SCHOOL OF ENGINEERING AND TECHNOLOGY

# B. Tech (Computer Science & Engineering)

**Programme Code: 01**

**FOUR YEAR UNDERGRADUATE PROGRAMME**

**(with effect from 2024-25 session)**

**Approved in the 34th Meeting of Academic Council Held on 29 June 2024**

# Table of Contents

## Contents

# Preamble

Welcome to the School of Engineering and Technology at K. R. Mangalam University. It is with great enthusiasm that we introduce you to an institution dedicated to nurturing future leaders in engineering and technology.

Established in 2013, our School has rapidly evolved into a premier center for innovation, quality education, and skill development. With a focus on imparting advanced knowledge and fostering creativity, we are committed to providing a transformative educational experience. Our state-of-the-art infrastructure, cutting-edge laboratories, and a distinguished team of faculty members collectively create an environment where academic and professional excellence thrives.

Our diverse programs encompass undergraduate degrees (B.Tech, BCA, B.Sc), postgraduate studies (M.Tech, MCA), and doctoral research across all engineering disciplines. Notably, we offer specialized B.Tech programs in areas such as Artificial Intelligence & Machine Learning, Data Science, Cyber Security, Full Stack Development, and UI/UX Development. These programs are designed to equip students with both technical proficiency and a deep understanding of emerging technologies.

At the heart of our mission is a commitment to a curriculum that integrates the best practices from leading global institutions while also incorporating insights from the Open-Source Society University.

Our emphasis on industry integration is reflected in our collaborations with renowned organizations such as IBM, Samatrix, Xebia, E.C Council, and ImaginXP. These partnerships ensure that our students gain practical experience and insights that are directly applicable to industry demands. Elective options across diverse domains, including AI, Cloud Computing, Cyber Security, and Full Stack Development, offer students the flexibility to tailor their educational experience to their career aspirations.

We are also dedicated to fostering a culture of innovation and entrepreneurship through our Entrepreneurship and Incubation Center and initiatives like 'MindBenders,' 'Hack-KRMU,' and participation in the 'Smart India Hackathon.' These programs are designed to inspire and prepare students to become forward-thinking leaders in the technology sector.

Our modern computing facilities and comprehensive infrastructure support advanced research, simulations, and hands-on projects, ensuring that our students are well-prepared for the challenges of the professional world. K. R. Mangalam University is recognized for its commitment to providing quality education, and our alumni have made notable contributions across various sectors, from multinational corporations to public sector enterprises.

We are excited to accompany you on this journey and look forward to supporting your academic and professional growth. Welcome to a community where excellence and innovation are at the core of everything we do.

**School of Engineering & Technology**
**K.R Mangalam University**

## Categories of Courses

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

**Minor**: Students will have the option to choose courses from disciplinary/interdisciplinary minors and skill-based courses. Students who take a sufficient number of courses in a discipline or an interdisciplinary area of study other than the chosen major will qualify for a minor in that discipline or in the chosen interdisciplinary area of study.

**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):** Students are required to achieve competency in a Modern Indian Language (MIL) and in the English language with special emphasis on language and communication skills. The courses aim at enabling the students to acquire and demonstrate the core linguistic skills, including critical reading and expository and academic writing skills, that help students articulate their arguments and present their thinking clearly and coherently and recognize the importance of language as a mediator of knowledge and identity.

**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 (Honours with Research) are required to take up research projects under the guidance of a faculty member. The students are expected to complete the Research Project in the final semester. The research outcomes of their project work may be published in peer-reviewed journals or may be presented in conferences /seminars or may be patented.

## University Vision and Mission

**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.

**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

## About the School

The School of Engineering and Technology at K. R. Mangalam University started in 2013 to create a niche of imparting quality education, innovation, entrepreneurship, skill development and creativity. It has excellent infrastructure, state of the art Labs, and a team of qualified and research-oriented faculty members.

The school is offering undergraduate programs (B.Tech, BCA, B.Sc), postgraduate programs (M.Tech, MCA) and Ph.D (all disciplines of Engineering). We are offering B.Tech programs in recent areas of specializations like AI & ML, Data Science, Cyber Security, Full stack development, UI/UX development etc.

Our strength lies in our highly qualified, research oriented, and committed teaching faculty. We believe in empowering minds through expert guidance, ensuring that our students receive a world-class education that prepares them for the challenges of the ever-evolving technological landscape.

The School of Engineering & Technology is committed to providing a cutting-edge curriculum by integrating the best practices from top global universities and leveraging the rich knowledge resources of the Open-Source Society University. The curriculum focuses on problem-solving, design, development, interdisciplinary learning, skill development, research opportunities and application of various emerging technologies with focus on innovative teaching learning methodologies. Aligned with the National Education Policy (NEP) 2020, our curriculum is designed to provide a holistic and contemporary learning experience.

We take pride in offering an industry-integrated curriculum that goes beyond traditional education. Collaborations and training led by industry experts, along with partnerships with renowned organizations such as IBM, Samatrix, Xebia, E.C Council, ImaginXP etc ensure that our students gain practical insights and skills that align with real-world industry demands.

With elective options across various domains, including AI, Cloud Computing, Cyber Security, and Full Stack Development, we empower students to customize their learning experience. Our goal is to provide the flexibility needed for each student to shape their academic and professional future.

We prioritize career growth by offering comprehensive training, placements, international internships, and preparation for further studies. Our commitment to nurturing globally competitive professionals is reflected in the diverse pathways we pave for our students.

SOET aims at transforming the students into competitive engineers with adequate analytical skills, making them more acceptable to potential employers in the country. At our school, we emphasize learning through doing. Whether it's project-based learning, field projects, research projects, internships, or engaging in competitive coding, our students actively shape their futures by applying theoretical knowledge to practical scenarios. We provide opportunities for industrial projects, R&D projects, and start-up projects in the final year, ensuring that our students engage in real-world innovation.

We are dedicated to fostering a culture of innovation and entrepreneurship, recognizing these as essential pillars for the success of our students in the rapidly evolving world of technology. We inspire innovation and entrepreneurship through our dynamic Entrepreneurship and Incubation Center, engaging contests like 'MindBenders' ,'Hack-KRMU,' participation in ‘Smart India Hackathon’, International Conference ‘MRIE’ empowering students to become forward-thinking leaders in the ever-evolving realm of technology.

We pride ourselves on providing state-of-the-art computing facilities and infrastructure. Our modern labs and computing resources are equipped to support the diverse needs of our students, enabling them to engage in advanced research, simulations, and hands-on projects.

K.R. Mangalam University has marked its presence in Delhi NCR as a value-based university, successfully imparting quality education in all domains. Our alumni are working across all sectors of technology, from MNCs to PSUs.

## School Vision and Mission

**Vision**

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

**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.

## About the Program

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 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.

- **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 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.

- **Credit**

  Credit refers to a unit by which the course work is measured. It determines the number of hours of instructions required per week. One credit is equivalent to 14-15 periods for theory, or 28-30 periods for workshop/labs and tutorials

## 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.

## Programme Outcomes (PO)

**Engineering Graduates will be able to:**

**PO1. Core Competencies in Engineering:** Graduates will possess a strong foundation in engineering knowledge, critical problem analysis, and solution design, equipped with skills for conducting thorough investigations to solve complex challenges.

**PO2. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO3. Societal and Environmental Responsibility:** Apply contextual knowledge to evaluate societal, health, safety, legal, and cultural issues, while understanding the impact of engineering solutions on the environment and advocating for sustainable development.

**PO4. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO5. Effective Communication and Team Collaboration:** Excel in both individual and team roles within diverse and multidisciplinary settings, while communicating complex engineering concepts clearly through effective reports, presentations, and interactions.

**PO6. Project management:** Apply engineering and management principles to lead and manage projects effectively in computer science and engineering contexts.

**PO7. Life-long learning:** Embrace and actively pursue continuous learning to stay current with technological advancements and evolving practices in computer science and engineering.

## 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

## 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.

## Duration

4 Years (8 Semesters) - Full-Time Program

## Eligibility Criteria for Award of Degree

Students must successfully complete the minimum required credits of 168 to be eligible for award of degree.

# Student's Structured Learning Experience in the Programme

**a. 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.

**b. 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 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 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.

## *c.* 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, aligned with the National Education Policy (NEP) 2020. 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 program, 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.

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.

**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 (C++), 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.

**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.

**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 Counselling and Entrepreneurship

The university offers comprehensive **career counselling 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, OE) the students have to register by themselves through ERP.

### Student Support Services

1. **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.

2. **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.

3. **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.

## Evaluation Scheme (Theory Courses):

| **Evaluation Components** | **Weightage** |
|---|---|
| **Internal Assessment** | |
| **1. Continuous Assessment (30 Marks)** | |
| **(Minimum 5 components to be used and to be evenly spaced)** | |
| Project/ Quizzes/ Assignments and Essays/ Presentations/ Participation/ Case Studies/ Reflective Journals | **30 Marks** |
| **2. Mid Term Exam** | **20 Marks** |
| **External Assessment: -** | **50 Marks** |
| End term Examination | |
| **Total** | **100 Marks** |

## Evaluation Scheme (Laboratory/Practical Courses):

| **Evaluation Components** | **Weightage** |
|---|---|
| **Internal Assessment –** | |
| 1. Conduct of Experiment | 10 Marks |
| 2. Lab Records | 10 Marks |
| 3. Lab Participation | 10 Marks |
| 4. Lab Project | 20 Marks |
| **External Assessment-** | 50 Marks |
| End term Practical Exam and Viva Voce | |
| **Total** | **100 rks** |

## Feedback and Continuous Improvement Mechanisms:

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.

4. **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 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.

## Scheme of Studies

| | |
|---|---|
| Program Name | B.Tech (Computer Science & Engineering) |
| Total Credits | 168 |
| Total Semesters | 8 |

## Credit Distribution Summary

| Program Name | I | II | III | IV | V | VI | VII | VIII | Total Credits |
|---|---|---|---|---|---|---|---|---|---|
| B. Tech CSE | 23 | 24 | 26 | 24 | 24 | 21 | 14 | 12 | 168 |

## Semester I

| SN | Category | Course Code | Course Title | L | T | P | C |
|---|---|---|---|---|---|---|---|
| 1 | Major-1 | ENMA101 | Engineering Calculus | 3 | 1 | - | 4 |
| 2 | Major-2 | ENPH101/ENCH101 | Engineering Physics / Engineering Chemistry | 4 | - | - | 4 |
| 3 | Major-3 | ENEE101 | Basics of Electrical & Electronics Engineering | 4 | - | - | 4 |
| 4 | Major-4 | ENCS101 | Fundamentals of Computer programming using Python | 4 | - | - | 4 |
| 5 | Major-5 | ENPH151/ENCH151 | Engineering Physics Lab/ Engineering Chemistry lab | - | - | 2 | 1 |
| 6 | Major-6 | ENEE151 | Basics of Electrical & Electronics Engineering Lab | - | - | 2 | 1 |
| 7 | Major-7 | ENCS151 | Fundamentals of Computer Programming Lab* | - | - | 2 | 1 |
| 8 | VAC-1 | VAC-151 | Environmental Studies & Disaster Management | 2 | - | - | 2 |
| 9 | SEC-1 | **SEC067** | Essentials of Computer Science (MOOC-I) | - | - | - | 2 |
| TOTAL | | | | 14 | 4 | 6 | 23 |

## Semester II

| SN | Category | Course Code | Course Title | L | T | P | C |
|---|---|---|---|---|---|---|---|
| 1 | Major-8 | ENMA102 | Linear Algebra and Ordinary Differential Equations | 3 | 1 | - | 4 |
| 2 | Major-9 | ENCS102 | Object Oriented Programming using C++ | 3 | 1 | - | 4 |
| 3 | Major-10 | ENCH101/ ENPH101 | Engineering Chemistry / Engineering Physics | 1 | 1 | - | 4 |
| 4 | SEC-2 | SEC033 | Engineering Drawing & Workshop Lab | - | - | 4 | 2 |
| 5 | Major-11 | ENCS152 | Object Oriented Programming using C++ Lab | - | - | 2 | 1 |
| 6 | Major-12 | ENCH151/ ENPH151 | Engineering Chemistry Lab / Engineering Physics lab | - | - | 2 | 1 |
| 7 | Open Elective-1 | | Students can choose one of the electives from the pool of open electives of University | 3 | - | - | 3 |
| 8 | SEC-3 | | Applied Generative AI: Practical Tools and Techniques | - | - | 4 | 2 |
| 9 | PROJ-1 | ENSI152 | Minor Project-I | - | - | - | 2 |
| TOTAL | | | | 10 | 3 | 12 | 23 |

## Semester III

| SN | Category | Course Code | Course Title | L | T | P | C |
|---|---|---|---|---|---|---|---|
| 1 | Major-13 | ENCS201 | Java Programming | 3 | 1 | - | **4** |
| 2 | Major-14 | ENCS203 | Discrete Mathematics | 3 | 1 | - | **4** |
| 3 | Major-15 | ENCS205 | Data Structures | 3 | 1 | - | **4** |
| 4 | VAC-2 | | Students can choose any one elective from the pool of the VAC Courses offered by School | 2 | - | - | **2** |
| 5 | Major-16 | ENCS251 | Java Programming Lab | - | - | 2 | **1** |
| 6 | Major-17 | ENCS253 | Data Structures Lab | - | - | 2 | **1** |
| 7 | **AEC-1** | **AEC006** | **Verbal Ability** | **3** | **-** | **-** | **3** |
| 8 | INT-1 | ENSI251 | Summer Internship-I | - | - | - | **2** |
| 9 | Open Elective-2 | | Students can choose any one elective from the open pool of the university | 3 | - | - | **3** |
| 10 | SEC-4 | SEC034 | Fundamentals of AI & Machine Learning | 2 | - | | **2** |
| 11 | AUDIT-1 | | Competitive Coding- I | 2 | - | - | **0** |
| 12 | CS-1 | CS002 | Community Service | 1 | - | - | **1** |
| **TOTAL** | | | | **22** | **3** | **4** | **27** |

For "Summer Internship" students have to complete 6 weeks internship during the summers and submit a completion certificate. Students will be evaluated on a scale of 100 based on his learning outcomes during the 3rd semester for allocation of marks for internship.

