outcome	Bloom's Level
CO1	Analyze and define a real-world problem statement by applying domain knowledge, research understanding, and requirement analysis techniques.	L4 – Analyze
CO2	Design and develop an appropriate project solution using suitable tools, technologies, methodologies, and engineering practices.	L3 – Apply
CO3	Evaluate the performance, functionality, and effectiveness of the developed project through testing, validation, and result analysis.	L5 – Evaluate
CO4	Prepare technical documentation and present the project outcomes effectively while	L3 –

CO	Course Outcome	Bloom's Level
	demonstrating teamwork, communication skills, and professional ethics.	Apply

Mapping Scale: 1 = Low correlation, 2 = Moderate correlation, 3 = Strong correlation, - = No direct correlation.

CO–PO Articulation Matrix

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	2	3	2	2	1	2	2	2	3
CO2	3	3	3	2	3	2	1	3	2	3	3
CO3	3	3	2	3	3	2	1	2	2	2	3
CO4	2	2	1	2	2	1	3	3	3	2	3

CO–PSO Articulation Matrix

COs \ PSOs	**PSO1**	**PSO2**	**PSO3**	**PSO4**	**PSO5**
CO1	2	2	3	2	2
CO2	2	3	2	3	3
CO3	2	3	3	3	2
CO4	1	2	2	2	2

1. Objectives

All student projects must:

- Employ industry-relevant technologies and tools.
- Reflect a well-defined development or research process from ideation to final deployment or publication.
- Promote innovation and real-world problem-solving.
- Deliver meaningful outcomes such as a functional application, published research, or validated prototype.

2. Project Scope and Categories

Projects must fall under one of the following approved categories:

Project Category	Details / Requirements
A. Research-Oriented Projects	• Must address futuristic domains like Generative AI (e.g., GANs, Transformers, LLMs), cybersecurity, quantum computing, etc. • Multidisciplinary Research Integration: Projects are encouraged to bridge multiple disciplines by combining core technology domains (e.g., AI, cybersecurity, quantum computing) with fields such as Healthcare, Finance, Law & Ethics, Education, Social Sciences, or Environmental Studies. • Require mandatory publication of a research paper in a reputed journal or international conference before final submission. • Supervised by internal faculty mentor
B. Industry-Based Projects	• Should be based on real-world industry challenges with practical applications. • Must demonstrate full implementation, deployment, and usability. • Students will work under external mentorship from an industry along with a faculty mentor from SOET. • External Industry Mentorship Confirmation certificate which will confirms that industry expert/mentor will be guiding the student(s) on their project work. The certificate must be issued and signed by the external mentor. It should be printed on the official letterhead of the company/organization (if available). • A certificate of completion from the associated organization is mandatory.
C. University-Focused Interdisciplinary Projects / Community Solution Based Projects	• Projects under this category should aim to identify and address real-world challenges, enhancement opportunities, or operational gaps within K.R. Mangalam University. • Students are encouraged to collaborate across disciplines, integrating knowledge from diverse domains to develop innovative, practical, and sustainable solutions for campus improvement. • Projects must demonstrate a clear understanding of institutional needs and propose scalable or implementable models that can be piloted or adopted by university stakeholders. • Real-World Problem Orientation (for Community Solution Based Projects) • Projects should address practical issues such as health, sanitation, education, environmental sustainability, digital literacy, social inclusion, transportation, waste management, or livelihood enhancement.
D. Start-Up Projects	• Projects under this category should focus on identifying real-world problems and developing innovative, technology-driven solutions with the potential for commercialization. • Students are encouraged to think entrepreneurially, transforming their ideas into viable products or services by building functional prototypes or Minimum Viable Products (MVPs). • Projects must include key elements of a start-up framework such as

Project Category	Details / Requirements
	market research, user validation, business model design, and go-to-market strategy. • The outcome should demonstrate not only technical feasibility but also market relevance, with the potential to seek incubation or funding support from the university's innovation/start-up ecosystem.
E. Deep-Tech & Systems (Learning-by-Replication)	• This category emphasizes deep technical mastery by replicating foundational technologies that power modern computing. Students build simplified versions of complex systems to understand how they truly work "under the hood," going beyond API usage or high-level application development. • Define a Clear Scope- • The system replica should have: A minimum viable core (e.g., for a compiler: lexer + parser + code-gen) • Extensions or optimizations (indexes, caching, scheduling algorithms) • Examples of "Build Your Own X": • Build your own database engine (query parser, storage engine, index manager) • Build your own compiler (lexer, parser, AST, code generator) • Build your own search engine (crawler, indexer, ranking algorithms) • Build your own machine learning framework (autograd engine, tensor ops) • Build your own virtual machine, interpreter, or runtime environment
F. Hardware & Software Integration Using IoT / Embedded Systems	• This category focuses on building solutions that combine physical hardware (sensors, actuators, microcontrollers) with software systems (apps, cloud platforms, analytics engines). Students must design end-to-end solutions that sense, process, communicate, and act on real-world data. • Projects without actual hardware implementation will not be accepted. A fully functional hardware prototype is mandatory, including working sensors/actuators, embedded Programmeming, connectivity, and a software interface. This ensures the project reflects true hands-on engineering and real-world deployability.

3. Technology Stack and Industry Tools
• Students must select a technology stack aligned with current industry standards, such as MERN, MEAN, Django, Flutter, or cloud-native architectures (AWS, Azure, GCP). • Projects must incorporate industry-relevant tools and frameworks for version control (Git/GitHub), containerization (Docker), CI/CD (GitHub Actions, Jenkins), and agile development practices. • Use of modern development environments, APIs, and deployment platforms is strongly encouraged to ensure scalability, maintainability, and real-world applicability of the solution.

4. Formation of Groups
4.1 Group Size & Composition • Optimal Group Size: Groups will consist of 2 students to allow effective collaboration while maintaining manageable group dynamics. • Skill Diversity: Form groups with a mix of skill sets (e.g., coding, design, research, communication, etc.) to enhance collaboration. Students should be encouraged to select group members based on complementary skills. 4.2 Group Formation Process • Self-selection: Allow students to form their own groups, encouraging them to choose teammates based on project interests. • Pre-formed Groups: Alternatively, groups can be assigned by the coordinator to ensure diversity of skills and ideas across all teams. 4.3 Team Finalization Rule Once teams are formed and registered, no changes in team composition will be allowed under any circumstances.

4.4 Timeline

Week	Activity/Stage
Week 0	Onboarding on Projexa (Project Management Tool) Team formation Faculty Mentor allotment
Week 1–2	Problem Statement Finalization Initial Mentor Interactions and discussion
Week 3–4	• Synopsis Submission • Project Proposal Evaluation and Approval
Week 5,6 and 7	Implementation Sprint 1: Core Modules, MVP Build, Mentor Interaction
Week 8	Development Phase – Part 1 • Module-wise implementation begins
Week 9–10	• Development Phase – Part 2 • Module-wise implementation begins
Week 11–12	Mid Term Evaluation
Week 13–14	• Testing and Debugging • Performance evaluation
Week 15–16	Final Report Submission Project Video (1-2 mins) demonstrating working project Presentation & Viva Preparation

5. Guidelines for Project Selection

For any project category (Research-Based, Development-Based, Start-Up, or University-Focused Interdisciplinary), students may select their project through either of the following approaches:
1. Faculty-Suggested Problem Statements:
• Faculty members will provide curated problem statements based on their domain expertise, current research trends, or institutional needs.
• Students can choose from these problems and discuss with the faculty mentor before finalizing.
2. Student-Proposed Problem Statements:
• Student teams are encouraged to identify real-world problems based on personal interests, industry trends, or societal challenges.
• The proposed problem must be original, relevant, and feasible within the given timeline and resources.
• Final approval is subject to evaluation by the project review committee.

6. Faculty Allocation Process

1. Faculty-Suggested Problem Statement:
If a team selects a problem provided by a faculty member, the team will be allocated to that faculty after peer review and approval by the Project Coordinator.
2. Student-Proposed Problem Statement:
If the team proposes its own problem statement, then after peer review and validation by the Project Coordinator, an appropriate faculty mentor will be assigned based on domain expertise.
3. Inter School Problem Statement:
If a team chooses a problem statement proposed by a faculty member from another school/department, the team will be assigned to that faculty member as their project mentor only after a peer review and approval by the Project Coordinator.

7. Roles and Responsibilities of Faculty Mentors

1. Regular Interactions:Conduct regular meetings with the assigned team(s) through Projexa in online/offline mode. All meeting records, including agenda, minutes, and attendance, must be documented and uploaded on Projexa.
A minimum of five (5) interactions is mandatory, with at least two (2) interactions per month. If the required interactions are not completed within the specified timeline, the pending interaction slots will be automatically closed and marked as incomplete.

2. Task Monitoring:Assign tasks with clear deadlines and track the progress and completion status for each team through the project cycle.

3. Evaluation After Interaction:After every interaction, the faculty mentor must evaluate the team's progress and update the evaluation/comments directly on Projexa as part of the meeting record.

4. Meeting Confirmation & Rescheduling:Faculty must respond to meeting requests initiated by student teams. If unable to conduct a scheduled meeting, it should be rescheduled within one week.

5. Justification for Meeting Cancellation:Any cancellation of scheduled team meetings by the mentor must be supported with a valid and documented reason.

6. GitHub Activity Monitoring:Regularly monitor each team's GitHub repository to verify individual contributions and maintain performance records for all team members.