Audit Course: it is zero credit and must pass course. End term exam of 100 marks will be conducted

*VAC-III

| CODE | COURSE TITLE | L | T | P | C |
|---|---|---|---|---|---|
| | | | | | |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| **VAC170** | Design thinking & Innovations for Engineers | - | - | - | 2 |
| **VAC171** | AWS Cloud Fundamentals | - | - | - | 2 |
| **VAC172** | Web Development with open-source Frameworks | - | - | - | 2 |
| **VAC173** | Google Data Analytics | - | - | - | 2 |
| **VAC174** | Software Testing using Open-Source Frameworks | - | - | - | 2 |
| **VAC175** | Database Management with Open-Source Frameworks | - | - | - | 2 |
| **VAC176** | Cyber Security with Open-source Frameworks | - | - | - | 2 |
| **VAC185** | Practical Robotics and UAV Applications | - | - | - | 2 |
| **VAC186** | Applied Automotive Engineering: Hands-On Practices and Innovations | - | - | - | 2 |
| **VAC187** | Practical Research Methodology for Engineers | - | - | - | 2 |

## Semester IV

| SN | Category | Course Code | Course Title | L | T | P | C |
|---|---|---|---|---|---|---|---|
| 1 | Major-18 | ENMA202 | PROBABILITY AND STATISTICS | 3 | 1 | - | **4** |
| 2 | SEC-5 | SEC035 | Web Programming with Python and JavaScript Lab | - | - | 4 | **2** |
| 3 | Major-19 | ENCS202 | Analysis and Design of Algorithms | 3 | 1 | - | **4** |
| 4 | Major-20 | ENCS204 | Database Management Systems | 3 | 1 | - | **4** |
| 5 | AEC-2 | **AEC007** | **Communication & Personality Development** | 3 | - | - | **3** |
| 6 | Major-21 | ENCS254 | Database Management Systems Lab | - | - | 2 | **1** |
| 7 | Major-22 | ENCS256 | Analysis and Design of Algorithms Lab | - | - | 2 | **1** |
| 8 | Open Elective-3 |  | Students can choose any one elective from the open pool of the university | 3 | - | - | **3** |
| 9 | PROJ-2 | ENSI252 | Minor Project-II | - | - | - | **2** |
| 10 | AUDIT-2 |  | Competitive Coding- II | 2 | - | - | **0** |
| 11 | CS-2 | CS001 | Club/Society | 1 | - | - | **1** |
| **TOTAL** |  |  |  | **18** | **3** | **8** | **25** |

Note: For the "Minor Project," students will undergo internal evaluation, which will be graded on a scale of 100 marks.

Audit Course: it is zero credit and must pass course. End term exam of 100 marks will be conducted

## Semester V

| SN | Category | Course Code | Course Title | L | T | P | C |
|---|---|---|---|---|---|---|---|
| 1 | Major-23 | ENCS301 | Theory of Computation | 3 | 1 | - | 4 |
| 2 | Major-24 | ENCS303 | Operating Systems | 3 | 1 | - | 4 |
| 3 | Major-25 | ENCS351 | Operating System Lab | - | - | 2 | 1 |
| 4 | INT-2 | ENSI351 | Summer Internship-II | - | - | - | **2** |
| 5 | AEC-3 | **AEC008** | **Arithmetic and Reasoning Skills-I** | 3 | - | - | **3** |
| 6 | DSE-1 | | Discipline Specific Elective-I | 4 | - | - | **4** |
| 7 | DSE-2 | | Discipline Specific Elective-I lab | - | - | 2 | **1** |
| 8 | Major-26 | ENCS305 | Software Engineering | 4 | - | - | **4** |
| 9 | AUDIT-3 | | Competitive Coding - III | 2 | - | - | **0** |
| | | | **TOTAL** | **19** | **2** | **4** | **23** |

For "Summer Internship" students have to complete 6 weeks internship during the summers and submit a completion certificate. Students will be evaluated on a scale of 100 during the 3rd semester for allocation of marks for internship.

Audit Course: it is zero credit and must pass course. End term exam of 100 marks will be conducted

Discipline Specific Elective I (CSE)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| (i) | DSE | ENSP301 | Secure Coding and Vulnerabilities | 4 | - | - | 4 |
| | DSE | ENSP351 | Secure Coding and Vulnerabilities lab | - | - | 2 | 1 |
| (ii) | DSE | ENSP303 | Cyber Crime Investigation & Digital Forensics | 4 | - | - | 4 |
| | DSE | ENSP353 | Cyber Crime Investigation & Digital Forensics lab | - | - | 2 | 1 |
| (iii) | DSE | ENSP305 | AI in Cyber Security | 4 | - | - | 4 |
| | DSE | ENSP355 | AI in Cyber Security Lab | - | - | 2 | 1 |
| (iv) | DSE | ENSP307 | Social Media Security | 4 | - | - | 4 |
| | DSE | ENSP357 | Social Media Security Lab | - | - | 2 | 1 |

## Semester VI

| SN | Category | Course Code | Course Title | L | T | P | C |
|---|---|---|---|---|---|---|---|
| 1 | Major-27 | ENCS302 | Computer Organization & Architecture | 3 | 1 | - | **4** |
| 2 | Major-28 | ENCS304 | Computer Networks | 3 | 1 | - | **4** |
| 3 | Major-29 | ENCS306 | Introduction of Neural Network and Deep Learning | 4 | - | - | **4** |
| 4 | Major-30 | ENCS352 | Computer Networks Lab | - | - | 2 | **1** |
| 5 | Major-31 | ENCS354 | Introduction to Neural Networks & Deep Learning Lab | - | - | 2 | **1** |
| 6 | DSE-3 | | Discipline Specific Elective -II | 4 | - | - | **4** |
| 7 | DSE-4 | | Discipline Specific Elective -II Lab | - | - | 2 | **1** |
| 8 | PROJ-3 | ENSI352 | Minor Project-III | - | - | - | **2** |
| 9 | AUDIT-4 | | Competitive Coding - IV | 2 | - | - | **0** |
| **TOTAL** | | | | **16** | **2** | **6** | **21** |
| Note: For the "Minor Project," students will undergo internal evaluation, which will be graded on a scale of 100 marks. | | | | | | | |
| Audit Course: it is zero credit and must pass course. End term exam of 100 marks will be conducted | | | | | | | |

Discipline Specific Elective II (Artificial Intelligence)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| (i) | DSE | ENSP302 | Natural Language Processing | 4 | - | - | 4 |
| | DSE | ENSP352 | Natural Language Processing lab | - | - | 2 | 1 |
| (ii) | DSE | ENSP304 | Image Processing & Computer Vision | 4 | - | - | 4 |
| | DSE | ENSP354 | Image Processing & Computer Vision lab | - | - | 2 | 1 |
| (iii) | DSE | ENSP306 | Introduction to Generative AI | 4 | - | - | 4 |
| | DSE | ENSP356 | Generative AI lab | - | - | 2 | 1 |
| (iv) | DSE | ENSP308 | Transfer Learning | 4 | - | - | 4 |
| | DSE | ENSP358 | Transfer Learning lab | - | - | 2 | 1 |

## Semester VII

| SN | Category | Course Code | Course Title | L | T | P | C |
|---|---|---|---|---|---|---|---|
| 1 | DSE-5 | | Discipline Specific Elective-III | 4 | - | - | 4 |
| 2 | DSE-6 | | Discipline Specific Elective-IV | 4 | - | - | 4 |
| 3 | DSE-7 | | Discipline Specific Elective-III Lab | - | - | 2 | 1 |
| 4 | DSE-8 | | Discipline Specific Elective-IV lab | - | - | 2 | 1 |
| 5 | INT-3 | ENSI451 | Summer Internship-III | - | - | - | 2 |
| 6 | MOOC-2 | | Applied Programming and Problem-Solving Skills for Campus Interviews | - | - | - | 2 |
| TOTAL | | | | 8 | 0 | 4 | 14 |

| Discipline Specific Elective - III (Cloud Computing) | | | | | | | |
|---|---|---|---|---|---|---|---|
| (i) | DSE | ENSP401 | Computational Services in The Cloud | 4 | - | - | 4 |
| | DSE | ENSP451 | Computational Services in The Cloud Lab | - | - | 2 | 1 |
| (ii) | DSE | ENSP403 | Microsoft Azure Cloud Fundamentals | 4 | - | - | 4 |
| | DSE | ENSP453 | Microsoft Azure Cloud Fundamentals Lab | - | - | 2 | 1 |
| (iii) | DSE | ENSP405 | Storage and Databases on Cloud | 4 | - | - | 4 |
| | DSE | ENSP455 | Storage and Databases on Cloud Lab | - | - | 2 | 1 |
| (iv) | DSE | ENSP407 | Application Development and DevOps on Cloud | 4 | - | - | 4 |
| | DSE | ENSP457 | Application Development and DevOps on Cloud Lab | - | - | 2 | 1 |

| Discipline Specific Elective - IV (Full Stack Development) | | | | | | | |
|---|---|---|---|---|---|---|---|
| (i) | DSE | ENSP409 | Mobile Application Development using iOS | 4 | - | - | 4 |
| | DSE | ENSP459 | Mobile Application Development using iOS Lab | - | - | 2 | 1 |
| (ii) | DSE | ENSP411 | DevOps & Automation | 4 | - | - | 4 |
| | DSE | ENSP461 | DevOps & Automation Lab | - | - | 2 | 1 |
| (iii) | DSE | ENSP413 | .Net FRAMEWORK | 4 | - | - | 4 |
| | DSE | ENSP463 | .Net FRAMEWORK Lab | - | - | 2 | 1 |
| (iv) | DSE | ENSP415 | New Age Programming languages | 4 | 0 | 0 | 4 |
| | DSE | ENSP465 | New Age Programming languages Lab | 0 | 0 | 2 | 1 |

## Semester VIII

| SN | Category | Course Code | Course Title | L | T | P | C |
|---|---|---|---|---|---|---|---|
| 1 | PROJ | ENSI452 | Industrial Project/R&D Project/Start-up Project | - | - | - | 12 |
| TOTAL | | | | | | | 12 |

## Syllabus

## Semester: I

## ENGINEERING CALCULUS

| | | | | |
|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name: Engineering Calculus** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENMA101** | 3-1-0 | 4 | 40 |

**Type of Course:** Major-1

**Pre-requisite(s):** Basics understanding of Calculus and Algebra at higher secondary level

**Course Perspective.** This course is to familiarize students with techniques in calculus, multivariate calculus, vector calculus, and their applications. It aims to equip students with standard concepts and tools from intermediate to advanced levels that will enable them to tackle more advanced mathematical and engineering problems relevant to their disciplines. The course is divided into 4 modules:

a) Differential Calculus- I
b) Multivariable Calculus (Partial Differentiation and applications)
c) Multivariable Calculus-II (Integration)
d) Vector Calculus

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|

| | |
|---|---|
| **CO 1** | **Understanding** fundamental concepts of differential calculus |
| **CO 2** | **Applying** concepts of multivariable calculus |
| **CO 3** | **Solving** integration problems in multiple dimensions |
| **CO 4** | **Analyzing** vector calculus to understand physical and geometrical problems |

**Course Outline:**

| **Unit Number: 1** | **Title: Differential Calculus- I** | **No. of hours: 10** |
|---|---|---|

**Content:**

- Introduction to limits, continuity, and differentiability.
- Rolle's Theorem, Lagrange's Mean value theorem with geometrical interpretation and applications.
- Cauchy's Mean Value Theorem.
- Taylor's Series.
- Applications of definite integrals to evaluate surface areas and volumes of revolutions of curves (Cartesian coordinates).
- Successive Differentiation, Leibnitz theorem and its application.
- Curve tracing in Cartesian and Polar coordinates.
- Infinite series: Tests for convergence of series (Comparison, Ratio, Root test)
- Alternating series
- Absolute convergence
- Conditional convergence.

| **Unit Number: 2** | **Title: Multivariable Calculus (Partial Differentiation and applications)** | **No. of hours: 10** |
|---|---|---|

**Content:**

- Partial derivatives.
- Total derivative.
- Euler's Theorem for homogeneous functions.
- Taylor and Maclaurin theorems for functions of one and two variables.

- Maxima and Minima of functions of several variables.
- Lagrange Method of Multipliers.

**Unit Number: 3** **Title: Multivariable Calculus-II (Integration)** **No. of hours: 10**

**Content:**

- Area between two curves; Polar Coordinates.
- Volumes by slicing, Washer and Shell Methods.
- Length of a plane curve.
- Areas of Surfaces of Revolution.
- Evaluation of Double Integrals (Cartesian and polar coordinates).
- Change of order of integration (Cartesian form).
- Evaluation of Triple Integrals: Change of variables (Cartesian to polar for double, Cartesian to Spherical and Cylindrical polar for triple integrals).
- Applications: Areas (by double integrals) and volumes (by double and triple integrals).
- Centre of mass and centre of gravity (Constant and variable densities).

**Unit Number: 4** **Title: Vector Calculus** **No. of hours: 10**

**Content:**

- Vector differentiation: Gradient, Curl, and Divergence with physical interpretation.
- Directional derivatives, Tangent and Normal planes.
- Vector Integration: Line integral, Surface integral, Volume integral.
- Applications to work done by the force
- Gauss's Divergence theorem, Green's theorem, Stoke's theorem (without proof) and applications.

**Learning Experience:**

**Classroom Learning Experience**

1. **Interactive Lectures**: Use PPTs and MATLAB to explain key calculus concepts.
2. **Conceptual Understanding**: Cover theorems (Rolle's, Taylor's, etc.) and solve problems.
3. **Problem-Solving Sessions**: In-class exercises on differential and multivariable calculus.
4. **Theory Assignments**: Solve theoretical problems, reviewed in class.
5. **Group Work**: Collaborative problem-solving for real-world engineering tasks.
6. **Case Studies**: Discuss real-world applications of calculus concepts.
7. **Continuous Feedback**: In-class quizzes and feedback sessions.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Apply calculus techniques in take-home assignments.
2. **Question Bank**: Practice with model papers and self-assessment.
3. **Online Forums**: Discuss and collaborate on calculus problems online.
4. **Self-Study**: Research and apply calculus to real-world scenarios.
5. **Collaborative Projects**: Group work on applying multivariable and vector calculus.

**Textbooks:**

- G.B. Thomas and R.L. Finney, Calculus and Analytic Geometry, 9th Edition, Pearson, Reprint, 2002.

**Reference Books:**

- B. V. Ramana, Higher Engineering Mathematics, Tata Mc Graw-Hill, 2008.
- B. S. Grewal, Higher Engineering Mathematics, Khanna Publisher, 2005.
- R K. Jain & S R K. Iyenger, Advance Engineering Mathematics, Narosa Publishing House, 2002.
- E. Kreyszig, Advanced Engineering Mathematics, John Wiley & Sons, 2005.
- Ray Wylie C and Louis C Barret, Advanced Engineering Mathematics, Tata Mc-Graw-Hill, Sixth Edition.