8. Evaluation Metrics
Projects will be evaluated for a total of 100 marks through two components: 1. Project Mentor Interaction: 25 marks, distributed across synopsis, mid-term and final phases. 2. Project Evaluation Committee Assessment: 75 marks, distributed as Synopsis Presentation (15 marks), Mid-Term Presentation (20 marks), and Final-Term Presentation (40 marks). For each evaluation stage, marks shall be awarded based on project mentor evaluation and project evaluation committee assessment.

Evaluation Stage	Project Mentor Interaction	Project Evaluation Committee
Synopsis Presentation (20)	5	15
Mid-Term Presentation (30)	10	20
Final-Term Presentation (50)	10	40
Total	25	75
Total Marks		(25 + 75) = 100

Project Evaluation Committee Marking Scheme

Evaluation Stage	Criteria	Marks
Synopsis Presentation	Creative/Novel/Real-world Problem	5
	Clarity & Feasibility of Project Objectives	5
	Preparation & Presentation of PPT	5
	Total	15
Mid-Term Presentation	Project Efforts – UI/Implementation	10
	Effective Use of Modern Technology Stack	5
	Effective Coding Practices / Use of Git	5
	Total	20
Final-Term Presentation	Final Production Demonstration & Completion w.r.t Objectives	20
	Impact & Use Cases of the Project	10
	Deployment	10
	Total	40

Evaluation Stage	Criteria	Marks
9. Policy on Missed Presentation Schedules		
Students are strictly required to adhere to the assigned schedules for all project presentations, including the Synopsis, Mid-Term, and Final-Term evaluations. • Only one rescheduling opportunity will be granted in case a team or individual misses their assigned presentation slot. This rescheduling must be approved in advance (where possible) or justified immediately after the missed session with valid reasons. • A penalty of 5 marks will be deducted for each missed presentation schedule, irrespective of the evaluation stage. • Failure to appear even after the rescheduled opportunity will result in zero marks being awarded for that evaluation component. This policy ensures fairness, accountability, and professionalism in the evaluation process.		

10. Project Documentation Requirements – Stage-Wise

Stage	Submission Timing	Required Documents / Notes
1. Synopsis Stage	To be submitted before the Synopsis Presentation:	• Synopsis Report • Synopsis PPT • Mentorship Confirmation Certificate • Must be on industry letterhead, duly signed by the External Mentor
2. Mid-Term Evaluation Stage	To be submitted during the Mid-Term Presentation:	• Mid Evaluation PPT • Project Use Case & Deployment Confirmation Certificate • Issued by the Industry Mentor, confirming project relevance and real-world deployment potential
3. Final-Term Evaluation Stage	To be submitted during or before Final Presentation:	• Final Project Report • PPT for Final Presentation • Video Demonstration (1–2 minutes) • Should clearly document: • Key features of the project • Execution workflow or demonstration of functionality • Real-world impact or use-case • To be submitted in MP4 format or via a shared video link (as per instructions) • Plagiarism Certificate • Must be compulsorily appended to the Final Project Report • Should be obtained using institutional or approved plagiarism checking tools

11. Plagiarism & Ethical Guidelines
• Maximum allowable plagiarism (e.g., ≤ 15%) • Check plagiarism using open-source tools and proper citations • No reuse of previous projects • Intellectual property rights (if applicable) • Compulsory Plagiarism Certificate to be appended in the report file

12. Project Outcome Requirements by each Project Category

Project Category	Desired Outcome
Industry-Based Project	• Project Use Case & Deployment Confirmation Certificate from Industry Mentor • Fully deployed solution (mandatory for final evaluation)
Research-Based Project	• Proof of publication in a reputed conference or journal (UGC CARE / Scopus / IEEE / Springer, etc.)
University-Focused Interdisciplinary Projects / Community Solution Based Projects	• Project Use Case & Deployment Confirmation Certificate from Faculty Mentor • Fully deployed solution (mandatory for final evaluation)
Start-Up Project	• Incubation Letter from recognized incubator/mentor/startup cell • Product delivery and deployment with functional MVP Patent Grant & Publish: The project must result in a patent application, and obtaining a patent grant is compulsory for the project to qualify under this category.
Deep-Tech & Systems (Learning-by-Replication)	Working System Prototype: A functional, simplified replica of a core deep-tech system (e.g., compiler, database, OS module) demonstrating internal mechanisms and correctness.
Hardware & Software Integration Using IoT / Embedded Systems	A fully functional hardware prototype is mandatory, including working sensors/actuators, embedded Programmeming, connectivity, and a software interface.

SUMMER INTERNSHIP-III (ASSESSMENT)

Program Name	B.Tech. CSE Core			
Course Name: Summer Internship-III / Industry Internship Assessment	Course Code	L-T-P	Credits	Contact Hours
	SICSE004	0-0-4	02	—
Type of Course:	INT			
Semester:	VII (Seventh Semester)			
Pre-requisite(s):	Basic technical knowledge in any relevant domain; completion of core engineering courses			

Course Description

The "Industry Internship / Campus Bootcamp Program" course is designed to provide students with intensive industry-oriented practical exposure through either a short-term professional internship or a university-organized campus bootcamp program aligned with current industry requirements. The course focuses completely on hands-on technical learning, skill enhancement, practical implementation and professional readiness.

Students may either:

- Undertake a valid industry internship in a company/startup/research organization, or
- Participate in an approved campus bootcamp program conducted by the university in collaboration with industry experts, trainers or technology partners.

The course emphasizes:

- Practical technical learning
- Industry tool exposure
- Hands-on implementation
- Modern technology workflows
- Collaborative problem solving
- Coding practices and
- Professional development.

The internship/bootcamp shall primarily focus on emerging technologies and industry-relevant domains such as:

- Artificial Intelligence & Machine Learning
- Data Science
- Full Stack Development
- Cloud Computing & DevOps
- Cybersecurity
- UI/UX

- Mobile App Development
- IoT
- Software Development
- Product Engineering and related areas.

By the end of the course, students are expected to demonstrate practical understanding, technical competency, professional discipline and industry readiness through hands-on activities, mentor interactions, assignments and presentations.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply technical skills in practical industry-oriented tasks.	L3 – Apply
CO2	Demonstrate professional communication and teamwork abilities.	L3 – Apply
CO3	Analyze real-world workflows, tools and implementation practices.	L4 – Analyze
CO4	Present technical learning outcomes and professional growth effectively.	L4 – Analyze

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	1	1	2	1	2				
CO2	1	1	2	3	3	2	2				
CO3	3	2	2	2	2	2	3				
CO4	2	2	1	2	3	2	2				

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	2	3	1	1	1
CO2	1	2	2	1	1
CO3	2	2	3	2	1
CO4	1	2	2	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Objectives

The course aims to:

- Provide students with practical industry-oriented learning experiences through internships or bootcamp programs
- Strengthen technical competency and implementation skills using modern tools and technologies
- Enhance teamwork, communication, professionalism and workplace discipline
- Develop exposure to real-world workflows, coding practices and collaborative environments
- Improve industry readiness, self-learning ability and employability skills
- Encourage practical problem-solving and technology-driven learning approaches

Approved Participation Categories

Students may opt for only one of the following approved categories:

Category 1: Industry Internship

- Internship in companies, startups, software firms, technology organizations or research environments
- Students will work on practical technical tasks, software development, analytics, testing, cloud systems, AI/ML applications or related industry activities
- Mandatory submission of internship completion certificate and mentor verification

Category 2: Campus Bootcamp Program

- Intensive skill-based bootcamp conducted by the university or industry partners
- Focused completely on hands-on implementation and practical assignments
- May cover domains such as:
- Data Science
- AI & ML
- Full Stack Development
- DevOps & Cloud
- Cybersecurity
- UI/UX
- Mobile App Development
- Product Development
- Generative AI and related technologies
- Includes coding assignments, mini projects, assessments and technical mentoring sessions

Technology Exposure & Professional Tools

Students are encouraged to gain practical exposure to:

- Git & GitHub workflows
- Cloud platforms such as AWS, Azure or GCP
- Programming tools and IDEs
- AI/ML and Data Science frameworks
- Software development workflows
- Collaboration and productivity tools
- Documentation and presentation platforms
- Industry-standard development and deployment practices

Week	Activity / Stage
Week 0	Registration & Mentor Allocation
Week 1	Orientation & Goal Setting
Week 2–3	Technical Learning & Task Execution
Week 4–5	Hands-on Assignments / Practical Activities
Week 6	Mentor Interaction & Progress Review
Week 7	Advanced Implementation / Documentation
Week 8	Final Submission, Presentation & Viva

Internship / Bootcamp Duration & Timeline

Participation Guidelines

Approval Requirement

- Students must obtain departmental approval before joining any internship or bootcamp program
- The selected domain/program must align with technical learning objectives and specialization areas

Attendance Requirement

- Minimum attendance and active participation are mandatory throughout the course duration
- Attendance records must be maintained and verified by the mentor/coordinator

Mentor Allocation

- Each student shall be assigned a faculty mentor for monitoring and evaluation
- Industry internships shall additionally require an industry mentor/supervisor

Progress Monitoring

- Students must participate in scheduled mentor interactions and progress review sessions
- Faculty mentors shall monitor technical learning, assignments and outcomes

Professional Conduct

Students are expected to:

- Maintain discipline, punctuality and professionalism
- Respect organizational and institutional policies
- Maintain confidentiality wherever applicable
- Demonstrate teamwork, responsibility and ethical behavior

Roles & Responsibilities of Faculty Mentors

Regular Monitoring

- Conduct mentor interaction sessions regularly
- Review student progress and technical learning outcomes
- Guide students in documentation and presentation preparation