**Additional Readings:**

1. Link to NPTEL course contents: https://onlinecourses.nptel.ac.in/noc18_ma05/preview
2. Link to topics related to course:
https://www.whitman.edu/mathematics/calculus_online/chapter14.html

# ENGINEERING PHYSICS

| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
|---|---|---|---|---|
| **Course Name:** **Engineering Physics** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENPH101** | 3-1-0 | 4 | 40 |
| **Type of Course:** | Major-2 | | | |
| **Pre-requisite(s), if any: Knowledge of Basic physics and calculus (differentiation, integration).** | | | | |

**Course Perspective.** This course introduces students to the fundamental concepts of Engineering Physics, bridging the gap between theoretical physics principles and practical engineering applications. Engineering Physics is crucial for understanding and designing new technologies and systems in various engineering fields such as electronics, materials science, and mechanical engineering. Students will explore core topics including mechanics, Optics, Polarization, and modern physics, with a special emphasis on their relevance to real-world engineering problems. The course is divided into 4 modules:

a) Mechanics
b) Optics
c) Polarization
d) New Engineering Materials

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the principles and applications of lasers, fiber optics, and electromagnetic waves. |
| **CO 2** | **Applying** the concepts of polarization to analyze and manipulate light in various optical systems. |
| **CO 3** | **Evaluating** the properties and applications of dielectric materials, superconducting materials, and nano-materials in engineering contexts. |

|  |  |
|---|---|
| **CO 4** | **Designing** and propose innovative applications of lasers, fiber optics, and smart materials for specific engineering challenges. |
| **CO 5** | **Analyzing** problems related to the behavior of electromagnetic waves, polarization, and optical communication systems**.** |

**Course Outline:**

**Unit Number: 1** **Title: Mechanics** **No. of hours: 10**

**Content:**

Centre of mass, centre of mass of two particle system and a rigid body, Rotational motion, Moment of Inertia and its physical significance, Radius of gyration, Acceleration due to gravity, simple harmonic motion, differential equation of S.H.M., Examples of S.H.M. (simple and compound pendulum)

**Unit Number: 2** **Title: Optics** **No. of hours: 10**

**Content:**

**Light**: Introduction of light, properties of light, Dual Nature of light, refraction, Refraction by prism, Interference of light, interference by divison of wavefront (Young’s double slit experiment), Interference by division of wave amplitude (Newton’s ring), difference between diffraction and interference, types of diffraction, Fraunhoffer diffraction (single and double slit), theory of plane diffraction grating, determination of wavelength of a spectral line using transmission grating

**Laser**: Introduction, principle of Laser, stimulated and spontaneous emission, Ruby laser, He-Ne Laser, Application of Lasers.

**Unit Number: 3** **Title: Polarization** **No. of hours: 10**

**Content:**

**Polarization:** Polarization by reflection and refraction, Brewster's law, double refraction, nicol prism, quarter and half-wave plates, Production and analysis of circularly and elliptically polarized light

**Unit Number: 4** **Title: New Engineering Materials** **No. of hours: 10**

**Content:**

**Dielectric materials**: Definition – Dielectric Breakdown – Dielectric loss – Internal field – Claussius Mossotti relation.

**Superconducting materials**: Introduction – Properties- Meissner effect – Type I & Type II superconductors – BCS theory-Applications.

**Nanomaterials**: Introduction – Synthesis of nano materials – Top down and Bottom-up approach- Ball milling- PVD method- Applications. Smart materials: Shape memory alloys-Biomaterials (properties and applications)

**Learning Experience**

**Inside Classroom Learning Experience:**

- Interactive Lectures: Use PPTs to explain key concepts.
- Conceptual Understanding: Cover topics like SHM, polarization, and laser principles through real-world examples and problem-solving.
- Problem-Solving Sessions: Engage students with in-class exercises on mechanics, optics, and electromagnetic waves to strengthen theoretical understanding.
- Theory Assignments: Assign problems on polarization and nanomaterials to be solved during class and discussed with peers.
- Group Work: Students collaborate on real-world engineering tasks, such as analyzing optical systems using lasers or fiber optics.
- Case Studies: Discuss engineering applications of superconducting materials and smart materials to connect theory with industry practices.
- Continuous Feedback: Conduct quizzes and short assessments in class to provide immediate feedback on students' grasp of core physics principles.

**Outside Classroom Learning Experience:**

- Theory Assignments: Assign take-home problems that require applying physics concepts like laser technology or material properties to practical situations.
- Question Bank: Encourage students to practice with model questions related to mechanics, optics, and materials for self-assessment and revision.
- Online Forums: Students participate in online discussions to collaborate on solving physics problems, particularly in areas like polarization and diffraction.
- Self-Study: Research and explore modern physics technologies, such as the latest advancements in superconducting materials or nanotechnology.
- Collaborative Projects: Work in teams to design innovative applications of physics concepts, such as building models based on polarization or developing smart materials for engineering challenges.

**Textbooks:**

1. Fundamentals of Physics" by David Halliday, Robert Resnick, and Jearl Walker
2. University Physics with Modern Physics" by Hugh D. Young and Roger A. Freedman

**Reference Book**

1. Engineering Physics" by Gaur and Gupta
2. Concepts of Modern Physics" by Arthur Beiser
3. Physics for Scientists and Engineers" by Raymond A. Serway and John W. Jewett.

**Additional Readings:**

**Online Learning Resources for Engineering Physics**

**R 1.** **MIT OpenCourseWare - Physics**

- MIT offers a variety of free courses in physics that cover topics from classical mechanics to quantum physics.
- Link: MIT OpenCourseWare Physics

**R 2.** **Physics LibreTexts**

- A comprehensive online library covering numerous topics in physics at various levels of complexity. It's part of the LibreTexts project, which aims to develop freely accessible textbooks.
- Link: Physics LibreTexts

**R 3.** **YouTube - Stanford University Physics Lectures**

- This channel features recorded lectures from Stanford University's physics department, covering advanced topics such as quantum mechanics and general relativity.
- Link: Stanford Physics YouTube

## ENGINEERING PHYSICS LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **Engineering Physics Lab** | **ENPH151** | 0-0-2 | 1 |
| **Type of Course:** | Major-5 | | |
| **Pre-requisite(s), if any:** | | | |

**Defined Course Outcomes**

| COs | **Statements** |
|---|---|
| CO 1 | **Understanding** the principles and concepts related to the experiments involving bar pendulum, flywheel, Kater's pendulum, Newton's ring apparatus, plane diffraction grating, spectrometer, and half shade polarimeter. |
| CO 2 | **Applying** the principles and concepts learned to conduct experiments and analyze experimental data, plot graphs, and interpret the results to determine various physical quantities. |
| CO 3 | **Evaluating** the accuracy and reliability of experimental measurements and results obtained from the conducted experiments. |
| CO 4 | **Applying** critical thinking and problem-solving skills to troubleshoot experimental setups, identify sources of errors, and propose solutions to improve the accuracy and precision of measurements |

## LAB EXPERIMENTS

| **Ex. No** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| 1 | To plot a graph between the distance of the knife edge from the center of gravity and the time period of the bar pendulum. From the graph, find the acceleration due to gravity, the radius of gyration and the moment of inertia of the bar about an axis. | CO2, CO3 |

|  |  |  |
|---|---|---|
| 2 | To determine the moment of inertia of a flywheel about its own axis of motion. | CO1, CO2, CO3, CO4 |
| 3 | To determine the value of acceleration due to gravity using Kater`s pendulum. | CO1, CO2, CO3, CO4 |
| 4 | To determine the wavelength of sodium light using Newton`s ring apparatus. | CO1, CO2, CO3 |
| 5 | To determine the wavelength of prominent lines of mercury by plane diffraction grating. | CO1, CO2, CO3 |
| 6 | To determine the refractive index of the material of the prism for the given colours (wavelengths) of mercury light with the help of spectrometer. | CO1, CO2, CO3 |
| 7 | To determine the specific rotation of cane sugar solution with the help of half shade polarimeter. | CO1, CO2, CO3, CO4 |
| 8 | To determine the wavelength of He-Ne LASER using transmission diffraction grating. | CO1, CO2, CO3 |

# BASICS OF ELECTRICAL & ELECTRONICS ENGINEERING

| | | | | |
|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Basics of Electrical & Electronics Engineering** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENEE101** | 4-0-0 | 4 | 40 |

**Type of Course:** Major-3

**Pre-requisite(s), if any:** Calculus knowledge at higher secondary level

**Course Perspective.** This course provides a foundational understanding of both electrical and electronics engineering principles, crucial for students in diverse engineering disciplines. It is designed to impart knowledge of circuit analysis, AC and DC circuits, semiconductor physics, transistors and digital electronics. Students will gain a solid grounding in theoretical concepts and practical applications, enabling them to analyse, design, and troubleshoot electrical and electronic circuits.

The course is structured into four main modules, each addressing key areas of electrical and electronics engineering:

a) Circuit Analysis
b) A.C. Circuits & Cathode Ray Oscilloscope (CRO)
c) Semiconductor Physics & Transistors
d) Digital Electronics

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Appling** Ohm's Law, Kirchhoff's Laws, and various network theorems to analyze and solve DC circuits effectively. |
| **CO 2** | **Analyzing** the behavior of R-L, R-C, and R-L-C circuits in both series and parallel configurations under sinusoidal inputs, and understand the concept of resonance, Q-factor, and bandwidth. |

| | |
|---|---|
| **CO 3** | **Understanding** the principles of semiconductor materials, diode operations, and transistor functions. |
| **CO 4** | **Analyzing** and **Designing** fundamental digital circuits by applying principles of logic gates, number systems, and Boolean algebra. |

**Course Outline:**

| **Unit Number: 1** | **Title: Circuit Analysis:** | **No. of hours: 10** |
|---|---|---|
| **Content Summary:**<br>Ohm's Law, KCL, KVL Mesh and Nodal Analysis, Circuit parameters, energy storage aspects, Superposition, Thevenin's, Norton's, Reciprocity, Maximum Power Transfer Theorem, Millman's Theorem, Star-Delta Transformation. Application of theorem to the Analysis of D.C. circuits. | | |
| **Unit Number: 2** | **Title: A.C. Circuits & CRO** | **No. of hours: 10** |
| **Content Summary:**<br>**A.C. Circuits:** R-L, R-C, R-L-C circuits (series and parallel), Time Constant, Phasor representation, Response of R-L, R-C and R-L-C circuits to sinusoidal input Resonance-series and parallel R-L-C Circuits, Q-factor, Bandwidth.<br>**Cathode Ray Oscilloscope:** Basic CRO circuit (Block Diagram), Cathode ray tube (CRT) & its component, Multimeter. | | |
| **Unit Number: 3** | **Title: Semiconductor Physics & Transistors** | **No. of hours: 10** |
| **Content Summary:**<br>**Semiconductor Physics:** Basic concepts, Intrinsic and extrinsic semiconductors.<br>**P-N Junction Diode:** Ideal diode, P-N junction under open-circuit and closed-circuit, Effect of Temperature, Applications of Diodes.<br>**Special Diodes:** Zener Diode, Photodiode, Light Emitting Diodes.<br>**Transistors:** Introduction to transistors, Bipolar Junction Transistor (BJT) - construction, operations, characteristics, and configurations, current gain, Introduction to FETs and its types. | | |
| **Unit Number: 4** | **Title: Digital Electronics** | **No. of hours: 10** |
| **Content Summary:**<br>**Basic Concepts:** Digital vs. Analog Signals, Number Systems, Logic Gates and Boolean Algebra, Sum of Product and Product of Sum, K-map up to 4 variables, types of digital circuits. | | |

**Combinational Circuits:** Adders, Subtractors, Multiplexers and Demultiplexers, Encoders and Decoders, 7 segment display.
**Sequential Circuits:** Flip-Flops and its types.

**Learning Experiences**

**Inside Classroom Learning Experience:**

- Interactive Lectures: Present core concepts of circuit analysis, AC/DC circuits, semiconductor physics, and digital electronics using PowerPoint presentations and visual aids.
- Demonstrations & Simulations: Use circuit simulation tools to demonstrate the behavior of DC and AC circuits, resonance in R-L-C circuits, and the operation of semiconductor devices like diodes and transistors.
- Problem-Solving Sessions: Conduct in-class problem-solving sessions where students analyze circuits using Ohm's Law, Kirchhoff's Laws, and network theorems. Have students work through R-L-C circuits and solve problems on resonance, bandwidth, and Q-factor.
- Concept Application Discussions: Engage students in discussions to link theoretical knowledge of semiconductors, transistors, and digital logic gates with real-world applications.
- Hands-on Lab Activities: Facilitate lab sessions where students use equipment like multimeters and cathode ray oscilloscopes (CROs) to measure and analyze circuits.
- Group Assignments: Students work in teams to design and analyze small electrical systems, such as combining logic gates to build simple digital circuits or constructing transistor-based amplifiers.
- Case Studies & Real-world Applications: Use case studies to demonstrate practical applications of digital electronics in systems such as calculators or logic controllers.

**Outside Classroom Learning Experience:**

- Theory Assignments: Assign problems that require analyzing more complex circuits, applying theorems to DC and AC circuits, and designing digital circuits using logic gates.
- Laboratory Report Writing: Students complete reports on laboratory exercises, documenting their methods and findings from using CROs and other measurement tools to analyze electrical circuits.
- Collaborative Online Discussions: Create forums where students can discuss challenging topics like circuit analysis, semiconductor behavior, and digital electronics. Encourage peer problem-solving.
- Independent Research: Assign students to research and present on topics like the latest semiconductor technologies or applications of digital electronics in modern devices.

- Project-Based Learning: Students work in groups on a final project that involves designing a comprehensive electronic system, incorporating transistors, digital logic, and other circuit elements.
- Flipped Classroom Approach: Encourage students to review online materials and videos on topics such as AC circuit analysis and digital electronics before class to engage more deeply during lecture and problem-solving sessions.

**Textbooks**

1. B L Thareja – A textbook of Electrical Technology
2. Boylestad & Nashelsky, "Electronic Devices & Circuits", Pearson Education, 10th Edition
3. M. Morris Mano, "Digital design", Pearson Education, 6th Edition

**Reference Book**

1. D.P. Kothari & I J Nagrath, Basic Electrical Engineering, Tata McGraw Hill, New Delhi
2. V. K. Mehta & Rohit Mehta, "Principles of Electronics", S. Chand Publishers, 27th Edition

**Additional Readings:**

**Online Learning Resources for Electrical & Electronics Engineering**

**R 1. MIT OpenCourseWare - Electrical Engineering and Computer Science**

- MIT provides free access to a variety of courses in Electrical Engineering and Computer Science, ranging from foundational circuit analysis to advanced topics like digital systems and machine learning. These courses include lecture notes, assignments, and exams.
- Link: MIT OpenCourseWare - EECS

**R 2. All About Circuits**

- A comprehensive online resource that offers tutorials, textbooks, and forums covering key topics in electrical and electronics engineering, such as circuit design, semiconductors, and digital logic. It's widely used for both educational and practical applications in electronics.
- Link: All About Circuits

**R 3. YouTube - NPTEL Electrical Engineering**

- NPTEL (National Programme on Technology Enhanced Learning) offers free recorded lectures from India's top technical institutions. The Electrical Engineering playlist includes topics such as circuit theory, semiconductor devices, and digital electronics.
- Link: NPTEL Electrical Engineering YouTube

## BASICS OF ELECTRICAL & ELECTRONICS

# ENGINEERING LAB

| Program Name | B. Tech (Computer Science and Engineering) | | |
|---|---|---|---|
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **Basics of Electrical & Electronics Lab** | **ENEE151** | 0-0-2 | 1 |

**Type of Course:** Major-6

**Pre-requisite(s):** Calculus knowledge at higher secondary level

**Defined Course Outcomes**

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the working principles and applications of essential electronic instruments such as CRO, multimeter, function generator, and power supply, and utilize them for various measurements. |
| **CO 2** | **Applying** theoretical knowledge to verify fundamental electrical theorems and analyze their practical implications in electrical circuits |
| **CO 3** | **Developing** skills to measure and analyze electrical parameters such as voltage, current, power, and impedance in AC and DC circuits, and study the behavior of electronic components |
| **CO 4** | **Evaluating** and interpret the characteristics of semiconductor devices and digital logic circuits, and demonstrate proficiency in using electronic components and displays |

## LAB EXPERIMENTS

| Ex. No. | Experiment Title | Mapped CO/COs |
|---|---|---|
| 1 | To get familiar with the working knowledge of the following instruments: a) Cathode ray oscilloscope (CRO) b) Multimeter (Analog and Digital) c) Function generator d) Power supply | CO1 |

|  |  |  |
|---|---|---|
| 2 | To measure phase difference between two waveforms using CRO To measure an unknown frequency from Lissajous figures using CRO | CO1 |
| 3 | To Verify the Thevenin's and Norton’s theorem | CO2 |
| 4 | To Verify the Superposition theorem | CO2 |
| 5 | To measure voltage, current and power in an A.C. circuit by LCR impedance method | CO3 |
| 6 | To study the frequency response curve in series and parallel-L-C circuit | CO3 |
| 7 | To plot the forward and reverse V-I characteristics of P-N junction diode | CO4 |
| 8 | To plot and study the input and output characteristics of BJT in common-emitter configuration. | CO4 |
| 9 | Verification of truth tables of logic gates (OR, AND, NOT, NAND, NOR). | CO4 |
| 10 | To get familiar with the working and use of seven-segment display. | CO4 |

## FUNDAMENTALS OF COMPUTER PROGRAMMING

| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
|---|---|---|---|---|
| **Course Name: Fundamentals of Computer Programming** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCS101** | 4-0-0 | 4 | 40 |

**Type of Course:** Major-4

**Pre-requisite(s), if any:** None

**Course Perspective.** This course introduces students to the foundational aspects of computer programming, with a focus on understanding computer fundamentals, programming in Python, and applying these skills to data pre-processing, classification, and visualization. Designed to equip students with both theoretical knowledge and practical programming skills, the course bridges the gap between computer science principles and their application in real-world scenarios.

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| CO 1 | **Understanding** the fundamentals of Python, including syntax, variables, data types, and operators, and apply them in basic programming tasks |
| CO 2 | **Applying** control flow structures and functions to solve problems using Python's built-in data structures like lists, dictionaries, sets, and tuples. |
| CO 3 | **Implementing** object-oriented programming concepts and manage file handling and exception handling to develop robust Python applications. |
| CO 4 | **Analyzing** and **visualizing** data using Python libraries, and explore the basics of web development and client-server architecture. |

**Course Outline:**

| Unit Number: 1 | Title: Getting started with Python | No. of hours: 10 |
|---|---|---|
| **Content:**<br>**Introduction to Python Programming:** History of Python, Python Features, Local Environment Setup, setting up Python environment: Installing Python, IDEs (e.g., VSCode, Anaconda, PyCharm);<br>**Python Programming Basics:** Python Syntax, Keywords, Understanding Variables, numbers, and data types.<br>**Operators:** Arithmetic, Assignment, Comparison, Logical, Identity, Membership, Bitwise.<br>**Python String:** Manipulating strings, Modify Strings, String Concatenation, Format – Strings, Escape Characters, Inbuilt method of Strings. | | |
| **Unit Number: 2** | **Title: Control Flow and Data Structures in Python** | **No. of hours: 10** |

| | | |
|---|---|---|
| **Content:**<br>**Conditional statements**: if, elif, else; Loops: for loop, while loop, nested loops; Control flow statements: break, continue.<br>**Functions**: Defining functions, parameters, function calls, return statement; Scope and lifetime of variables. Recursive and Lambda Functions.<br>**Basics of Python Data Structure**: Mutable: List, Dictionary, Set, Immutable Types: Numbers, String, tuple.<br>**Lists**: Operations, methods, slicing; Tuples and sets: Properties, operations; Dictionaries: Creating, accessing, modifying. | | |
| **Unit Number: 3** | **Title: Object-Oriented Programming and File Handling in Python** | **No. of hours: 10** |
| **Content:**<br>**OOPs Concept:** Introduction to object-oriented programming (OOP), Abstraction, encapsulation, Polymorphism.<br>**Classes and objects:** Defining classes, creating objects; Constructors in Python, Parameterized and Non-parameterized.<br>**Inheritance, and polymorphism:** Types, Method overriding and overloading; Special methods (dunder methods): __init__, __str__, __repr__.<br>**File handling:** Opening, reading, writing, and closing files;<br>**Exception handling**: try, except, finally blocks. | | |
| **Unit Number: 4** | **Title: Data handling and web development in python** | **No. of hours: 10** |
| **Content:**<br>**Data visualization with matplotlib:** line plot, multiple subplots in one figure, histograms, bar charts, pie charts, scatter plots<br>**Handling data with pandas:** series, dataframes, read and write csv file, operations using dataframe<br>**Numpy arrays:** numpy - datatype, array operations, statistical functions.<br>**Introduction to Web Development with Python:** Overview of web development<br>Basics of client-server architecture. | | |

**Learning Experiences**

**Inside Classroom Learning Experience:**

- Interactive Lectures and Presentations: Engage students with lecture slides (PPTs) that introduce the history, features, and basic programming constructs of Python, such as syntax, data types, and operators.
- Problem-Based Learning Assignments: Use in-class problem-solving activities to help students understand core Python programming concepts like control flow (if-else, loops), functions, and basic data structures (lists, tuples, sets, dictionaries).
- Hands-On Lab Sessions: Facilitate lab exercises where students apply their theoretical knowledge by coding in Python. Focus on creating programs that demonstrate control structures and data manipulation techniques.
- Object-Oriented Programming Exercises: Use lecture notes and class exercises to introduce object-oriented programming (OOP) concepts, such as classes, objects, methods, inheritance, and encapsulation. Include activities that involve creating and manipulating objects and handling file operations and exceptions.
- Interactive Data Handling Sessions: Teach data handling and visualization techniques using libraries like pandas, numpy, and matplotlib. Provide practical exercises in class to reinforce these skills.
- Class Discussions and Review: End each topic with short quizzes or interactive discussions to assess student comprehension and offer immediate feedback.

**Outside Classroom Learning Experience:**

- Mini Projects and Lab Assignments: Assign practical mini-projects that allow students to apply their knowledge of Python to real-world problems. Projects could involve working with control flows, data structures, or object-oriented programming.
- Self-Directed Learning on Data Visualization: Encourage students to explore additional data handling and visualization tools by completing exercises outside of class, using libraries like matplotlib, pandas, and numpy.
- Question Bank and Practice Tests: Provide students with a bank of sample questions and past papers to practice Python programming concepts independently. This will help students review and test their understanding outside of class.
- Independent Study on Web Development: Encourage students to research basic web development concepts and how Python integrates into web technologies, allowing them to expand their learning through self-study.
- Online Forums and Collaboration: Set up or encourage participation in online forums where students can collaborate, ask questions, and solve problems related to Python programming.

- Capstone Projects: For deeper learning, students can work on larger projects that integrate multiple Python concepts, such as OOP, data handling, and visualization, providing an opportunity for independent exploration and problem-solving.

**Text Book**

1. John V Guttag. "Introduction to Computation and Programming Using Python", Prentice Hall of India

**Reference Book**

1. R. Nageswara Rao, "Core Python Programming", Dreamtech
2. Wesley J. Chun. "Core Python Programming, Second Edition", Prentice Hall
3. Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms in Python", Wiley

**Additional Readings:**

**R 1.** https://www.tutorjoes.in/python_programming_tutorial/

**R 2.** https://www.udemy.com/course/100-days-of-code/

**R 3.** https://favtutor.com/blog-details/7-Python-Projects-For-Beginners

**R 4.** https://github.com/NaviRocker/100-days-of-python

**R 5.** https://hackr.io/blog/python-projects

**Online Learning Resources**

1. **Codecademy**
    - Offers interactive Python courses that cover basic to advanced programming concepts, including data types, control structures, functions, and object-oriented programming.
    - Link: Codecademy Python
2. **Python.org**
    - The official Python website provides a comprehensive beginner's guide, documentation, and tutorials to get started with Python programming.
    - Link: Python Beginner's Guide
3. **GitHub Learning Lab**
    - An interactive learning platform where you can learn to code and collaborate on projects directly within GitHub, which is an essential tool for programmers.
    - Link: GitHub Learning Lab
4. **LeetCode**
    - Ideal for practicing Python coding skills through interactive coding challenges and problems, particularly useful for preparing for technical job interviews.
    - Link: LeetCode

# FUNDAMENTALS OF COMPUTER PROGRAMMING LAB

| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
|---|---|---|---|---|
| **Course Name: Fundamentals of Computer Programming lab** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCS151** | 0-0-2 | 1 | 30 |
| **Type of Course:** | Major-7 | | | |
| **Pre-requisites** | none | | | |

**Defined Course Outcomes**

| | |
|---|---|
| CO1 | **Demonstrating** proficiency in using Python's development environment and effectively utilize basic and advanced programming constructs to solve problems. |
| CO 2 | **Developing** Python programs that incorporate fundamental data structures, control flows, and functions to process and manipulate data |
| CO 3 | **Constructing** object-oriented Python applications demonstrating an understanding of classes, objects, inheritance, and polymorphism to solve complex problems |
| CO 4 | **Implementing** the effectiveness of algorithms involving searching, sorting, and recursion in Python, and visualize data using Python's libraries to communicate results clearly and effectively |

## LAB EXPERIMENTS

| **Ex. No** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| 1 | Write a Python function that takes two numbers as input and returns a string indicating whether the first number is greater than, less than, or equal to the second number using comparison operators. | CO1 |
| 2 | Create a Python program that functions as a calculator. It should support addition, subtraction, multiplication, division, and modulus operations. Maintain a history of calculations performed and provide an option to display the history. Implement error handling for division by zero. | CO1 |

|  |  |  |
|---|---|---|
| 3 | Develop a Python function that takes a list of tuples as input, where each tuple contains two numbers. For each tuple, compare the numbers using all comparison operators and store the results in a dictionary. Return a list of dictionaries for all comparisons. | CO1 |
| 4 | Write a Python program to simulate the operations of basic logic gates (AND, OR, NOT, NAND, NOR, XOR) using functions. The program should take two boolean inputs and the type of gate, then return the result of the logical operation. | CO1 |
| 5 | Create a Python function to check if any permutation of an input string is a palindrome. For example, the string "civic" is a palindrome, and "ivicc" is a permutation of "civic". The function should return True if any permutation is a palindrome, otherwise False. | CO2 |
| 6 | Develop a Python function that takes a string and a substring as input and returns the number of times the substring appears in the string. Implement this without using built-in string functions and optimize it for large inputs. | CO2 |
| 7 | Write a Python function that performs bitwise operations (AND, OR, XOR) on two arrays of integers. The function should take two arrays and an operator as input, perform the bitwise operation element-wise, and return the resulting array. | CO2 |
| 8 | Create a class CustomString that mimics some of the built-in string methods in Python. Implement methods like find(), replace(), split(), and join(). The class should be able to handle these operations efficiently and correctly. | CO3 |
| 9 | Create a Python function that finds all prime numbers up to a given number n. Use nested loops to check for primality and optimize the function to handle large inputs efficiently. Additionally, allow the user to break the loop prematurely if they input a specific command. | CO3 |
| 10 | Write a Python program that takes a list of tuples representing student names and grades, and sorts the list by grades using a lambda function. Ensure that if two students have the same grade, they are sorted by their names. | CO3 |
| 11 | Create a Python function that takes a list of integers and returns a new list containing only the unique elements of the original list. Use a set to remove duplicates and maintain the order of elements as they appear in the original list. | CO2 |
| 12 | Write a Python function that takes a string of text and returns a dictionary with the frequency of each word in the text. The function should handle punctuation and capitalization correctly and provide a case-insensitive count. | CO2 |
| 13 | Create a Python function that takes a tuple as an argument and returns a new tuple with the elements reversed. Demonstrate the use of tuples as function parameters and unpacking tuple elements within the function. | CO2 |

|  |  |  |
|---|---|---|
| 14 | Create a Python function that checks if a given string containing parentheses is balanced. The function should return True if the parentheses are balanced and False otherwise. Use a stack to solve this problem and handle multiple types of parentheses: (), {}, and []. | CO2 |
| 15 | Create a Python function that takes a list of strings and groups them into anagrams. The function should return a list of lists, where each sublist contains words that are anagrams of each other. Use dictionaries to store and group the anagrams efficiently | CO2 |
| 16 | Create a class BankAccount that represents a bank account with attributes account_number, account_holder, and balance. Implement methods for depositing, withdrawing, and checking the balance. Override the __str__ and __repr__ methods to provide meaningful string representations of the account. Include error handling to prevent overdrafts. | CO3 |
| 17 | Create a class Book with attributes title, author, and isbn. Implement a class Library that manages a collection of books. Implement methods to add, remove, and search for books by title or author. Use file handling to save and load the library collection from a file. | CO3 |
| 18 | Write a Python program that copies the contents of one file to another. Implement functions to open, read, write, and close files. Use exception handling to manage file errors, such as file not found or permission denied. | CO3 |
| 19 | Create a base class Employee with attributes name and employee_id. Derive classes Manager and Developer from Employee, each with additional attributes and methods specific to their roles. Override methods where necessary and use polymorphism to write a function that prints the details of all employees. | CO3 |
| 20 | Develop a class Item with attributes name, price, and quantity. Create a class ShoppingCart that manages a collection of items | CO3 |
| 21 | Write a Python program that encrypts and decrypts the contents of a file using a simple Caesar cipher. | CO3 |
| 22 | **Project Title: Data Visualization Dashboard using Flask** | CO4 |
| 23 | **COVID-19 Data Analysis and Visualization**<br>Create a Python application to analyze and visualize COVID-19 data using pandas, numpy, and matplotlib. | CO4 |
| 24 | **Project Title: Blog Web Application using Flask**<br>Create a blog web application using Flask that allows users to create, read, update, and delete blog posts | CO4 |
| 25 | **Project : Health and Fitness Tracker** | CO4 |

Develop a Health and Fitness Tracker that reads user data from a CSV file, including steps taken, calories burned, and heart rate.