Evaluation Support

- Verify assignment submissions and technical activities

- Assess participation, learning outcomes and presentations
- Maintain evaluation records and feedback reports

Evaluation Criteria

Deliverable	Description	Marks
Participation & Attendance	Attendance, discipline and active participation	20
Mentor Interaction & Progress Review	Participation in mentor review sessions and progress updates	40
Technical Assignments / Activities	Completion of practical assignments, coding activities or technical tasks	20
Final Report / Documentation	Internship or bootcamp learning report and activity summary	10
Final Presentation & Viva	Technical presentation and viva on learning outcomes	10
Total		**100**

Rubrics

Assessment Component	Excellent (9–10)	Good (7–8)	Satisfactory (5–6)	Needs Improvement (0–4)
Participation & Attendance	Active participation with excellent discipline and consistency	Regular participation with minor issues	Basic participation with occasional inconsistency	Poor participation or attendance
Mentor Interaction & Progress Review	Excellent progress updates and technical engagement	Good participation and progress reporting	Limited interaction and inconsistent updates	Poor mentor interaction
Technical Assignments / Activities	High-quality technical work with strong implementation	Good practical implementation with minor gaps	Basic completion of assigned tasks	Incomplete or weak implementation
Final Report / Documentation	Professionally documented and comprehensive	Well-organized with minor gaps	Basic documentation submitted	Incomplete or unclear documentation
Final Presentation & Viva	Clear, confident and technically strong presentation	Good presentation and understanding	Basic presentation with limited clarity	Weak presentation and insufficient understanding

Mandatory Submission Requirements

Students must submit:

- Participation/attendance record
- Internship completion certificate or bootcamp completion certificate
- Technical assignment/activity reports
- Final learning report/documentation
- Presentation PPT
- Mentor verification/evaluation (if applicable)

Plagiarism & Ethical Guidelines

- All submitted reports and assignments must be original
- Fabricated internship/bootcamp records shall lead to disciplinary action
- Maximum allowable plagiarism shall follow university norms (≤15%)
- Students must follow professional ethics, discipline and confidentiality practices

Expected Learning Outcomes

By the end of the course, students should be able to:

- Demonstrate practical technical competency in industry-relevant domains
- Apply modern tools and workflows in real-world implementation tasks
- Communicate and collaborate effectively in professional environments
- Develop industry readiness, adaptability and self-learning abilities
- Present technical learning outcomes in a structured and professional manner

Discipline Specific Electives-III						
1	**CSE009**	Parallel & Concurrent Programming	**3**	**0**	**2**	**4**
2	**CSE010**	Natural Language Processing & Generative AI	**3**	**0**	**2**	**4**
3	**CSE011**	Introduction to Blockchain Technologies	**3**	**0**	**2**	**4**
4	**CSE012**	Robotics Programming & ROS	**3**	**0**	**2**	**4**

PARALLEL & CONCURRENT PROGRAMMING

Program Name	B.Tech. CSE Core			
Course Name: Parallel & Concurrent Programming	Course Code	L-T-P	Credits	Contact Hours
	CSE009	3-0-2	04	45
Type of Course:	Discipline Specific Elective (DSE-III)			
Semester:	VII (Seventh Semester)			
Pre-requisite(s):	Basic understanding of programming, operating systems, computer architecture and data structures			

Course Description

This course introduces the principles, models, techniques, and applications of Parallel and Concurrent Programming for designing high-performance, scalable, and reliable software systems. The course focuses on process and thread models, synchronization, shared-memory concurrency, message passing, parallel algorithm design, task decomposition, load balancing, GPU programming concepts, and performance evaluation.

Students will learn how modern computing systems exploit multicore processors, distributed platforms, and accelerator-based architectures to execute multiple tasks efficiently. The course also introduces concurrency hazards such as race conditions, deadlocks, livelocks, starvation, memory consistency issues, and correctness challenges in parallel software development.

The laboratory component emphasizes practical implementation of multithreaded programs, synchronization mechanisms, producer-consumer workflows, parallel algorithms, message-passing programs, and performance benchmarking using modern programming frameworks. By the end of the course, students will be able to design, implement, debug, evaluate, and optimize concurrent and parallel programs for real-world computing problems.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Concise Course Outcome	Bloom's Level
CO1	Explain the fundamentals of parallel computing, concurrency models, threads, processes and performance measures.	L2 – Understand
CO2	Analyze synchronization problems, communication mechanisms, correctness issues and concurrent program behavior.	L4 – Analyze
CO3	Develop parallel and concurrent programs using threads, tasks, shared-memory and message-passing models.	L3 – Apply

CO4	Evaluate parallel programs for scalability, correctness, synchronization overhead, resource utilization and performance.	L5 – Evaluate

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	–	2	–	–	1	1	–	2
CO2	3	3	2	2	3	1	1	2	2	1	2
CO3	3	3	3	2	3	1	1	2	2	1	2
CO4	3	3	3	3	3	2	2	2	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	2	1	–	1
CO2	3	3	3	2	2
CO3	3	3	2	2	3
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	**Fundamentals of Parallel and Concurrent Computing**	**No. of Hours: 12**

- Introduction to Parallel and Concurrent Programming: need, scope, applications and evolution of parallel computing systems
- Concurrency vs Parallelism: task-level concurrency, data parallelism, instruction-level parallelism and execution models
- Computer Architecture for Parallelism: multicore processors, shared memory, distributed memory, cache hierarchy and memory bandwidth
- Process and Thread Models: processes, threads, lightweight processes, context switching and thread lifecycle
- Performance Measures: speedup, efficiency, scalability, throughput, latency and Amdahl's law
- Parallel Program Design: decomposition, dependency analysis, granularity, mapping and load balancing

Unit Number: 2	Synchronization, Communication and Correctness	No. of Hours: 12

- Synchronization Mechanisms: locks, mutexes, semaphores, monitors, condition variables and barriers
- Concurrency Problems: race condition, critical section, deadlock, livelock, starvation and priority inversion
- Classical Synchronization Problems: producer-consumer, readers-writers, dining philosophers and sleeping barber problem
- Memory Consistency and Atomic Operations: memory models, atomic variables, compare-and-swap and happens-before relationship
- Thread Safety and Reentrancy: shared data protection, immutable objects, synchronization overhead and safe APIs
- Debugging Concurrent Programs: nondeterminism, testing strategies, deadlock detection, profiling and correctness validation

Unit Number: 3	Parallel Programming Models and Algorithms	No. of Hours: 11

- Shared-Memory Programming: POSIX threads, OpenMP-style programming model, parallel loops and work-sharing constructs
- Message Passing Programming: distributed-memory concepts, MPI-style communication, point-to-point and collective operations
- Task-Based Parallelism: thread pools, futures, async tasks, fork-join model and work stealing
- Parallel Algorithms: parallel sorting, searching, matrix multiplication, reduction, prefix-sum and graph traversal basics
- Load Balancing and Scheduling: static scheduling, dynamic scheduling, chunking, dependency-aware scheduling and workload distribution
- Performance Optimization: communication overhead, synchronization cost, cache locality, false sharing and parallel bottleneck analysis

Unit Number: 4	Advanced Concurrent Systems and Parallel Applications	No. of Hours: 10

- Lock-Free and Wait-Free Programming: non-blocking data structures, atomic primitives and progress guarantees
- GPU and Accelerator Programming Basics: SIMD/SIMT concepts, CUDA/OpenCL overview, kernels, threads, blocks and memory hierarchy
- Concurrent Data Structures: concurrent queues, stacks, maps, thread-safe collections and producer-consumer pipelines

- Parallel Patterns and Frameworks: map-reduce, pipeline, master-worker, divide-and-conquer and dataflow programming
- Distributed and Cloud-Based Parallelism: cluster computing, distributed tasks, fault tolerance and scalable execution
- Security, Reliability and Ethics: safe parallel software, reproducibility, resource fairness and energy-aware computing
- Case Studies: scientific computing, big data analytics, image processing, simulation, web servers and machine learning workloads

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Parallel Computing Fundamentals and Threading Toolkit	CO1	L2 – Understand
2	Synchronization and Correctness Analysis Framework	CO2	L4 – Analyze
3	Parallel Algorithms and Programming Models	CO3	L3 – Apply
4	Advanced Parallel Applications and Optimization System	CO4	L4 – Analyze
5	Capstone: End-to-End Parallel and Concurrent Programming Project	CO1, CO2, CO3, CO4	L5 – Evaluate

Lab Assignments

Unit 1 Lab Assignment: Parallel Computing Fundamentals and Threading Toolkit	**CO1**

Problem Statement

Design a programming toolkit that demonstrates process creation, thread management, task decomposition and basic performance measurement for concurrent execution.

Sub-Problems

- Implement process and thread creation using a suitable programming language
- Compare sequential and multithreaded execution for simple computational tasks
- Demonstrate task decomposition and workload distribution
- Measure execution time, speedup and efficiency
- Analyze CPU utilization and scalability behavior
- Identify overheads introduced by thread management
- Generate parallel execution and performance reports

Concepts Covered: concurrency, parallelism, threads, task decomposition and performance measures

Unit 2 Lab Assignment: Synchronization and Correctness Analysis Framework	CO2

Problem Statement

Develop concurrent programs that apply synchronization mechanisms and analyze correctness issues such as race conditions, deadlocks and starvation.