**Online learning resources**

- **Codecademy**
    - Interactive coding platform that offers hands-on Python courses, teaching both the basics and more advanced topics in Python. Ideal for practicing specific programming tasks.
    - Link: Codecademy Python Course
- **HackerRank**
    - Provides a vast range of programming problems across various domains of computer science, along with a dedicated Python domain. Great for practicing coding skills and understanding algorithms.
    - Link: HackerRank Python
- **LeetCode**
    - Known for its extensive array of programming challenges that can help improve your understanding of data structures and algorithms. It's particularly good for preparing for technical job interviews.
    - Link: LeetCode
- **GitHub**
    - Not just a code repository, GitHub offers collaborative features and a wealth of open-source projects where students can engage in real-world software development and contribute to ongoing projects.
    - Link: GitHub

## Essentials of Computer Science

| | | | | |
|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Essentials of Computer Science** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **SEC067** | 0-0-0 | 2 | 40 |
| **Type of Course:** | SEC-1 | | | |
| **Pre-requisite(s):** NA | | | | |

**Course Perspective.** This course introduces students to the Fundamentals of Computer Science, a foundational area that underpins many modern technological principles and practices. The Fundamentals of Computer Science explores the essential concepts and techniques used in computing and programming, emphasizing both theoretical understanding and practical application. The course covers a range of topics from basic computer architecture and programming languages to advanced concepts like data structures, algorithms, software engineering, and web development. This comprehensive approach equips students with the skills necessary to analyze, model, and solve complex computing problems. The course is divided into 4 units:

a) Introduction to Computer Science and Programming Basics
b) Data Structures and Algorithms
c) Software Development and Engineering
d) Advanced Topics and Applications

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | Developing a basic understanding of the foundational principles and history of computer science. |
| **CO 2** | Developing a basic understanding of fundamental programming skills using languages such as Python, C++, and JavaScript. |
| **CO 3** | Developing a basic understanding of essential data structures and algorithms. |

|  |  |
|---|---|
| **CO 4** | Developing a basic understanding of software development and engineering principles, including web development, databases, and cybersecurity basics. |

**Course Outline:**

| **Unit Number: 1** | **Title: Introduction to Computer Science and Programming Basics** | **No. of hours: NA** |
|---|---|---|

**Content:**

- **Introduction to Computer Science**
  - Overview of computer science
  - History and impact of computing
  - Basic computer architecture
  - Number system & conversion
- **Basics of Programming**
  - Introduction to programming languages (Python, C, JavaScript)
  - Basic syntax and semantics
  - Writing and running simple programs
- **Problem-Solving Techniques**
  - Algorithms and pseudocode
  - Debugging and error handling
  - Basic problem-solving strategies
- **Basics of Windows and Linux Commands**
  - Introduction to operating systems
  - Basic Windows commands (e.g., dir, copy, del)
  - Basic Linux commands (e.g., ls, cp, rm)
  - File system navigation and management
  - Understanding working environments and command-line interfaces
- **Introduction to Networks**
  - Basics of computer networks
  - Network topologies and protocols
  - Introduction to the Internet and how it works

**Sources:**

- CS50's Introduction to Computer Science
- Computer Science 101 by Stanford University

- Introduction to Computer Science (Udemy)
- IT Fundamentals - Everything you need to know about IT (Udemy)
- Master Computer Fundamentals Course - Beginner to Intermediate (Udemy)

**Unit Number: 2 Title: Data Structures and Algorithms No. of hours: NA**

**Content:**

- **Basic Data Structures**
    - Arrays and lists
    - Stacks and queues
    - Linked lists
- **Algorithms**
    - Sorting algorithms (bubble sort, merge sort, quicksort)
    - Searching algorithms (linear search, binary search)
    - Algorithm analysis (time and space complexity)
- **Recursion**
    - Introduction to recursion
    - Recursive problem solving
    - Examples of recursive algorithms (e.g., factorial, Fibonacci sequence)

**Sources**:

- Fundamentals of Computing by Rice University
- CS50's Introduction to Computer Science
- Introduction to Computer Science (Udemy)
- Computer Science 101 by Stanford University

**Unit Number: 3 Title: Software Development and Engineering No. of hours: NA**

**Content:**

- **Software Development Lifecycle**
    - Requirements analysis
    - Design and architecture
    - Implementation and testing
- **Programming Paradigms**
    - Procedural programming
    - Object-oriented programming

    - Functional programming
- **Software Tools and Environment**
    - Integrated Development Environments (IDEs)
    - Version control systems (Git)
    - Debugging and profiling tools
- **Open-Source Tools**
    - Introduction to open-source tools and platforms
    - Using GitHub for version control and collaboration
    - Data science and machine learning with Kaggle
    - Other useful open-source tools (e.g., Jupyter Notebooks, Visual Studio Code)
- **Agile Methodologies**
    - Introduction to Agile principles
    - Scrum framework
    - Kanban and other Agile methodologies

**Sources**:

- CS50's Introduction to Computer Science
- Fundamentals of Computing by Rice University
- Computer Science 101 by Stanford University
- Computer Science 101 - Computers & Programming for Beginners (Udemy)
- IT Fundamentals - Everything you need to know about IT (Udemy)

**Unit Number: 4** **Title: Advanced Topics and Applications** **No. of hours: NA**

**Content:**

- **Web Development**
    - HTML, CSS, and JavaScript
    - Client-server architecture
    - Introduction to web frameworks
- **Databases**
    - SQL and relational databases
    - NoSQL databases
    - Basic database design and querying
- **Cybersecurity Basics**
    - Principles of cybersecurity
    - Common threats and vulnerabilities

    - Basic encryption and security protocols
- **Latest Technologies and Careers in Computer Science**
    - Overview of latest technologies (e.g., AI, blockchain, IoT, cloud computing)
    - Emerging domains in computer science
    - Various careers in computer science
    - Skills and qualifications needed for different career paths
- **Introduction to Machine Learning**
    - Basic concepts of machine learning
    - Types of machine learning (supervised, unsupervised, reinforcement learning)
    - Introduction to neural networks

**Sources**:

- CS50's Introduction to Computer Science
- Computer Science 101 by Stanford University
- Fundamentals of Computing by Rice University
- Introduction to Computer Science (Udemy)

**Learning Experience**

**Inside Classroom Learning Experience:**

- Interactive Tutorials and Quizzes: Pre-recorded video lectures with interactive quizzes at regular intervals to reinforce key programming and computer science concepts.
- Project-Based Learning: Learners will complete individual projects, such as developing a basic web application or implementing algorithms, with clear guidelines and milestones.
- Instructor Support and Feedback: Instructors will provide periodic feedback on assignments and be available for Q&A through virtual office hours or email support.

**Outside Classroom Learning Experience:**

- Self-Guided Coding Exercises: Students will engage in hands-on coding tasks in Python, C, and JavaScript through online platforms like Replit or CodeSandbox, with instant feedback and automated testing.
- Discussion Forums and Peer Reviews: Students will collaborate and share knowledge via discussion boards and peer reviews, fostering a supportive learning environment.

**Students can choose any one of the following online modules for certification.**

1. **CS50's Introduction to Computer Science by Harvard University**
    - Platform: Harvard Online Learning
    - Link: CS50's Introduction to Computer Science
2. **Computer Science 101 by Stanford University**
    - Platform: Stanford Online
    - Link: Computer Science 101
3. **Fundamentals of Computing by Rice University**
    - Platform: Coursera
    - Link: Fundamentals of Computing
4. **Introduction to Computer Science**
    - Platform: Udemy
    - Link: https://www.udemy.com/course/introduction-to-computer-science/
5. **Computer Science 101 - Computers & Programming for Beginners**
    - Platform: Udemy
    - Link: https://www.udemy.com/course/computer-science-101-computers-programming-for-beginners/
6. **IT Fundamentals - Everything you need to know about IT**
    - Platform: Udemy
    - Link: https://www.udemy.com/course/it-fundamentals-everything-you-need-to-know-about-it/
7. **Master Computer Fundamentals Course-Beginner to Intermediate**
    - Platform: Udemy
    - Link: https://www.udemy.com/course/master-computer-fundamentals-skills-beginner-to-intermediate/

**Please Note:**

1. **Enrollment**: Students must enroll in any one of the above specified courses on their respective platforms.
2. **Certification**: students will be required to submit the course completion certificate as an outcome.
3. **Self-paced**: Students will be required to complete any of the certifications on their own. No physical classes shall be conducted
4. **Assignments**: Complete all assignments, quizzes, and problem sets as required by each course.

## SEMESTER: II

## LINEAR ALGEBRA AND ORDINARY DIFFERENTIAL EQUATIONS

| | | | | |
|---|---|---|---|---|
| **Programme Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Linear Algebra and Ordinary Differential Equations** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENMA102** | 3-1-0 | 4 | 40 |
| **Type of Course:** | Major-8 | | | |
| **Pre-requisite(s):** Basic knowledge on Single variable calculus, Matrices, Differentiation and Integration | | | | |

**Course Perspective.** **T**his course aims to provide students with a thorough understanding of linear algebra and ordinary differential equations (ODEs), foundational mathematical tools essential in various fields of science and engineering. The course integrates theoretical knowledge with practical applications, enabling students to solve complex problems and understand systems that vary across time and space. The course is divided into 4 modules:

a) Matrices and Systems of Linear Equations
b) Eigenvalues and Eigenvectors
c) Vector Spaces
d) Ordinary Differential Equations

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Identifying** the properties of various types of matrices, such as symmetric, skew-symmetric, Hermitian, skew Hermitian, unitary, and orthogonal matrices. |
| **CO 2** | **Analyzing** quadratic forms and apply eigenvalues and eigenvectors in practical situations. |
| **CO 3** | **Defining** vector spaces, subspaces, linear independence, and basis. |
| **CO 4** | **Determining** the dimension of vector spaces and compute row space, column space, and null space of matrices. |

|  |  |
|---|---|
| **CO 5** | **Solving** first-order linear, separable, exact, and homogeneous differential equations. |

**Course Outline:**

| **Unit Number: 1** | **Title: Matrices and Systems of Linear Equations** | **No. of hours: 10** |
|---|---|---|
| **Content:** Matrix with operation, Types of Matrix (Symmetric and skew symmetric matrix, Hermitian and skew Hermitian matrix, unitary and orthogonal matrix), Determinant of Matrix, Inverse and transpose of matrices, Elementary row operations, Systems of Linear Equations, Homogeneous and non-homogeneous systems, Solutions of linear systems Gaussian, elimination and row echelon form, Rank of matrix. | | |
| **Unit Number: 2** | **Title: Eigenvalues and Eigenvectors** | **No. of hours: 10** |
| **Content:** Definition and properties of eigenvalues and eigenvectors, Diagonalization of matrices, Eigenvalues and eigenvectors of symmetric, skew symmetrix, hermition, skew hermition, unitary and orthogonal matrices, Calyey Hamilton Theorem, Rank and nullity of a matrix, Diagonalization of matrices, Minimal polynomial, characteristic polynomial, and generalized eigenvectors. The Jordan Normal Form Theorem for linear operators on a finite dimensional complex vector space, Quadratic forms, Applications of eigenvalues and eigenvectors. | | |
| **Unit Number: 3** | **Title: Vector Spaces** | **No. of hours: 10** |
| **Content:** Introduction to vector spaces**,** Subspaces and spanning sets**,** Linear independence and basis**,** Dimension of vector spaces**,** Row space, column space, and null space**,** Linear transformations**,** Matrix representation of linear transformations, Inner Product Spaces, Inner products and orthogonality, Orthonormal bases and Gram-Schmidt process, Orthogonal projections and least squares approximations, Applications of Linear Algebra, Markov chains and transition matrices. | | |
| **Unit Number: 4** | **Title: Ordinary Differential Equations** | **No. of hours: 10** |
| **Content:** Introduction to ordinary differential equations, Definition and classification of differential equations, First-order linear differential equations, Separable differential equations, Exact differential equations, Integrating factors, Applications of first-order differential equations, Second-order linear differential equations**,** Homogeneous differential equations**,** Method of undetermined coefficients**,** Variation of parameters**,** Applications of second-order differential equations | | |

**Learning Experiences**

**Inside Classroom Learning Experience**

1. **Interactive Lectures**: Use PPTs and visual aids to explain key concepts in linear algebra and differential equations.
2. **Conceptual Understanding**: Cover fundamental topics like matrix operations, eigenvalues, and solutions of ODEs.
3. **Problem-Solving Sessions**: Conduct in-class exercises on systems of equations and differential equations.
4. **Theory Assignments**: Assign theoretical problems with solutions discussed in class.
5. **Group Work**: Collaborate on problem-solving for real-world applications in engineering and science.
6. **Case Studies**: Analyze applications of linear algebra and differential equations in various fields.
7. **Continuous Feedback**: Implement in-class quizzes and feedback sessions to assess understanding.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home projects applying concepts to practical problems.
2. **Lab Projects**: Facilitate hands-on projects involving software tools for linear algebra and ODEs.
3. **Question Bank**: Provide practice problems and model papers for self-assessment.
4. **Online Forums**: Create platforms for students to discuss and collaborate on problems.
5. **Self-Study for Case Studies**: Encourage independent research on applications of linear algebra and ODEs.
6. **Collaborative Projects**: Organize group projects focused on modelling real-world phenomena using linear algebra and differential equations.

**Book Reference:**

- Strang, G. (2009). Introduction to Linear Algebra (4th ed.). Wellesley-Cambridge Press.
- Boyce, W. E., & DiPrima, R. C. (2017). Elementary Differential Equations and Boundary Value Problems (11th ed.). John Wiley & Sons.
- Tenenbaum, M., & Pollard, H. (1963). Ordinary Differential Equations. Dover Publications.