Sub-Problems

- Implement critical-section handling using mutex locks or semaphores
- Demonstrate race conditions and correct them using synchronization
- Solve producer-consumer and readers-writers problems
- Implement barrier synchronization and condition variables
- Create and detect a deadlock scenario in a controlled program
- Analyze thread safety and nondeterministic output behavior
- Generate synchronization and correctness evaluation reports

Concepts Covered: synchronization, race condition, deadlock, semaphores, monitors and thread safety

Unit 3 Lab Assignment: Parallel Algorithms and Programming Models	CO3

Problem Statement

Create parallel programs using shared-memory, message-passing or task-based models to solve computational problems efficiently.

Sub-Problems

- Implement parallel matrix multiplication or vector operations
- Develop parallel sorting or searching algorithms
- Implement reduction and prefix-sum operations
- Use thread pools, futures or fork-join style execution
- Compare static and dynamic scheduling strategies
- Analyze communication, synchronization and load-balancing overhead
- Generate algorithmic performance and scalability reports

Concepts Covered: parallel algorithms, shared memory, task parallelism, scheduling and scalability

Unit 4 Lab Assignment: Advanced Parallel Applications and Optimization System	CO4

Problem Statement

Develop an advanced parallel application that applies concurrent data structures, performance optimization and accelerator-aware design principles.

Sub-Problems

- Implement a thread-safe queue, stack or map-based data structure
- Design a producer-consumer pipeline for streaming data processing
- Demonstrate map-reduce or master-worker parallel pattern
- Optimize program performance by reducing synchronization overhead
- Analyze cache locality, false sharing or memory bottlenecks

- Explore GPU or accelerator programming concepts using a simple kernel or simulation
- Generate optimization, reliability and resource-utilization reports

Concepts Covered: concurrent data structures, parallel patterns, optimization, GPU basics and reliability

Capstone Lab Assignment: End-to-End Parallel and Concurrent Programming Project	**CO1, CO2, CO3, CO4**

Problem Statement

Design and implement a complete parallel or concurrent software system that demonstrates decomposition, synchronization, correctness, performance optimization and technical documentation.

Sub-Problems

- Select a computationally intensive or concurrent real-world problem
- Design task decomposition and parallel execution workflow
- Implement concurrency using threads, tasks, message passing or parallel frameworks
- Apply synchronization and correctness validation techniques
- Optimize scalability, memory behavior and execution performance
- Evaluate speedup, efficiency, throughput and reliability
- Prepare visual performance dashboards and comparative analysis
- Prepare technical documentation and project presentation

Concepts Covered: parallel software design, synchronization, algorithm optimization, benchmarking and scalable systems

Text and Reference Books

Textbooks:

1. Peter S. Pacheco, "An Introduction to Parallel Programming," 2nd Edition, Morgan Kaufmann, 2021. ISBN: 978-0128046050
2. Maurice Herlihy and Nir Shavit, "The Art of Multiprocessor Programming," Revised 1st Edition, Morgan Kaufmann, 2012. ISBN: 978-0123973375

Recommended Reference Books:

1. Bryan O'Sullivan, John Goerzen and Don Stewart, "Real World Haskell," 1st Edition, O'Reilly Media, 2008. ISBN: 978-0596514983
2. Clay Breshears, "The Art of Concurrency," 1st Edition, O'Reilly Media, 2009. ISBN: 978-0596521530
3. Anthony Williams, "C++ Concurrency in Action," 2nd Edition, Manning Publications, 2019. ISBN: 978-1617294693
4. Timothy G. Mattson, Beverly A. Sanders and Berna L. Massingill, "Patterns for Parallel Programming," 1st Edition, Addison-Wesley, 2004. ISBN: 978-0321228116

Recommended Online Resources:

- OpenMP Documentation — https://www.openmp.org/resources

- MPI Forum Documentation — https://www.mpi-forum.org/docs
- POSIX Threads Programming Resources — https://computing.llnl.gov/tutorials/pthreads
- NVIDIA CUDA Documentation — https://docs.nvidia.com/cuda
- Intel oneAPI Documentation — https://www.intel.com/content/www/us/en/developer/tools/oneapi/documentation.html
- NPTEL – Parallel Computing — https://nptel.ac.in

NATURAL LANGUAGE PROCESSING & GENERATIVE AI

Program Name	B.Tech. CSE Core			
Course Name: Natural Language Processing & Generative AI	Course Code	L-T-P	Credits	Contact Hours
	CSE010	3-0-2	04	45
Type of Course:	Discipline Specific Elective (DSE III)			
Semester:	VII (Seventh Semester)			
Pre-requisite(s):	Basic knowledge of programming, data structures, probability, machine learning and Python programming.			

Course Description

This course introduces students to Natural Language Processing (NLP) and Generative AI technologies used in modern intelligent systems. The course covers text preprocessing, language modelling, sentiment analysis, transformers, large language models (LLMs), prompt engineering and AI-generated content systems.

Students gain practical exposure to NLP pipelines, deep learning-based language processing, conversational AI and generative AI frameworks using Python and modern AI libraries. The course emphasizes real-world applications such as chatbots, AI assistants, text summarization, retrieval-augmented generation (RAG), content generation and responsible AI practices.

The laboratory component focuses on practical implementation of NLP workflows, transformer models, generative AI systems, conversational agents and deployment-oriented AI applications using modern open-source tools and frameworks.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Concise Course Outcome	Bloom's Level
CO1	Apply NLP techniques for text analysis and language understanding.	L3 – Apply
CO2	Design NLP models for classification and language processing tasks.	L4 – Analyze
CO3	Develop generative AI applications using transformers and LLMs.	L3 – Apply
CO4	Evaluate generative AI systems for performance, ethics and deployment.	L4 – Analyze

CO–PO Articulation Matrix

COs \	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11

POs											
CO1	3	2	1	–	2	–	–	1	1	–	2
CO2	3	3	2	2	3	1	–	1	2	1	2
CO3	3	3	3	2	3	1	1	2	2	1	2
CO4	3	3	3	3	3	1	1	2	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	2	1	–	1
CO2	3	3	3	2	2
CO3	3	3	2	2	3
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Fundamentals of Natural Language Processing	No. of Hours: 12

- Introduction to Natural Language Processing
- Text Preprocessing & Cleaning Techniques
- Tokenization, Stemming & Lemmatization
- Stop-word Removal & Text Normalization
- Part-of-Speech Tagging & Named Entity Recognition
- Syntax & Semantic Analysis Basics
- Vector Space Models & Bag-of-Words

Unit Number: 2	Advanced NLP Applications & Language Models	No. of Hours: 12

- Text Classification & Sentiment Analysis
- Topic Modeling & Document Clustering
- Sequence Models: RNN, LSTM and GRU
- Attention Mechanisms in NLP
- Machine Translation & Text Summarization
- Question Answering Systems & Conversational AI Basics
- Information Retrieval & Recommendation Systems

Unit Number: 3	Fundamentals of Generative AI & Transformer Models	No. of Hours: 11

- Introduction to Generative AI Concepts
- Evolution of Generative Models
- Transformer Architecture Fundamentals
- Large Language Models (LLMs) & Foundation Models
- Prompt Engineering Techniques
- Text Generation & Completion Models

Unit Number: 4	Generative AI Applications, Ethics & Deployment	No. of Hours: 10

- Conversational AI & AI Assistants
- AI-Based Text, Image & Multimedia Generation
- Retrieval-Augmented Generation (RAG) Concepts
- AI Agents & Workflow Automation
- Hallucination, Bias & Explainability in AI Systems
- Responsible AI & Ethical Considerations
- Deployment of NLP & Generative AI Applications

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Text Preprocessing & NLP Pipeline Development	CO1	L3 – Apply
2	NLP-Based Sentiment Analysis & Language Understanding	CO2	L4 – Analyze
3	Generative AI & Transformer-Based Content Generation	CO3	L3 – Apply
4	Conversational AI & Responsible Generative AI Systems	CO4	L4 – Analyze
5	Capstone: Intelligent NLP & Generative AI-Based Virtual Assistant System	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Text Preprocessing & NLP Pipeline Development	CO1

Problem Statement

Develop NLP pipelines for preprocessing and analyzing textual datasets using Python-based NLP libraries and tools.

Sub-Problems

- Perform text cleaning and normalization
- Implement tokenization and stemming techniques
- Apply lemmatization and stop-word removal
- Generate TF-IDF and vector representations
- Build word embedding models
- Analyze text frequency and language patterns
- Generate processed datasets for NLP applications

Concepts Covered: Text preprocessing, tokenization, embeddings and NLP pipelines

Unit 2 Lab Assignment: NLP-Based Sentiment Analysis & Language Understanding	**CO2**

Problem Statement

Build NLP applications for sentiment analysis, text classification and conversational language understanding.

Sub-Problems

- Implement sentiment analysis models
- Build text classification systems
- Develop topic modeling and clustering workflows
- Implement sequence-based NLP models
- Create chatbot or question-answering modules
- Evaluate NLP model accuracy and performance
- Generate analytical reports and visualizations

Concepts Covered: Sentiment analysis, text classification and language modelling

Unit 3 Lab Assignment: Generative AI & Transformer-Based Content Generation	**CO3**

Problem Statement

Develop generative AI applications using transformer models and large language model frameworks for automated content generation.

Sub-Problems

- Implement transformer-based text generation models
- Apply prompt engineering techniques
- Generate AI-based summaries and content
- Fine-tune pretrained generative models
- Develop code or text completion systems
- Experiment with multimodal AI tools
- Evaluate generated outputs using quality metrics

Concepts Covered: Transformers, LLMs, prompt engineering and generative AI

Unit 4 Lab Assignment: Conversational AI & Responsible Generative AI	**CO4**

Systems	

Problem Statement

Design conversational AI systems and evaluate generative AI applications with respect to ethics, security, explainability and deployment considerations.