## OBJECT ORIENTED PROGRAMMING USING C++

| | | | | |
|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Object Oriented Programming using C++** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCS102** | 3-1-0 | 4 | 40 |
| **Type of Course:** | Major-9 | | | |
| **Pre-requisite(s), if any: Basics of C programming** | | | | |

**Course Perspective.** This course introduces students to the advanced principles and techniques of object-oriented programming (OOP) using C++. It is designed to build upon foundational programming knowledge, particularly for those who have a basic understanding of C programming. The course focuses on teaching students how to think about software development in an object-oriented way, enabling them to design and implement software solutions that are modular, extensible, and maintainable. The course is divided into 4 modules:

a) Foundations of Object-Oriented Programming
b) Classes, Objects, and Advanced Features
c) Inheritance, Polymorphism, and Software Engineering Principles
d) File Handling, Exception Management, and Unit Testing

**The Course Outcomes (COs).** On completion of the course the participants will be able to:

| **COs** | **Statements** |
|---|---|
| **CO 1** | **Understanding** the procedural and object-oriented paradigm with concepts of streams, classes, functions, data and objects. |
| **CO 2** | **Analyzing** dynamic memory management techniques using pointers, constructors, destructors, etc |
| **CO 3** | **Applying** the concept of function overloading, operator overloading, virtual functions and polymorphism |
| **CO 4** | **Classifying** inheritance with the understanding of early and late binding, usage of exception handling, file handling and generic programming. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| **Unit Number: 1** | **Title: Foundations of Object-Oriented Programming** | **No. of hours: 10** |
|---|---|---|
| **Content Summary:**<br>▪ **Programming Approaches:** Procedure-Oriented Approach vs. Object-Oriented Approach<br>▪ **Basic Concepts of Object-Oriented Programming:** Objects and Classes, Principles of OOP: Abstraction, Encapsulation, Inheritance, Polymorphism, Dynamic Binding and Message Passing<br>▪ **Characteristics of Object-Oriented Languages:** Benefits and features of OOP languages<br>▪ **Introduction to Object-Oriented Modelling Techniques:** Basic concepts of modeling in OOP | | |
| **Unit Number: 2** | **Title: Classes and Objects** | **No. of hours: 10** |
| **Content Summary:**<br>▪ **Abstract Data Types and Classes:** Concept of abstract data types, Objects and classes, attributes, and methods<br>▪ **C++ Class Declaration:** Declaring classes in C++, State, identity, and behavior of objects<br>▪ **Objects:** Local Objects and Global Objects, Scope resolution operator<br>▪ **Functions in C++:** Friend Functions, Inline Functions<br>▪ **Constructors and Destructors:** Instantiation of objects, Types of constructors (default, parameterized, copy), Static Class Data, Array of Objects, Constant member functions and objects<br>▪ **Memory Management Operators:** New and delete operators for dynamic memory allocation. | | |
| **Unit Number: 3** | **Title: Inheritance and Polymorphism** | **No. of hours: 10** |
| **Content Summary:**<br>▪ **Inheritance:** Types of inheritance (single, multiple, hierarchical, multilevel, hybrid), Access specifiers: public, private, and protected, Abstract Classes, Ambiguity resolution using scope resolution operator and virtual base class<br>▪ **Advanced Inheritance Concepts:** Aggregation and composition vs. classification hierarchy, Overriding inheritance methods<br>▪ **Polymorphism:** Types of Polymorphism (compile-time and run-time), Function Overloading, Operator Overloading<br>▪ **Pointers and Virtual Functions:** Pointer to objects, this pointer, Virtual Functions and pure virtual functions | | |

| | | |
|---|---|---|
| **Unit Number: 4** | **Title: Advanced C++ Features** | **No. of hours: 10** |
| **Content Summary:**<br>▪ **Strings and Streams:** Manipulating strings, Streams and file handling, File streams and string streams<br>▪ **Operators and Error Handling:** Overloading operators, Error handling during file operations, Formatted I/O<br>▪ **Generic Programming:** Function templates, Class templates<br>▪ **Exception Handling:** Throwing an exception, The try block, Catching an exception, Exception objects, Exception specifications, Rethrowing an exception, Catching all exceptions | | |

**Learning Experience**

**Inside Classroom Learning Experience**

1. **Interactive Lectures**: Use PPTs and live coding demonstrations to explain key OOP concepts.
2. **Conceptual Understanding**: Cover fundamental topics like classes, objects, inheritance, and polymorphism.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on implementing OOP principles in C++.
4. **Theory Assignments**: Assign programming problems that reinforce OOP concepts, discussed in class.
5. **Group Work**: Collaborate on projects that require designing and implementing class structures.
6. **Case Studies**: Analyze real-world applications of OOP in software development.
7. **Continuous Feedback**: Implement in-class quizzes and code reviews for ongoing assessment.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home projects emphasizing OOP design principles in C++.
2. **Lab Projects**: Facilitate hands-on programming tasks that apply OOP concepts to real-world scenarios.
3. **Question Bank**: Provide practice problems and resources for self-assessment.
4. **Online Forums**: Create platforms for students to discuss coding challenges and share solutions.
5. **Self-Study for Case Studies**: Encourage independent research on OOP best practices and design patterns.

6. **Collaborative Projects**: Organize group projects focused on developing software applications using OOP in C++.

**Text books:**

1. Robert Lafore, “Object-Oriented Programming in C++”, Sams Publishing, 4th Edition, 2004.
2. E. Balagurusamy, “Object-Oriented Programming with C++”, McGraw Hill Education, 6th Edition, 2017**.**

**Reference Book**

1. Schildt Herbert, “C++: The Complete Reference”, Wiley DreamTech, 2005.Parasons, “Object Oriented Programming with C++”, BPB Publication, 1999.
2. Steven C. Lawlor, “The Art of Programming Computer Science with C++”, Vikas Publication, 2002.
3. Yashwant Kanethkar, “Object Oriented Programming using C++”, BPB, 2004

**Additional Readings:**

**Online Learning**

**R 1. C++ Documentation on cppreference.com**

- A comprehensive reference that includes detailed documentation of C++ syntax, library functions, and features organized by version.
- **Link: cppreference.com**

# OBJECT ORIENTED PROGRAMMING USING C++ LAB

| | | | |
|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name: Object Oriented Programming using C++ Lab** | **Course Code** | **L-T-P** | **Credits** |
| | **ENCS152** | 0-0-2 | 1 |
| **Type of Course:** | Major -11 | | |

**Pre-requisite(s), if any: Basics of C programming**

**Defined Course Outcomes**

| COs | Statements |
|---|---|
| CO 1 | **Understanding** class object concepts by using C++. |
| CO 2 | **Developing** programs using inheritance and polymorphism. |
| CO 3 | **Demonstrating** the significance of constructors and destructor. |
| CO 4 | **Illustrating** generic classes using template concepts. |
| CO5 | **Implementing** the concept of file handling. |

## Lab Experiments

**Defined Course Outcomes**

| **Ex. No.** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| 1. | Implement a simple calculator in C++ that can perform basic arithmetic operations such as addition, subtraction, multiplication, and division. The program should prompt the user to enter two numbers and an operator, then display the result. Use appropriate data types and control structures to handle the calculations and validate user inputs. | CO 1 |
| 2 | Create a C++ program that checks if a given number is a prime number. The program should prompt the user to enter a number, then use control structures and functions to determine if the number is prime. Display an appropriate message indicating the result. | CO1 |

| | | |
|---|---|---|
| 3. | Implement a C++ program that sorts an array of integers using the bubble sort algorithm. The program should allow the user to input the array elements, then use a function to sort the array in ascending order. Display the sorted array as the output. | CO1 |
| 4. | Write a C++ program to demonstrate pointer arithmetic by creating an array of integers and using pointers to traverse and manipulate the array elements. Implement functions to calculate the sum, average, and maximum value of the array using pointer arithmetic. | CO1 |
| 5. | Write a C++ program to perform basic matrix operations such as addition, subtraction, and multiplication. Use two-dimensional arrays to represent the matrices and implement functions for each operation. Ensure the program handles matrices of appropriate sizes and displays the results accurately. | CO1 |
| 6. | Create a C++ program that generates the Fibonacci sequence up to a specified number of terms. Use a loop and control structures to generate the sequence and store the terms in an array. Display the generated sequence as the output. | CO1 |
| 7. | Write a C++ program to evaluate a string expression containing numbers, arithmetic operators (+, -, *, /), and parentheses. Implement a function that parses the expression and computes the result, considering operator precedence and parentheses. | CO1 |
| 8. | Write a C++ program to find all unique palindromic substrings in a given string. The function should take a string as input and return a set of strings containing all unique palindromic substrings. | CO1 |
| 9. | Create a class Rational to represent rational numbers with attributes numerator and denominator, implementing default, parameterized, and copy constructors, methods to add, subtract, multiply, and divide rational numbers, overloading the << and >> operators for input and output, and a friend function to compare two rational numbers. | CO2 |
| 10. | Create a class Matrix that represents a 2D matrix with dynamic memory allocation, implementing default, parameterized constructors, and a destructor, methods to add, subtract, and multiply matrices, overloading the [] operator to access matrix elements, and inline functions for basic matrix operations. | CO2 |

|  |  |  |
|---|---|---|
| 11. | Create a class Student with attributes studentID, name, and grades (an array of integers), implementing default, parameterized constructors, and a destructor, methods to calculate the average grade and display student details, using constant member functions to display details, and implementing dynamic memory allocation for the grades array. | CO2 |
| 12. | Create an abstract class Shape with a pure virtual function calculateArea(), deriving classes Circle, Rectangle, and Triangle each with attributes relevant to their shapes, implementing default and parameterized constructors, methods to calculate and display the area of each shape, and an array of Shape pointers to store different shapes and calculate their areas. | CO2 |
| 13. | Create a class InventoryItem with attributes itemID, itemName, and quantity, implementing default, parameterized, and copy constructors, methods to add, remove, and display inventory items, overloading the ++ and -- operators to increase and decrease item quantity, and implementing dynamic memory allocation for inventory items. | CO2 |
| 14. | Create a class Polynomial to represent a polynomial with dynamic memory allocation for coefficients, implementing default, parameterized constructors, and a destructor, methods to add, subtract, and multiply polynomials, overloading the +, -, and * operators for polynomial operations, and friend functions to input and output polynomials. | CO2 |
| 15. | Develop a Vehicle Management System that demonstrates different types of inheritance and polymorphism in C++. The system should manage various types of vehicles, including cars, trucks, and motorcycles, and should be able to perform operations such as adding new vehicles, displaying vehicle details, and comparing vehicles. | CO3 |
| 16. | Create a base class Account with methods deposit() and withdraw(). Derive classes SavingsAccount and CurrentAccount from Account. Overload the deposit() and withdraw() methods in the derived classes to include additional parameters like interest rate for SavingsAccount and overdraft limit for CurrentAccount. | CO3 |

|  |  |  |
|---|---|---|
| 17. | Create a class ComplexNumber to represent complex numbers. Implement operator overloading for +, -, *, and / operators to perform arithmetic operations on complex numbers. Use inheritance to extend the class with additional functionality for polar representation. | CO3 |
| 18. | Create a base class Animal with a virtual function makeSound(). Derive classes Dog and Cat from Animal, each implementing makeSound(). Write a function playWithAnimal() that takes a pointer to Animal and calls makeSound(). Demonstrate polymorphism by calling playWithAnimal() with pointers to Dog and Cat. | CO3 |
| 19. | Create a base class Person with attributes name and age. Derive classes Student and Teacher from Person. Further derive a class TeachingAssistant from both Student and Teacher. Use a virtual base class to avoid ambiguity in accessing attributes of Person. | CO3 |
| 20. | Create a base class Vehicle with attributes make and model, and methods start() and stop(). Derive classes Car, Truck, and Motorcycle from Vehicle. Use dynamic memory allocation (new and delete operators) to create and manage objects of these classes. Implement a function to display details of all vehicles. | CO3 |
| 21. | Write a C++ program that compresses a string using the counts of repeated characters. For example, the string "aabcccccaaa" would become "a2b1c5a3". If the "compressed" string would not become smaller than the original string, the function should return the original string. Use streams for efficient string manipulation. | CO4 |
| 22. | Write a template-based function in C++ to sort an array of any data type using the quicksort algorithm. Ensure the function works with different data types such as integers, floating-point numbers, and strings. | CO4 |
| 23. | Create a custom exception class InvalidInputException in C++ to handle invalid inputs. Implement a function that takes user input and throws an InvalidInputException if the input is not valid. Use try, catch, and throw blocks to handle the exception and display an appropriate error message. | CO4 |
| 24. | Write a C++ program that reads a text file, processes the text to remove punctuation, convert to lowercase, and count the frequency of each word. Use string streams for text manipulation and file streams for reading and writing files. | CO4 |

| | | |
|---|---|---|
| 25. | Implement a template-based stack class in C++ that supports basic stack operations such as push, pop, top, and isEmpty. Ensure the class works with different data types and includes appropriate exception handling for stack underflow and overflow. | CO4 |

# ENGINEERING CHEMISTRY

| | | | | |
|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name: ENGINEERING CHEMISTRY** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCH101** | 3-1-0 | 4 | 40 |
| **Type of Course:** | Major-10 | | | |
| **Pre-requisite(s), if any: Nil** | | | | |

**Course Perspective:** This course introduces students to the fundamental concepts and applications of chemistry in engineering. It is tailored specifically for engineering students to understand the chemical principles underlying various technological processes and materials essential in modern engineering. By exploring topics like water technology, chemical fuels, battery technology, and polymers, the course aims to provide students with a robust foundation in the chemical sciences that directly relates to their future fields of work. The course is divided into 4 modules:

a) Water technology
b) Chemical Fuels
c) Battery Technology
d) Polymer

**The Course Outcomes (COs).** On completion of the course the participants will be able to:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the methods for water hardness and alkalinity testing, and the basics of boiler water treatment. |
| **CO 2** | **Explaining** the process of dissolved oxygen determination and chemical oxygen demand analysis. |
| **CO 3** | **Determining** various methods to enhance the quantity & quality of Fuel. |
| **CO 4** | **Identifying** between hard and soft water, solve the related numerical problems on water purification and its significance in industry and daily life. |
| **CO 5** | **Articulating** basic concepts of chemistry in daily life. |

| | |
|---|---|
| **CO 6** | **Designing** efficient process for water analysis and purification |

**Course Outline:**

**Unit Number: 1 Title: Water technology No. of hours: 10**

**Content:**

**Introduction to Water Technology:** Importance and applications of water in various industries.

**Water Analysis:** Hardness: Determination by EDTA method, Alkalinity: Determination by double indicator method.

**Treatment of Boiler Feed Water**

Internal Treatment: Phosphate conditioning, Colloidal conditioning, Calgon conditioning

External Treatment: on exchange process, Lime-soda process, Zeolite process

**Determination of Dissolved Gases:** Dissolved oxygen: Determination by Winkler's method, Chemical oxygen demand: Determination.

**Boiler Scales Formation and Prevention:** Formation and ill effects of boiler scales., Methods of prevention of scales.

**Numerical Problems:** Calculations related to water analysis and treatments.

**Unit Number: 2 Title: Chemical Fuels No. of hours: 10**

**Content:**

**Fuels**: Introduction, classification, calorific value (HCV & LCV), Determination of calorific value of fuel using Bomb calorimeter.

**Solid fuel**: Coal- its analysis by proximate and ultimate analysis, Numerical problems.

**Liquid fuels**: Refining of petroleum, Petroleum cracking, Reformation of petrol-explanation with reactions, Knocking in IC engine, its ill effects and prevention of knocking. Anti-knocking agent: Leaded and unleaded petrol. Power alcohol and its advantages. Synthetic petrol - Bergius process.

**Gaseous fuels**: LPG, CNG and their applications.

**Unit Number: 3 Title: Battery Technology No. of hours: 10**

**Content:**

**Introduction to Battery Technology:** Galvanic cell, Electrode potential, EMF of the cell, Cell representation.

**Batteries and Their Importance:** Classification of batteries: Primary, Secondary, and Reserve batteries. Examples of each type.

**Battery Characteristics:** Voltage, Capacity, Energy density, Power density, Energy efficiency, Cycle life, Shelf life.

**Commercial Batteries:** Basic requirements for commercial batteries.

**Construction, Working, and Applications:** Ni-Cd battery, Lithium-ion battery.

**Fuel Cells: Differences between batteries and fuel cells.** Classification of fuel cells based on: Type of fuel, Electrolyte, Temperature.

**Unit Number: 4** **Title: Polymer** **No. of hours: 10**

**Content:**

**Basic Concepts of Polymers:** Definition and types of polymers.

**Types of Polymers:** Thermoplastic polymers, Thermosetting plastics.

**Preparation and Applications of Industrially Important Polymers:** Natural rubber, Buna S, Buna-N, Neoprene, Isoprene, Nylon-6, Nylon-6,6, Dacron, Terylene.

**Advanced Polymers:** Conducting polymers, Biodegradable polymers.

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures**: Use PPTs and demonstrations to explain key chemistry concepts relevant to engineering.
2. **Conceptual Understanding**: Cover fundamental topics like thermodynamics, kinetics, and material science.
3. **Problem-Solving Sessions**: Conduct in-class exercises on chemical calculations and reactions.
4. **Theory Assignments**: Assign theoretical problems, with solutions discussed in class.
5. **Group Work**: Collaborate on projects involving chemical processes and materials.
6. **Case Studies**: Analyze real-world applications of chemistry in engineering fields.
7. **Continuous Feedback**: Implement in-class quizzes and feedback sessions to assess understanding.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home projects applying chemistry concepts to engineering challenges.
2. **Lab Projects**: Facilitate hands-on experiments that explore chemical principles in practical applications.
3. **Question Bank**: Provide practice problems and model papers for self-assessment.
4. **Online Forums**: Create platforms for students to discuss and collaborate on chemistry problems.

5. **Self-Study for Case Studies**: Encourage independent research on recent advancements in engineering chemistry.
6. **Collaborative Projects**: Organize group projects focused on developing sustainable chemical processes or materials.

**Text Books**

**T1:** Principles of Physical Chemistry by B. R. Puri, L. R. Sharma and M. S. Pathania, S. Nagin Chand and Co.

**T2:** Physical Chemistry by Soni and Dharmatha, S. Chand & Sons.

**T3:** Polymers science by Gowarikar and Vishwanathan.

**Reference Books:**

**R 1.** Corrosion Engineering by M. G. Fontana, Mc Graw Hill Publications.

**R 2.** Engineering Chemistry by Jain and Jain.

**Additional Readings:**

**Basics of electrochemistry:**

https://mrcet.com/downloads/digital_notes/HS/4%20ENGINEERING%20CHEMISTRY.pdf

**Basics of polymer:**

https://gnindia.dronacharya.info/APS/Downloads/SubjectInformation/Chemistry/Unit2/Lecture_1_13022019.pdf

# ENGINEERING CHEMISTRY LAB

| | | | |
|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **ENGINEERING CHEMISTRY LAB** | **ENCH151** | 0-0-2 | 1 |
| **Type of Course:** | Major-12 | | |

**Defined Course Outcomes**

| | |
|---|---|
| CO1 | **Applying** various experimental techniques commonly used in chemistry labs, such as titrations, distillations, extractions, chromatography, spectroscopy, and electrochemical methods. |
| CO2 | **Acquiring** proficiency in handling and operating laboratory equipment, including but not limited to balances, pipettes, burettes, spectrophotometers, pH meters, and other analytical instruments. |
| CO3 | **Developing** skills in recording and analyzing experimental data, including data interpretation of results. |
| CO4 | **Understanding** hands-on experience in synthesizing various chemical compounds and organic polymers |
| CO5 | **Illustrating** to write **concise** and accurate laboratory reports, including experimental procedures, observations, results, and conclusions. |
| CO6 | **Understanding** the ethical responsibilities and laboratory safety protocols associated with conducting experiments. |

## Lab Experiments

| **Ex. No** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| 1 | Determination of temporary and permanent hardness in water sample using EDTA. | CO1, CO3, CO5 |
| 2 | Determination of alkalinity in the given water sample. | CO1, CO3, CO5 |
| 3 | Determination of viscosity of given liquid. | CO2, CO3, CO5 |
| 4 | Determination of surface tension of given liquid. | CO2, CO3, CO5 |
| 5 | Determination of pH by pH-metric titration. | CO1, CO3, CO5 |

| | | |
|---|---|---|
| 6 | Preparation of Phenol-formaldehyde and Urea-formaldehyde resin | CO4, CO5, CO6 |
| 7 | To determine the iron concentration in the given water sample by Spectrophotometer using potassium thiocyanate as colour developing agent. | CO1, CO3, CO5 |
| 8 | Determination of chloride content in water sample. | CO1, CO3 CO5, CO6 |
| 9 | Estimation dissolved oxygen (DO) content in the given water sample by Winkler's method. | CO1, CO3, CO5 |
| 10 | Determination of iron content in the given solution by Mohr's method. | CO1, CO3, CO5 |
| 11 | Determination of rate constant of hydrolysis of esters. | CO3, CO5 |
| 12 | To determine the Iron content in the given salt by using external indicator | CO1, CO3, CO5 |
| 13 | Determination of wavelength of absorption maximum and colorimetric estimation of Fe3+ in solution | CO2, CO3, CO5 |
| 14 | Determination of molar absorptivity of a compound (KMnO4 or any water-soluble food colorant). | CO2, CO3, CO5 |
| 15 | Preparation of a nickel complex [Ni(NH3)6]Cl2 and estimation of nickel by complexometric titration. | CO4, CO5, CO6 |
| 16 | Synthesis of drug like Aspirin, /Paracetamol etc. | CO4, CO5, CO6 |

## ENGINEERING DRAWING & WORKSHOP LAB

| | | | |
|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **Engineering Drawing and Workshop Lab** | **SEC033** | 0-0-4 | 2 |
| **Type of Course:** | SEC-2 | | |

**Pre-requisite(s), if any: Basic geometry and familiarity with tools.**

**Defined Course Outcomes**

| COs | **Statements** |
|---|---|
| CO1 | **Understanding** the polygons, circles and lines with different geometric conditions |
| CO2 | **Drawing** the projection of points, lines and planes under different conditions and orthographic views from isometric views of simple objects |
| CO3 | **Determining** manufacturing methods in different fields of engineering and Practical exposure to different fabrication techniques |
| CO4 | **Creating** of simple components using different materials |
| CO5 | **Exposing** to some of the advanced and latest manufacturing techniques being employed in the industry. |

## LAB EXPERIMENTS

| **EX. NO.** | **EXPERIMENT TITLE** | **Mapped CO/COs** |
|---|---|---|
| **Engineering Graphics** | | |
| 1 | Manual drafting of basic geometric constructions and shapes using set squares, and compass. | CO1 |
| 2 | Understand and draw different projections, apply to create points and lines, in all quadrants. | CO1 |
| 3 | Draw orthographic projections of simple objects like cubes, cylinders, and prisms. | CO2 |
| 4 | Create isometric drawings of simple assemblies. | CO1 |
| 5 | Introduction to CAD System & AutoCAD and understand basic commands. | CO3 |
| 6 | Use AutoCAD to recreate the manually drawn orthographic projections. | CO3 |
| 7 | Create similar drawings using an open-source tool like LibreCAD. | CO5 |

| | | |
|---|---|---|
| 8 | Model simple objects (like a nut and bolt) in 3D using AutoCAD or similar software. | CO1 |
| 9 | Draw and assemble a small mechanical device (like a piston or gear assembly) using CAD software | CO2 |
| 10 | Design and draw an entire machine or a significant part of it, incorporating all the skills learned (Mini Project) | CO3 |
| | **Workshop Technology** | |
| 1 | Demonstrate safe handling and use of various hand tools and power tools. | CO2 |
| 2 | File, saw, and drill a metal piece to create a simple object such as a fitting job. | CO3 |
| 3 | Create a joint or assemble parts using hand tools, ensuring tight fit and proper alignment. | CO4 |
| 4 | Perform simple welding and brazing tasks to join metal pieces. | CO5 |
| 5 | Design and create a shaft on lathe machine using MS Rod. | CO2 |
| 6 | Design and create a flat V Job on Shaper machine using MS Block. | CO3 |
| 7 | Design and manufacture a sheet metal tray, In sheet Metal Shop. | CO3 |
| 8 | Measure, cut, and assemble wooden parts to create a simple structure, such as frame or T-Joint. | CO5 |
| 9 | Ability to program basic CNC machine operations and understand CNC machining processes. | CO1 |
| 10 | Design and 3D print a small part or model using CAD software. | CO3 |

**Inside Classroom Learning Experience:**

1. **Interactive Demonstrations**: Hands-on demonstrations for drawing geometric constructions and projections using manual drafting tools.
2. **CAD Software Tutorials**: In-class sessions to learn basic commands and operations in CAD software like AutoCAD or LibreCAD.
3. **Problem-Solving Sessions**: Exercises on creating orthographic projections and isometric views of various objects, discussed and solved in class.
4. **Workshop Practice**: Supervised sessions to use workshop tools and machines, such as lathes, shapers, and CNC machines, with live demonstrations.
5. **Group Work**: Collaborative activities for assembling small mechanical devices or creating 3D models using CAD software.

6. **Mini-Projects**: Students work on mini-projects, applying all learned skills to design and draw complete machine components or parts.
7. **Continuous Feedback**: In-class quizzes and frequent assessments on drafting techniques and workshop safety practices.

**Outside Classroom Learning Experience:**

1. **Take-home Drafting Assignments**: Practice drawing projections and creating geometric shapes at home using manual drafting tools.
2. **CAD Practice**: Students work on additional exercises to enhance their CAD skills, creating complex models outside of class.
3. **Online Forums**: Participate in discussions or forums to troubleshoot CAD and workshop-related queries with peers and instructors.
4. **Independent Workshop Projects**: Apply workshop techniques learned in class to create simple objects at home or during independent lab hours.
5. **Self-Study on Manufacturing Methods**: Research the latest trends and methods in manufacturing, like CNC machining or 3D printing, and relate them to class content.
6. **Collaborative Projects:** Work in groups to design and 3D print small parts or models using CAD software, applying both theoretical and practical knowledge.

# Applied Generative AI: Practical Tools and Techniques

| | | | |
|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **Applied Generative AI: Practical Tools and Techniques for the Modern Professional** | | 0-0-4 | 2 |
| **Type of Course:** | SEC-3 | | |

**Defined Course Outcomes**

| ***COs*** | ***Statements*** |
|---|---|
| ***CO1*** | **Applying** basic functionalities of Hugging Face and LangChain to generate text-based applications and automate simple tasks |
| ***CO2*** | **Analyzing** ethical dilemmas and create basic data visualizations using GenAI tools, automating standard business communications and financial predictions. |
| ***CO3*** | **Creating** roleplaying chatbots, automate market insights extraction, and generate social media content using GenAI. |
| ***CO4*** | **Evaluating** advanced GenAI models for automating complex tasks, generating interactive visualizations, and conducting ethical analyses. |

**LAB EXPERIMENTS**

| **Ex. No.** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| **1.** | **Experiment 1: Introduction to Generative AI: Overview of Generative AI, Hugging Face, and LangChain.**<br>a. Explore basic functionalities of Hugging Face and LangChain by creating a simple text generation application.<br>b. Learn to create simple prompts for GenAI models to generate various types of text outputs.<br>c. Automate tasks such as scheduling and data entry using GenAI.<br>d. Perform basic data analysis and generate summary reports with GenAI.<br>e. Generate automated content such as emails and reports using GenAI. | **CO1** |

| | | |
|---|---|---|
| **2.** | **Experiment 2: Ethical Considerations and Data Visualization**<br>a. Create a presentation or report outlining ethical issues and potential solutions.<br>b. Develop scripts for generating data visualizations like bar charts and pie charts using GenAI.<br>c. Automate business communications such as appointment reminders<br>d. Use GenAI for financial predictions based on historical data<br>**e.** Plan and manage tasks using GenAI for project scheduling | **CO2** |
| **3.** | **Experiment 3: Advanced GenAI Applications and Customer Interaction**<br>a. Create a chatbot to handle basic customer queries.<br>b. Automate market insights extraction with GenAI.<br>c. Generate presentations and reports using GenAI.<br>d. Develop roleplaying chatbots for customer service training.<br>**e.** Generate social media posts and content using GenAI. | **CO3** |
| **4.** | **Complex Applications and Ethical Frameworks**<br>**Experiment 4: Explore Advanced GenAI Models and Their Applications in Various Industries**<br>Explore Advanced GenAI Models and Their Applications in Various Industries. Explore advanced Generative AI models such as GPT-4, DALL-E, and BERT. Develop a comprehensive report or presentation detailing these models and their potential uses in various industries, including healthcare, finance, marketing, and customer service. Example models like GPT-4 (OpenAI), DALL-E (OpenAI), BERT (Google), T5 (Google), and CLIP (OpenAI) will be covered. The outcome will be a thorough understanding of how these models can be applied to natural language processing, image generation, conversational agents, and automated content creation. | **CO4** |
| **5.** | **Project 1: Intelligent Email Assistant**<br>**Problem Statement:** Develop an intelligent email assistant that uses Hugging Face and LangChain to draft, respond to, and organize emails. This project aims to streamline email management for professionals by leveraging generative AI tools. | CO1,CO2, CO3,CO4 |
| **6.** | **Project 2: Social Media Content Generator** | CO1,CO2, CO3,CO4 |

|  |  |  |
|---|---|---|
|  | **Problem Statement:** Design a social media content generator that uses generative AI models to create posts, captions, and hashtags for different platforms. This project will help social media managers generate engaging content efficiently. |  |
| **7.** | **Project 3: Ethical AI Implementation Framework for Healthcare**<br>**Problem Statement:** Develop a comprehensive ethical AI implementation framework for healthcare organizations to ensure the responsible use of generative AI in medical applications. This project addresses the ethical challenges and ensures that AI is used in a fair, transparent, and accountable manner. | CO1,CO2, CO3,CO4 |
| **8.** | **Project 4: Financial Report Generation System**<br>**Problem Statement:** Create a financial report generation system that uses generative AI models to analyze financial data and generate comprehensive reports. This project will assist financial analysts in making informed decisions based on accurate and data-driven insights. | CO1,CO2, CO3,CO4 |

**Learning Experiences**

**Inside Classroom Learning Experience:**

1. **Interactive Lectures**: Introduce key concepts in generative AI using PPTs and live demonstrations.
2. **Conceptual Understanding**: Cover topics like neural networks, GANs, and prompt engineering.
3. **Problem-Solving Sessions**: Conduct in-class exercises to build and refine generative models.
4. **Theory Assignments**: Assign projects applying generative AI techniques, discussed in class.
5. **Group Work**: Collaborate on projects utilizing generative AI tools for creative applications.
6. **Case Studies**: Analyze real-world applications of generative AI across industries.
7. **Continuous Feedback**: Implement quizzes and peer reviews to assess understanding.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home projects applying generative AI techniques to practical scenarios.
2. **Lab Projects**: Facilitate hands-on activities using generative AI tools to create models and content.
3. **Question Bank**: Provide practice problems and resources for self-assessment.
4. **Online Forums**: Create platforms for discussing generative AI challenges and solutions.
5. **Self-Study for Case Studies**: Encourage independent research on advancements in generative AI.
6. **Collaborative Projects**: Organize group projects focused on innovative applications of generative AI.