Sub-Problems

- Build AI chatbot and virtual assistant systems
- Implement retrieval-augmented generation workflows
- Analyze hallucination and bias in generated outputs
- Apply responsible AI and explainability techniques
- Integrate APIs for AI-based automation
- Deploy AI applications for real-world usage
- Generate evaluation and ethical assessment reports

Concepts Covered: Conversational AI, RAG systems and responsible AI

Capstone Lab Assignment: Intelligent NLP & Generative AI-Based Virtual Assistant System	CO1, CO2, CO3, CO4

Problem Statement

Design and develop an intelligent AI-powered virtual assistant integrating NLP pipelines, transformer-based generative models, conversational AI and responsible AI practices for real-world enterprise applications.

Sub-Problems

- Design NLP preprocessing and language understanding modules
- Build conversational AI and chatbot systems
- Integrate transformer-based generative AI models
- Implement prompt engineering and retrieval workflows
- Generate AI-driven summaries, recommendations or reports
- Evaluate AI outputs for bias, security and explainability
- Deploy AI-powered assistant systems using APIs and cloud services
- Generate technical documentation and evaluation reports

Concepts Covered: NLP pipelines, LLMs, conversational AI and responsible AI systems

Text and Reference Books

Textbooks:

1. Daniel Jurafsky & James H. Martin, “Speech and Language Processing,” 3rd Edition, Pearson, 2023.
2. Jacob Eisenstein, “Natural Language Processing,” MIT Press, 2019. ISBN: 978-0262042840

Recommended Reference Books:

1. Denis Rothman, “Transformers for Natural Language Processing and Generative AI,” 3rd Edition, Packt Publishing, 2024. ISBN: 978-1836204510
2. Yoav Goldberg, “Neural Network Methods for Natural Language Processing,” Morgan & Claypool Publishers, 2017. ISBN: 978-1627052986

3. Lewis Tunstall, Leandro von Werra & Thomas Wolf, “Natural Language Processing with Transformers,” O’Reilly Media, 2022. ISBN: 978-1098103248
4. Jay Alammar & Maarten Grootendorst, “Hands-On Large Language Models,” O’Reilly Media, 2024. ISBN: 978-1098150952

Recommended Online Resources:

- Hugging Face Documentation — https://huggingface.co/docs
- OpenAI API Documentation — https://platform.openai.com/docs
- TensorFlow NLP Tutorials — https://www.tensorflow.org/text/tutorials
- PyTorch NLP Resources — https://pytorch.org/tutorials/beginner/nlp/index.html
- Stanford CS224N NLP Course — https://web.stanford.edu/class/cs224n/
- Kaggle NLP Datasets — https://www.kaggle.com/datasets?search=nlp

INTRODUCTION TO BLOCKCHAIN TECHNOLOGIES

Program Name	B.Tech. CSE Core			
Course Name: Introduction to Blockchain Technologies	Course Code	L-T-P	Credits	Contact Hours
	CSE011	3-0-2	04	45
Type of Course:	Discipline Specific Elective (DSE III)			
Semester:	VII (Seventh Semester)			
Pre-requisite(s):	Basic understanding of programming, databases, networking and cryptography fundamentals			

Course Description

This course introduces students to blockchain technologies, decentralized systems, cryptocurrencies, smart contracts and distributed ledger platforms with a strong focus on practical blockchain development and decentralized application design. The course covers blockchain architecture, cryptographic foundations, consensus mechanisms, Ethereum, Hyperledger, decentralized finance (DeFi), Web3 integration and blockchain security.

Students will learn how blockchain systems are designed, validated, secured and deployed for real-world applications in finance, healthcare, supply chain, governance and digital identity systems. The course emphasizes hands-on implementation of smart contracts, decentralized applications (DApps), blockchain testing and enterprise blockchain workflows using modern blockchain development tools and frameworks.

The laboratory component focuses on cryptocurrency transactions, blockchain simulation, smart contract development, decentralized application integration, blockchain security analysis and deployment-oriented blockchain application development.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Concise Course Outcome	Bloom's Level
CO1	Apply blockchain and cryptographic concepts in decentralized systems.	L3 – Apply
CO2	Design blockchain networks and consensus-based systems.	L6 – Create
CO3	Develop smart contracts and decentralized applications.	L6 – Create
CO4	Analyze blockchain scalability, security and enterprise applications.	L4 – Analyse

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	1	1	2	-	1	-	-	-	1
CO2	3	3	3	2	3	1	1	1	-	1	2
CO3	2	2	3	2	3	1	1	2	1	1	2
CO4	3	3	2	3	2	2	2	1	1	-	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	2	1	–	1
CO2	3	3	3	2	2
CO3	3	3	2	2	3
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Fundamentals of Blockchain & Cryptography	No. of Hours: 12

- Introduction to Blockchain Technology: distributed ledgers and decentralization concepts
- Evolution of Blockchain & Cryptocurrencies
- Blockchain Architecture & Transaction Lifecycle
- Blocks, Hashing & Merkle Trees
- Public Key & Private Key Cryptography
- Digital Signatures & Secure Transactions
- Peer-to-Peer Networks & Distributed Systems

Unit Number: 2	Blockchain Networks & Consensus Mechanisms	No. of Hours: 12

- Consensus Mechanisms & Distributed Trust
- Proof of Work (PoW) & Proof of Stake (PoS)
- Delegated Proof of Stake & PBFT Concepts
- Blockchain Mining & Validator Systems
- Ethereum Blockchain Architecture
- Hyperledger Fabric Fundamentals

• Permissioned & Permissionless Blockchains

Unit Number: 3	Smart Contracts & Decentralized Applications	No. of Hours: 11
• Introduction to Smart Contracts • Solidity Programming Fundamentals • Ethereum Virtual Machine (EVM) Architecture • Smart Contract Deployment & Execution • Decentralized Applications (DApps) Architecture • Web3 Integration & Blockchain APIs		

Unit Number: 4	Blockchain Security, Scalability & Applications	No. of Hours: 10
• Blockchain Security Threats & Vulnerabilities • Smart Contract Security & Auditing • Privacy-Preserving Blockchain Technologies • Layer-2 Scaling Solutions & Sidechains • Blockchain Applications in Healthcare, Banking & Governance • NFT Ecosystems & Digital Ownership Concepts • Enterprise Blockchain Solutions & Cloud Integration		

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	Blockchain Fundamentals & Cryptocurrency Transactions	CO1	L3 – Apply
2	Consensus Mechanisms & Blockchain Network Simulation	CO2	L4 – Analyze
3	Smart Contract & DApp Development using Ethereum	CO3	L3 – Apply
4	Blockchain Security & Enterprise Application Development	CO4	L4 – Analyze
5	Capstone: Secure Decentralized Blockchain Application System	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: Blockchain Fundamentals & Cryptocurrency Transactions	CO1
Problem Statement	

Develop blockchain-based transaction systems demonstrating distributed ledger concepts, cryptographic hashing and cryptocurrency operations.

Sub-Problems

- Create and validate blockchain blocks
- Implement hashing and digital signature techniques
- Simulate cryptocurrency transactions
- Generate and manage blockchain wallets
- Explore blockchain transaction explorers
- Implement peer-to-peer transaction verification
- Analyze blockchain immutability and integrity mechanisms

Concepts Covered: Blockchain architecture, hashing, wallets and cryptocurrency transactions

Unit 2 Lab Assignment: Consensus Mechanisms & Blockchain Network Simulation	**CO2**

Problem Statement

Design blockchain network simulations implementing consensus algorithms and distributed validation mechanisms.

Sub-Problems

- Simulate Proof of Work consensus
- Implement Proof of Stake validation models
- Configure private and consortium blockchain networks
- Analyze mining and validator operations
- Measure blockchain scalability and latency
- Implement distributed ledger synchronization
- Generate consensus performance reports

Concepts Covered: Consensus algorithms, blockchain synchronization and network scalability

Unit 3 Lab Assignment: Smart Contract & DApp Development using Ethereum	**CO3**

Problem Statement

Develop smart contracts and decentralized applications using Ethereum blockchain and Solidity programming.

Sub-Problems

- Write Solidity-based smart contracts
- Deploy contracts on Ethereum test networks
- Implement token creation using ERC standards
- Build decentralized application interfaces
- Integrate Web3 APIs with frontend applications
- Test and debug smart contracts
- Generate transaction and deployment reports

Concepts Covered: Smart contracts, Solidity programming and DApp development

Unit 4 Lab Assignment: Blockchain Security & Enterprise Application Development	CO4

Problem Statement

Implement secure blockchain applications and evaluate enterprise blockchain solutions for scalability, privacy and governance.

Sub-Problems

- Analyze smart contract vulnerabilities
- Implement blockchain security auditing techniques
- Design privacy-preserving blockchain solutions
- Explore enterprise blockchain frameworks
- Implement blockchain-based healthcare or finance applications
- Evaluate blockchain scalability and governance models
- Generate enterprise blockchain deployment reports

Concepts Covered: Blockchain security, auditing and enterprise blockchain systems

Capstone Lab Assignment: Secure Decentralized Blockchain Application System	CO1, CO2, CO3, CO4

Problem Statement

Design and develop a secure decentralized blockchain-based application integrating smart contracts, distributed ledger technologies and decentralized applications for real-world applications.

Sub-Problems

- Design blockchain network architecture and workflows
- Implement secure smart contracts using Solidity
- Develop decentralized applications with Web3 integration
- Configure blockchain consensus and validation mechanisms
- Integrate tokenization and digital asset management features
- Apply blockchain security and auditing techniques
- Deploy blockchain applications on test or cloud environments
- Generate technical documentation and deployment reports

Concepts Covered: Blockchain architecture, smart contracts, DApps and blockchain security

Text and Reference Books

Textbooks:

1. Imran Bashir, “Mastering Blockchain,” 4th Edition, Packt Publishing, 2023. ISBN: 978-1803236025
2. Andreas M. Antonopoulos, “Mastering Bitcoin,” 2nd Edition, O’Reilly Media, 2017. ISBN: 978-1491954386

Recommended Reference Books:

1. Andreas M. Antonopoulos & Gavin Wood, “Mastering Ethereum,” O’Reilly Media, 2018. ISBN: 978-1491971949

2. Daniel Drescher, “Blockchain Basics: A Non-Technical Introduction in 25 Steps,” Apress, 2017. ISBN: 978-1484226032
3. Arshdeep Bahga & Vijay Madisetti, “Blockchain Applications: A Hands-On Approach,” VPT Publications, 2017. ISBN: 978-0996025515
4. Narayan Prusty, “Building Blockchain Projects,” Packt Publishing, 2017. ISBN: 978-1787122147

Recommended Online Resources:

- Ethereum Official Documentation — https://ethereum.org/en/developers/docs/
- Hyperledger Fabric Documentation — https://hyperledger-fabric.readthedocs.io/
- Solidity Documentation — https://docs.soliditylang.org/
- Bitcoin Developer Documentation — https://developer.bitcoin.org/
- Web3.js Documentation — https://web3js.readthedocs.io/
- OpenZeppelin Smart Contract Library — https://docs.openzeppelin.com/

ROBOTICS PROGRAMMING & ROS

Program Name	B.Tech. CSE Core			
Course Name: Robotics Programming & ROS	Course Code	L-T-P	Credits	Contact Hours
	CSE012	3-0-2	04	45
Type of Course:	Discipline Specific Elective (DSE III)			
Semester:	VII (Seventh Semester)			
Pre-requisite(s):	Basic understanding of programming, operating systems, sensors and fundamentals of artificial intelligence			

Course Description

This course introduces students to robotics programming and Robot Operating System (ROS) for developing intelligent robotic systems and autonomous applications. The course covers robotic systems, ROS architecture, robot communication, localization, navigation, robotic perception, motion planning and AI-enabled robotics.

Students gain practical exposure to robot simulation, ROS nodes, sensor integration, SLAM, robotic control systems and autonomous robot development using modern ROS2 frameworks and open-source robotic tools. The course emphasizes real-world applications in industrial automation, warehouse robotics, drones, healthcare robotics, autonomous vehicles and smart manufacturing systems.

The laboratory component focuses on hands-on robotic system development using ROS2, Gazebo, RViz, OpenCV and AI-based robotic frameworks. Students will develop practical skills in robot simulation, autonomous navigation, perception systems, robotic communication and intelligent automation aligned with modern robotics industry requirements.

The Course Outcomes (COs)

On completion of the course the participants will learn to:

CO	Concise Course Outcome	Bloom's Level
CO1	Apply ROS tools for robotic systems.	L3 – Apply
CO2	Design robotic navigation and control systems.	L3 – Apply
CO3	Develop autonomous robotic applications.	L3 – Apply
CO4	Analyze intelligent robotic systems and deployments.	L4 – Analyze

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	2	1	3	–	1	1	1	1	2
CO2	3	3	3	2	3	1	1	1	1	1	2
CO3	3	3	3	2	3	1	1	2	2	1	3
CO4	3	3	3	3	3	2	2	2	2	2	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	3	2	1	2
CO2	3	3	3	2	3
CO3	3	3	3	2	3
CO4	2	3	3	3	3

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

Course Outline

Unit Number: 1	Fundamentals of Robotics & ROS2	No. of Hours: 12

- Introduction to Robotics & Autonomous Systems
- Components of Robotic Systems: sensors, actuators, controllers and embedded systems
- Robot Operating System (ROS2) Fundamentals
- ROS2 Architecture: nodes, topics, services, actions and messages
- ROS2 Workspace, Package Management & Communication
- Robot Simulation using Gazebo & RViz
- Robot Kinematics & Motion Basics
- Linux Commands & Robotics Development Environment
- Robot Modeling using URDF Basics
- Practical Activities: ROS2 setup and robotic simulation

Unit Number: 2	Robot Navigation, Localization & Control Systems	No. of Hours: 12

- Coordinate Frames & Transformations
- Robot Localization & Mapping Concepts
- SLAM (Simultaneous Localization and Mapping) Basics
- Path Planning & Obstacle Avoidance
- PID Controllers & Motion Control Systems
- ROS2 Navigation Stack Fundamentals
- Sensor Integration: LiDAR, IMU, ultrasonic sensors and cameras
- Autonomous Navigation & Trajectory Generation
- Mobile Robot Communication & Feedback Systems
- Practical Activities: autonomous robot navigation and sensor fusion

Unit Number: 3	Robotic Perception, AI & Autonomous Systems	No. of Hours: 11

- Computer Vision for Robotics using OpenCV
- Object Detection & Recognition in Robotics
- AI & Machine Learning in Robotics
- ROS2-Based Robotic Manipulation
- Pick-and-Place Automation Workflows
- Human-Robot Interaction Fundamentals
- Multi-Robot Communication & Coordination
- Drone & Autonomous Aerial Robotics Basics
- Edge AI & Intelligent Robotics Systems
- Practical Activities: robotic perception and AI-driven automation

Unit Number: 4	Industrial Robotics, Smart Automation & Modern Trends	No. of Hours: 10

- Industrial Automation & Collaborative Robots (Cobots)
- Warehouse Robotics & Autonomous Mobile Robots (AMRs)
- Healthcare & Medical Robotics Applications
- IoT Integration & Smart Robotics Systems
- Cloud Robotics & Remote Monitoring Basics
- Robotic System Testing & Debugging
- Safety, Ethics & Security in Robotics
- Real-Time Robotic Deployment Considerations
- Future Trends: humanoid robots, digital twins and AI robotics
- Mini Project Development: intelligent robotic monitoring or autonomous robotic system

Lab Experiments – CO Mapping Table

S. N	Experiment Title	Mapped CO(s)	Bloom's Level
1	ROS2 Environment Setup & Robotic Simulation	CO1	L3 – Apply
2	Robot Navigation & Sensor Integration System	CO2	L3 – Apply
3	Autonomous Robotics & Perception Application	CO3	L3 – Apply
4	Intelligent Robotic Automation & Monitoring Platform	CO4	L4 – Analyze
5	Capstone: Autonomous ROS2-Based Intelligent Robotic System	CO1, CO2, CO3, CO4	L4 – Analyze

Lab Assignments

Unit 1 Lab Assignment: ROS2 Environment Setup & Robotic Simulation	**CO1**

Problem Statement

Develop robotic simulation environments using ROS2 tools, robot models and communication frameworks for autonomous robotic systems.

Sub-Problems

- Install and configure ROS2 environments
- Create ROS2 workspaces and packages
- Implement ROS2 publishers and subscribers
- Simulate robots using Gazebo and RViz
- Configure robotic sensors and actuators
- Perform robot movement and motion control
- Generate simulation and communication reports

Concepts Covered: ROS2 architecture, robotic simulation and communication systems

Unit 2 Lab Assignment: Robot Navigation & Sensor Integration System	**CO2**

Problem Statement

Design robotic navigation systems integrating localization, mapping and autonomous movement using ROS2 frameworks.

Sub-Problems

- Configure coordinate frames and transformations

- Integrate LiDAR, IMU and camera sensors
- Implement SLAM and localization techniques
- Perform obstacle detection and avoidance
- Configure navigation stack and path planning
- Implement PID-based motion control systems
- Analyze navigation and trajectory performance

Concepts Covered: SLAM, localization, path planning and robot navigation

Unit 3 Lab Assignment: Autonomous Robotics & Perception Application	**CO3**

Problem Statement

Build intelligent robotic systems integrating perception, AI-based control and robotic automation techniques.

Sub-Problems

- Implement object detection using OpenCV
- Integrate AI models with robotic systems
- Configure robotic arm manipulation tasks
- Develop pick-and-place automation workflows
- Implement multi-robot communication systems
- Build drone or aerial robotic simulations
- Generate robotic perception and automation reports

Concepts Covered: Robotic perception, AI integration and autonomous robotics

Unit 4 Lab Assignment: Intelligent Robotic Automation & Monitoring Platform	**CO4**

Problem Statement

Develop and evaluate intelligent robotic applications for industrial automation and smart robotic monitoring systems.

Sub-Problems

- Implement robotic automation workflows
- Configure robotic monitoring and debugging tools
- Integrate IoT-enabled robotic communication systems
- Analyze robotic safety and system performance
- Deploy autonomous robotic applications in simulation
- Evaluate scalability and real-time robotic behavior
- Generate deployment and evaluation reports

Concepts Covered: Industrial robotics, robotic monitoring and intelligent automation

Capstone Lab Assignment: Autonomous ROS2-Based Intelligent Robotic System	CO1, CO2, CO3, CO4

Problem Statement

Design and develop a complete autonomous robotic system integrating ROS2 communication, robotic perception, localization and AI-driven automation for real-world applications.

Sub-Problems

- Design robotic architecture and communication workflows
- Develop ROS2-based robot control systems
- Integrate sensors, localization and navigation modules
- Implement AI-based robotic perception systems
- Develop autonomous robotic movement tasks
- Integrate IoT-based robotic monitoring features
- Evaluate robotic safety, scalability and efficiency
- Deploy robotic systems in simulation environments
- Generate technical documentation and performance reports
- Present a complete intelligent robotic solution

Concepts Covered: ROS2, autonomous robotics, AI perception and intelligent automation

Text and Reference Books

Textbooks:

1. Morgan Quigley, Brian Gerkey & William D. Smart, "Programming Robots with ROS," O'Reilly Media, 2015. ISBN: 9781449323899
2. Aaron Martinez & Enrique Fernandez, "Learning ROS for Robotics Programming," 2nd Edition, Packt Publishing, 2018. ISBN: 9781788623315
3. John J. Craig, "Introduction to Robotics: Mechanics and Control," 4th Edition, Pearson, 2018. ISBN: 9780133489798

Recommended Reference Books:

1. Sebastian Thrun, Wolfram Burgard & Dieter Fox, "Probabilistic Robotics," MIT Press, 2005. ISBN: 9780262201629
2. Lentin Joseph, "Mastering ROS for Robotics Programming," 3rd Edition, Packt Publishing, 2022. ISBN: 9781801071024
3. Bruno Siciliano & Oussama Khatib, "Springer Handbook of Robotics," 2nd Edition, Springer, 2016. ISBN: 9783319325507
4. Francis X. Govers, "Artificial Intelligence for Robotics," Packt Publishing, 2018. ISBN: 9781788835442

Recommended Online Resources:

- ROS2 Official Documentation — https://docs.ros.org/en/rolling/index.html
- Gazebo Simulator Documentation — https://gazebosim.org/docs

- Open Robotics Learning Resources — https://www.openrobotics.org/
- NVIDIA Isaac Robotics Platform — https://developer.nvidia.com/isaac
- OpenCV Documentation — https://docs.opencv.org/
- Coursera Robotics Courses — https://www.coursera.org/courses?query=robotics
- edX Robotics Learning Resources — https://www.edx.org/learn/robotics
- GitHub ROS Open Source Projects — https://github.com/topics/ros2
- GeeksforGeeks ROS Tutorials — https://www.geeksforgeeks.org/robot-operating-system-ros-introduction/

SEMESTER: VIII

CAPSTONE PROJECT – II (INDUSTRY INTEGRATED MAJOR PROJECT)

Program Name	B.Tech. CSE Core			
Course Name: Capstone Project – II	Course Code	L-T-P	Credits	Contact Hours
	PRCSE005	0-0-24	12	—
Type of Course:	PROJ			
Semester:	VIII Semester			
Pre-requisite(s):	Capstone Project – I; Core Engineering and Domain-specific courses			

Course Description

Capstone Project – II is a major industry-integrated project course designed to provide students with intensive exposure to solving real-world industrial problems through end-to-end development, deployment and validation of technology solutions. The course emphasizes professional engineering practices, industry collaboration, scalable deployment, research orientation, innovation and production-ready implementation.

Unlike exploratory or academic mini-projects, this course strictly focuses on industry-specific problem statements sourced from companies, startups, industries, research labs, government organizations or professional mentors. Students are expected to work on practical business, engineering, automation, AI/ML, cybersecurity, cloud, data engineering, embedded systems or product-development challenges directly relevant to industrial environments.

The course follows a complete software/product engineering lifecycle including:

- Problem identification
- Requirement analysis
- Architecture design
- Implementation
- Deployment
- Testing
- Validation
- Optimization
- Documentation and
- Professional presentation.

Students will work under joint mentorship of:

- An Industry Mentor (mandatory), and
- A Faculty Mentor.

The project outcome must demonstrate:

- Deployable implementation
- Measurable impact
- Production readiness

- Scalability and
- Real-world usability.

The course also emphasizes:

- Collaborative development
- GitHub-based project management
- Agile workflows
- DevOps practices
- Technical documentation
- Client communication and
- Professional ethics.

By the end of the course, students must demonstrate a complete working solution aligned with real industrial use-cases and industry standards.

The Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Analyze and define an industry-specific real-world problem.	L4 – Analyze
CO2	Design and implement a deployable industry-grade solution.	L3 – Apply
CO3	Evaluate scalability, performance and usability of the developed system.	L5 – Evaluate
CO4	Demonstrate professional project execution, documentation and deployment practices.	L3 – Apply

CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	2	1	1	2	2	1	2	2	2
CO2	3	3	2	1	2	3	2	2	2	2	2
CO3	3	3	1	1	2	2	2	1	2	2	3
CO4	2	2	1	2	3	3	2	3	3	3	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	3	3	2	1

CO2	3	3	2	2	3
CO3	2	3	3	3	2
CO4	2	2	1	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

1. Objectives

All projects under Capstone Project – II must:

- Address a real-world industry problem provided by a company, startup, research lab, government organization or industrial mentor
- Demonstrate complete implementation, deployment and validation of the proposed solution
- Use industry-standard tools, frameworks and engineering workflows
- Follow professional software/product development lifecycle practices
- Deliver measurable outcomes such as deployment, automation, analytics, optimization or operational improvements
- Include proper documentation, deployment workflows and scalability considerations
- Encourage innovation, research orientation and product thinking

2. Mandatory Industry Integration Requirement

- Every project must be strictly mapped to a real industry problem statement

The problem statement must be approved by:

- Industry Mentor
- Faculty Mentor
- Project Review Committee

Each team must have:

- One external industry mentor (mandatory)
- One faculty mentor

The industry mentor must provide:

- Mentorship confirmation certificate
- Deployment/use-case validation certificate
- Completion certificate at the end of the project
- Projects without valid industry mapping and industry mentorship will not be approved under this course

3. Project Categories (Industry-Focused)

Projects may belong to the following domains only if mapped to real industrial use-cases:

A. AI/ML & Data Science Solutions

- Predictive analytics systems
- Recommendation engines
- AI chatbots
- Fraud detection systems
- Generative AI solutions

B. Cloud & DevOps Automation

- CI/CD automation
- Cloud deployment systems
- Infrastructure monitoring platforms
- Kubernetes-based deployment systems

C. Cybersecurity Solutions

- Threat detection systems
- Secure authentication systems
- Vulnerability analysis tools
- SIEM dashboards

D. Full Stack & Enterprise Applications

- ERP modules
- CRM systems
- SaaS platforms
- Enterprise workflow automation

E. IoT & Embedded Systems

- Smart monitoring systems
- Industrial automation solutions
- Sensor-based analytics systems
- Smart agriculture systems

F. Deep-Tech & Systems Engineering

- Database engines
- Search engines
- Distributed systems
- Compiler/interpreter systems

G. Research & Innovation Projects

- Research prototypes
- Experimental AI systems
- Emerging technology applications
- Interdisciplinary engineering systems

4. Technology Stack & Engineering Practices

Projects must incorporate modern engineering workflows including:

- Git & GitHub for version control and collaboration
- Cloud deployment platforms (AWS, Azure, GCP, Render, Vercel, etc.)
- Containerization using Docker
- CI/CD workflows using GitHub Actions or Jenkins
- API-driven architecture and scalable backend workflows
- Database systems and monitoring tools

- Agile development practices and sprint planning
- Technical documentation and issue tracking systems

5. Group Formation

5.1 Group Size

- 2–4 students per team depending on project complexity

5.2 Team Composition

- Teams should contain complementary technical skill sets

5.3 Team Finalization

- No team modification will be allowed after registration approval

6. Project Lifecycle & Timeline

Week	Activity
Week 0	Team formation, onboarding and mentor allocation
Week 1–2	Industry problem statement finalization
Week 3–4	Requirement analysis and architecture planning
Week 5–7	Sprint 1: MVP implementation
Week 8–10	Sprint 2: Core implementation & deployment
Week 11–12	Mid-term evaluation
Week 13–14	Testing, optimization and scalability analysis
Week 15	Deployment validation and documentation
Week 16	Final presentation, viva and industry demonstration

7. Mandatory Deliverables

Synopsis Stage

- Project Synopsis Report
- Synopsis PPT
- Industry Mentorship Confirmation Certificate
- Requirement Analysis Document

Mid-Term Stage

- Working MVP Demonstration
- Mid-Term PPT
- GitHub Repository Review
- Deployment Progress Report
- Industry Use-Case Validation Certificate

Final Stage

- Final Project Report
- Final PPT

- Complete Source Code Repository
- Deployment Link / Product Demonstration
- 1–2 Minute Product Demo Video
- Industry Completion Certificate
- Plagiarism Report
- User Manual & Technical Documentation

8. Evaluation Scheme (Total: 200 Marks)

Evaluation Component	Marks
Synopsis Evaluation	30
Mid-Term Evaluation	50
Final Demonstration & Deployment	80
Industry Mentor Evaluation	20
Faculty Mentor Evaluation	20
Total	**200**

9. Final Evaluation Rubrics

Criteria	Marks
Industry Problem Relevance	10
Innovation & Technical Complexity	15
Implementation Quality	15
Deployment & Scalability	15
UI/UX & Usability	10
Industry Impact & Real-World Applicability	10
GitHub & Engineering Practices	5
Documentation & Reporting	5
Presentation & Viva	5
Total	**90**

10. Mandatory Industry Outcome Requirements

Every project must demonstrate at least one of the following:

- Production deployment
- Industry pilot testing
- Usability validation from industry mentor
- Research publication
- Start-up incubation support
- Patent filing

- Client-side implementation or proof-of-concept validation

11. GitHub & Development Workflow Requirements

All projects must maintain:

- Private/public GitHub repository
- Commit history
- Issue tracking
- Sprint updates
- Documentation

Faculty mentors will regularly monitor:

- Commits
- Pull requests
- Contribution patterns
- Implementation progress

12. Plagiarism & Ethical Guidelines

- Maximum plagiarism allowed: ≤ 15%
- Use of copied source code without citation is strictly prohibited
- Use of AI-generated code/tools must be properly documented and justified
- All datasets, APIs and third-party resources must be properly cited
- Plagiarism certificate is mandatory during final submission

13. Expected Final Outcomes

The project must result in:

- A fully functional deployed system/product
- Industry validation or deployment certificate
- Professional technical documentation
- GitHub-based collaborative engineering workflow
- Demonstration of real-world impact and scalability
- Industry-grade engineering practices and deployment standards

K.R. MANGALAM UNIVERSITY

Standard Operating Procedure (Sop)

INDUSTRY INTERNSHIP / INDUSTRIAL TRAINING

Parameter	Details
Applicable Programs	B.Tech. (VIII Semester)
Course Type	Internship / Industrial Training
Course Code	PRCSE005
Credits	12
L:T:P	0:0:24
Duration	Minimum 5–6 Months
Mode	Full-Time Industry Internship
Evaluation Type	Continuous Evaluation + Final Assessment
Nature of Course	Industry Immersion & Professional Skill Development

1. Course Perspective

The "Industry Internship / Industrial Training" course is designed to provide students with extensive real-world industrial exposure through full-time professional internships in reputed companies, startups, industries, research organizations or technology-driven enterprises.

The course aims to bridge the gap between academic learning and industry expectations by enabling students to work on real projects, development workflows, business problems and engineering tasks under professional supervision. Students are expected to apply technical knowledge, problem-solving skills, communication abilities and engineering practices in actual workplace environments.

The internship shall primarily focus on:

- Practical implementation
- Industry exposure
- Professional conduct
- Workplace collaboration
- Product development workflows
- Software engineering practices
- Business communication and
- Organizational learning.

Students are strongly encouraged to secure internships on a paid basis to ensure professional accountability, industry engagement and meaningful work exposure.

The internship experience should contribute towards:

- Employability enhancement
- Industry readiness
- Technical competency
- Professional networking and
- Long-term career growth.

2. Objectives

The internship course aims to:

- Provide practical industry exposure through long-duration professional internships
- Enable students to apply academic concepts to real-world industrial problems
- Develop professional skills including communication, teamwork and workplace ethics
- Expose students to industrial workflows, tools, technologies and project management practices
- Enhance employability, technical competency and career readiness
- Encourage students to contribute to real products, services or organizational processes

3. Course Outcomes (COs)

On completion of the course, the participants will be able to:

CO	Course Outcome	Bloom's Level
CO1	Apply technical knowledge to real industrial tasks and projects.	L3 – Apply
CO2	Analyze industry workflows, tools and professional engineering practices.	L4 – Analyze
CO3	Evaluate workplace challenges and develop effective technical solutions.	L5 – Evaluate
CO4	Demonstrate professional communication, teamwork and ethical conduct.	L3 – Apply

5. CO–PO Articulation Matrix

COs \ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	1	1	2	2	2	2	2	2	2
CO2	3	3	1	1	2	2	3	2	2	3	3
CO3	3	3	2	2	2	3	3	2	2	3	3
CO4	2	2	2	3	3	2	3	3	3	2	2

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

5. CO–PSO Articulation Matrix

COs \ PSOs	PSO1 Understanding CS Concepts & Methodologies	PSO2 Applying CS Knowledge to Real-World Challenges	PSO3 Analyzing CS Problems & Situations	PSO4 Evaluating Alternative Solutions & Decisions	PSO5 Designing & Developing Innovative Solutions
CO1	3	3	2	2	2
CO2	3	3	3	2	2
CO3	2	3	3	3	2

CO4	2	2	2	2	1

Scale: 3 = High correlation, 2 = Moderate correlation, 1 = Low correlation, – = No correlation

6. Internship Requirements

6.1 Mandatory Internship Duration

- Minimum internship duration shall be 5–6 months full-time

The internship must be completed during:

- VIII Semester for B.Tech students
- VI Semester for BCA students

6.2 Internship Type

Students may pursue internships in:

- Product-based companies
- Service-based companies
- Startups
- Research organizations
- Government organizations
- MSMEs and technology firms
- AI/ML, Cloud, Cybersecurity, Data Science, Software Development, IoT or related domains

6.3 Paid Internship Preference

- Students are strongly encouraged to secure internships on a paid/stipend basis
- Paid internships shall be given preference during approval and evaluation processes

Unpaid internships may be approved only if:

- The company is technically relevant, and
- The work profile is meaningful and industry-oriented

6.4 Work Nature

The internship must involve:

- Real project contribution
- Technical development
- Engineering workflows
- Deployment/support activities
- Automation
- Analytics or
- Software/product engineering tasks.

Purely non-technical internships shall not be approved.

7. Internship Approval Process

Stage 1: Internship Identification

Students must:

- Apply to companies independently or through university placement support
- Identify suitable internship opportunities relevant to their specialization/domain

Stage 2: Internship Approval

Students must submit:

- Offer Letter / Internship Confirmation Letter
- Company Profile
- Job Description / Role Description
- Internship Duration Details
- Industry Mentor Details

Approval shall be granted by:

- Department Internship Coordinator
- HoD/Dean (where applicable)

Stage 3: Faculty Mentor Allocation

Each student/team shall be assigned one faculty mentor from the department. The faculty mentor shall:

- Monitor progress
- Conduct review meetings and
- Evaluate internship learning outcomes

8. Roles & Responsibilities

8.1 Student Responsibilities

Students must:

- Maintain professional conduct throughout the internship period
- Actively contribute to assigned tasks/projects
- Attend regular review meetings with faculty mentor
- Maintain internship logs and progress records
- Submit reports and required documentation on time
- Follow company policies, confidentiality and ethical practices

8.2 Industry Mentor Responsibilities

Industry mentors shall:

- Assign technical tasks/projects to interns
- Provide technical guidance and workplace mentorship
- Evaluate student performance and contribution
- Provide internship completion certificate and evaluation feedback

8.3 Faculty Mentor Responsibilities

Faculty mentors shall:

- Monitor internship progress regularly
- Conduct periodic review interactions
- Verify learning outcomes and technical involvement
- Guide students in documentation and reporting
- Evaluate presentations and final reports

9. Internship Monitoring & Review Process

Stage	Activity
Week 1–2	Internship registration & approval
Week 3–6	Initial review & task verification
Week 7–10	Mid-term progress review
Week 11–16	Performance monitoring & mentor interactions
Week 17–20	Final report preparation
Week 21–24	Final presentation & viva

10. Mandatory Deliverables

Initial Submission

- Internship offer letter
- Internship role description
- Company profile
- Internship approval form

Mid-Term Submission

- Mid-term internship report
- Task/activity summary
- Faculty mentor review record
- Industry mentor progress feedback

Final Submission

- Final Internship Report
- Internship Completion Certificate
- Industry Mentor Evaluation Form
- Monthly Work Logs / Activity Records
- Presentation PPT
- Internship Experience Video (2–3 minutes)
- Offer/Conversion Letter (if applicable)

11. Internship Report Guidelines

The final report should include:

- Company overview
- Internship objectives
- Technologies/tools used
- Tasks and responsibilities
- Project contribution details
- Learning outcomes
- Challenges faced and solutions implemented
- Professional skills gained
- Conclusion and future scope

12. Evaluation Scheme (Total: 200 Marks)

Evaluation Component	Marks
Internship Registration & Initial Documentation	20
Mid-Term Evaluation	40
Faculty Mentor Evaluation	30
Industry Mentor Evaluation	50
Final Internship Report	20
Final Presentation & Viva	40
Total	**200**

13. Industry Mentor Evaluation Rubrics (50 Marks)

Criteria	Marks
Technical Contribution	15
Problem-Solving Ability	10
Professional Conduct	10
Communication & Teamwork	5
Learning Attitude & Adaptability	5
Attendance & Discipline	5
Total	**50**

14. Final Presentation Evaluation Rubrics

Criteria	Marks
Technical Understanding	10
Internship Contribution	10
Practical Learning Outcomes	5
Presentation Skills	5
Problem-Solving & Innovation	5
Documentation Quality	5
Total	**40**

15. Internship Completion Criteria

To successfully complete the course, students must:

- Complete minimum internship duration requirements
- Obtain internship completion certificate from company
- Receive satisfactory evaluation from industry mentor

- Submit all reports and documents
- Successfully complete final presentation and viva

16. Policy Guidelines

- Fake internships or fabricated certificates shall lead to disciplinary action
- Students leaving internship midway without approval may be declared unsuccessful in the course
- Internship domain must align with the student's program specialization
- Internships may be conducted in hybrid/remote/on-site mode with approval
- Students are encouraged to secure Pre-Placement Offers (PPOs) through internship performance

17. Suggested Internship Domains

- Software Development
- AI & Machine Learning
- Data Science & Analytics
- Cloud Computing & DevOps
- Cybersecurity
- Full Stack Development
- UI/UX Engineering
- IoT & Embedded Systems
- Blockchain Applications
- Computer Vision & NLP Applications

18. Expected Outcomes

By the end of the internship, students should be able to:

- Work professionally in industry environments
- Contribute to real engineering or business problems
- Use modern tools, frameworks and workflows
- Demonstrate technical and professional competency
- Improve employability and placement readiness