**Text Books:**

- "Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play" by David Foster
- "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron

**Online References**

**General Introduction to Generative AI**

1. **OpenAI Blog**: Articles and research updates on advancements in AI - OpenAI Blog
2. **DeepMind Publications**: Research and analysis on the latest AI technologies - DeepMind Research

**Tools and Libraries**

1. **Hugging Face Documentation**: Comprehensive guide and API references for using transformer models - Hugging Face Docs
2. **LangChain Documentation**: Tools and libraries for building language applications - LangChain GitHub

**Ethical Frameworks for AI**

1. **AI Ethics Guidelines by the European Commission**: Framework for trustworthy AI - Ethics Guidelines for Trustworthy AI
2. **Partnership on AI**: Research and partnership initiatives on AI ethics - Partnership on AI

**Prompt Engineering and Usage**

3. **Practical Prompt Engineering Guide by OpenAI**: Guidelines on effective prompt engineering - Prompt Engineering with OpenAI
4. **Prompt Engineering Workshop**: Online courses and tutorials on prompt engineering - Prompt Engineering Course

# Minor Project-I

| | | | |
|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | |
| **COURSE NAME:** **Minor Project-I** | **COURSE CODE** | **L-T-P** | **CREDITS** |
| | **ENSI152** | 0-0-0 | 2 |
| **TYPE OF COURSE:** | PROJ-1 | | |

**PRE-REQUISITE(S), IF ANY:** NA

**Course Perspective:**
The objective of Minor Project-I for the B. Tech (Computer Science and Engineering) program is to provide students with the opportunity to apply theoretical knowledge to real-world societal problems. This course aims to develop students' ability to identify and understand complex societal issues relevant to computer science, engage in critical thinking to formulate and analyze problems, and conduct comprehensive literature reviews to evaluate existing solutions. Through this project, students will enhance their research skills, document their findings in a well-structured manner, and effectively present their analysis and conclusions. The course fosters professional development by encouraging students to approach problems from multiple perspectives, develop innovative solutions, and improve their communication and documentation skills. Ultimately, the Minor Project-I course seeks to prepare students for future professional challenges by integrating academic knowledge with practical problem-solving experiences.
**Duration:** 6 weeks.
Project must focus on following aspects:
**Project Requirements:**

1. **Understanding of Societal Problems:**
   - Students must have a basic understanding of societal problems, the concerned domain, and relevant issues.
2. **Critical Thinking and Problem Formulation:**
   - Students are expected to think critically about formulated problems and review existing solutions.
3. **Presentation of Findings:**
   - Students must be able to present findings from existing solutions in an appropriate format.
4. **Implementation:**
   - Students are not strictly expected to provide or implement these existing solutions.

**Guidelines:**

1. **Project Selection:**
    - Choose a societal problem relevant to the field of computer science and engineering.
    - Ensure the problem is specific and well-defined.
2. **Literature Review:**
    - Conduct a thorough review of existing literature and solutions related to the problem.
    - Identify gaps in existing solutions and potential areas for further investigation.
3. **Analysis and Critical Thinking:**
    - Analyze the problem critically, considering various perspectives and implications.
    - Evaluate the effectiveness and limitations of current solutions.
4. **Documentation:**
    - Document the entire process, including problem identification, literature review, analysis, and findings.
    - Use appropriate formats and standards for documentation.
5. **Presentation:**
    - Prepare a presentation summarizing the problem, existing solutions, analysis, and findings.
    - Ensure the presentation is clear, concise, and well-structured.

**Evaluation Criteria for Minor Project (Out of 100 Marks):**

1. **Understanding of Societal Problems (20 Marks):**
    - Comprehensive understanding of the problem: 20 marks
    - Good understanding of the problem: 15 marks
    - Basic understanding of the problem: 10 marks
    - Poor understanding of the problem: 5 marks
    - No understanding of the problem: 0 marks
2. **Critical Thinking and Analysis (30 Marks):**
    - Exceptional critical thinking and analysis: 30 marks
    - Good critical thinking and analysis: 25 marks
    - Moderate critical thinking and analysis: 20 marks
    - Basic critical thinking and analysis: 10 marks
    - Poor critical thinking and analysis: 5 marks
    - No critical thinking and analysis: 0 marks
3. **Literature Review (20 Marks):**
    - Comprehensive and detailed literature review: 20 marks

    - Good literature review: 15 marks
    - Moderate literature review: 10 marks
    - Basic literature review: 5 marks
    - Poor literature review: 0 marks
4. **Documentation Quality (15 Marks):**
    - Well-structured and detailed documentation: 15 marks
    - Moderately structured documentation: 10 marks
    - Poorly structured documentation: 5 marks
    - No documentation: 0 marks
5. **Presentation (15 Marks):**
    - Clear, concise, and engaging presentation: 15 marks
    - Clear but less engaging presentation: 10 marks
    - Somewhat clear and engaging presentation: 5 marks
    - Unclear and disengaging presentation: 0 marks

**Total: 100 Marks**

**Course Outcomes:**

By the end of this course, students will be able to:

1. **Understand Societal Issues:**
    - Demonstrate a basic understanding of societal problems and relevant issues within the concerned domain.
2. **Critical Thinking:**
    - Think critically about formulated problems and existing solutions.
3. **Literature Review:**
    - Conduct comprehensive literature reviews and identify gaps in existing solutions.
4. **Documentation:**
    - Document findings and analysis in a well-structured and appropriate format.
5. **Presentation Skills:**
    - Present findings and analysis effectively, using clear and concise communication skills.
6. **Problem Analysis:**
    - Analyze problems from various perspectives and evaluate the effectiveness of existing solutions.
7. **Professional Development:**
    - Develop skills in research, analysis, documentation, and presentation, contributing to overall professional growth.

**Learning Experiences**

- Real-World Application: Students will apply theoretical knowledge to analyze societal problems, gaining hands-on experience in tackling real-world issues related to computer science.
- Critical Thinking and Problem Solving: By identifying, formulating, and evaluating complex problems, students will enhance their critical thinking and analytical skills.
- Research Skills: Students will conduct comprehensive literature reviews, learning to assess existing solutions and identify research gaps for future exploration.
- Effective Communication: Through structured documentation and presentations, students will develop clear and concise communication skills essential for professional settings.
- Multi-Perspective Analysis: Students will learn to evaluate problems from diverse perspectives, fostering innovative thinking and problem-solving abilities.
- Professional Development: The project encourages research, analysis, and presentation skills, preparing students for future professional challenges in the tech industry.

## SEMESTER: III

## JAVA PROGRAMMING

**Program Name** **B. Tech (Computer Science and Engineering)**

| **COURSE NAME: JAVA PROGRAMMING** | **COURSE CODE** | **L-T-P** | **CREDITS** | **Contact Hours** |
|---|---|---|---|---|
| | **ENCS201** | 4-0-0 | 4 | 40 |
| **TYPE OF COURSE:** | Major-13 | | | |

**PRE-REQUISITE(S), IF ANY:** C PROGRAMMING

**Course Perspective.** This course provides a comprehensive introduction to Java, one of the most popular and widely used programming languages in the world, particularly known for its portability across platforms from mainframe data centers to smartphones. The "Java Programming" course is meticulously designed to introduce students to the core concepts of object-oriented programming using Java, covering everything from basic constructs to advanced programming features. The curriculum is structured to not only impart theoretical knowledge but also to enhance practical skills through extensive lab sessions, thereby preparing students for real-world software development. The course is divided into 4 modules:

a) Introduction to Java and OOP
b) Inheritance and Polymorphism (Abstract Class, Packages, and Interfaces)
c) Exception Handling, Multithreading and Wrapper Class
d) I/O Stream, File Handling, and Collections

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Applying** Java fundamentals and basic constructs to write Java programs. |

| | |
|---|---|
| **CO 2** | **Designing** object-oriented solutions using classes, objects, inheritance, and polymorphism. |
| **CO 3** | **Utilizing** interfaces and packages for code structure and reusability. |
| **CO 4** | **Implementing** error handling with try-catch-finally and custom exceptions. |
| **CO 5** | **Designing** multithreaded applications using synchronization. |
| **CO 6** | **Performing** file I/O, work with Java Collections Framework, and manipulate data using collections |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number: 1** | **Title: Introduction to Java and OOP** | **No. of hours: 10** |

**Content:**

**Introduction to Java** –

Features, and Importance, Java Virtual Machine, Byte Code; Keywords, constants, variables and Data Types, Operators and Expressions, Type casting and conversion;

**Java Control Structure -** Decision making – if, if-else, if-else-if ladder, nested if, switch-case, Loop – do, while, for, jump statements – break and continue;

**Simple Input and Output -** Scanner Class; Arrays Handling - Single and Multi-dimensional, Referencing Arrays Dynamically;

**Java Strings**: String class, Creating & Using String Objects, Manipulating Strings, String Immutability & Equality, Passing Strings To & From Methods.

**OOP Paradigm**: Features of OOP, Class and Object in Java: Creating Classes and Objects. Defining Data Members and Member Methods, Overloading Member Methods, Static Members, this Keyword. Constructors: default, parameterized and copy constructors.

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: Inheritance and Polymorphism (Abstract Class, Packages, and Interfaces)** | **No. of hours: 10** |

**Content**:

**Access Specifies, Introduction to Inheritance** – Derived Class and Super class, super Keyword;

**Types of inheritance** – simple, multilevel, multilevel, hierarchical, and hybrid;

**Polymorphism** – Static (Method overloading), Dynamic (Method Overriding);

Final Class and Method, finalize keyword, Garbage Collection;

**Abstract Method and Abstract Class**. Interfaces **-** Defining an Interface, Implementing an Interface;

**Packages -** Creating Package, Naming a Package, Using Package Members, Extending Interfaces and Packages, Package and Class Visibility.

| | | |
|---|---|---|
| **Unit Number: 3** | **Title: Exception Handling, Multithreading and Wrapper Class** | **No. of hours: 10** |

**Content:**

**Exception Handling -** Definition, Dealing with Errors, The Classification of Exceptions, Declaring Checked Exceptions, Throw an Exception, Creating Exception Classes, Catching Exceptions, finally clause;

**Multithreaded Programming -** Fundamentals, Java thread model: priorities, synchronization, messaging, thread classes, Runnable interface, inter thread Communication, suspending, resuming, and stopping threads.

**Wrapper Classes -** Autoboxing/Unboxing, Enumerations.

| | | |
|---|---|---|
| **Unit Number: 4** | **Title: I/O Stream, File Handling, and Collections** | **No. of hours: 10** |

**Content:**

**File Handling**: File Class Methods, Reading from a File, Writing to a File, Buffered I/O, Character Streams, Byte Streams, File Input/Output Stream, FileReader, FileWriter, BufferedWriter, BufferedReader, FileInputStream, FileOutputStream, File Navigation, File Permissions, Directory Operations, File and Directory Attributes

**Java Collections Framework**: Introduction to Java Collections Framework

**Collection Interfaces**: List (ArrayList, LinkedList, Vector), Set (HashSet, LinkedHashSet, TreeSet), Queue (PriorityQueue), Map (HashMap, LinkedHashMap, TreeMap), Iterators, Comparable and Comparator Interfaces, Sorting Collections, Generics in Collections

**Working with Collections**: Adding, Removing, Searching Elements, Iterating Elements

**Learning Experiences**

**Classroom Learning Experience**

1. **Interactive Lectures**: Introduce key Java concepts using PPTs and live coding demonstrations.
2. **Conceptual Understanding**: Cover topics like object-oriented programming, exception handling, and Java APIs.

3. **Problem-Solving Sessions**: Conduct in-class exercises focused on coding challenges and algorithm implementation.
4. **Theory Assignments**: Assign programming problems that reinforce Java concepts, discussed in class.
5. **Group Work**: Collaborate on projects requiring teamwork in Java application development.
6. **Case Studies**: Analyze real-world applications of Java in software development and enterprise solutions.
7. **Continuous Feedback**: Implement quizzes and code reviews to provide ongoing assessment and improvement.
8.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home projects applying Java programming concepts to practical problems.
2. **Lab Projects**: Facilitate hands-on programming tasks that apply Java in real-world scenarios.
3. **Question Bank**: Provide practice problems and resources for self-assessment in Java.
4. **Online Forums**: Create platforms for discussing Java programming challenges and solutions.
5. **Self-Study for Case Studies**: Encourage independent research on Java best practices and frameworks.

- **Collaborative Projects**: Organize group projects focused on developing Java applications and systems.

**Text Book**

1. Herbert Schildt, ―java – the complete reference‖, oracle press.
2. Cay s. Horstmann, ―core java volume – i fundamentals‖, pearson.

**Additional Readings:**

**Online Learning Resources**

1. **Oracle Java Tutorials**
   - The official tutorials from Oracle, which owns Java, are a great starting point. These cover the basics and advanced features of Java.
   - Link: Oracle Java Tutorials
2. **Codecademy**
   - Codecademy offers an interactive Java programming course that includes exercises and projects to help beginners understand Java from scratch.
   - Link: Codecademy Java Course

3. **Java Code Geeks**
    - A community-driven site that offers free Java tutorials, articles, and examples. It's a valuable resource for practical tips and best practices.
    - Link: Java Code Geeks
4. **LeetCode**
    - Excellent for practicing Java coding problems, LeetCode helps in enhancing problem-solving skills in Java, which is crucial for technical interviews.
    - Link: LeetCode

# JAVA PROGRAMMING LAB

| | |
|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** |

| **COURSE NAME:** | **COURSE CODE** | **L-T-P** | **CREDITS** |
|---|---|---|---|
| **JAVA PROGRAMMING LAB** | **ENCS251** | 0-0-2 | 1 |

**TYPE OF COURSE:** Major-16

**PRE-REQUISITE(S), IF ANY:** basic working knowledge of C++ programming will be an added advantage

**DEFINED COURSE OUTCOMES**

| **COS** | |
|---|---|
| **CO 1** | **Demonstrating** the use of primitive data types, type casting, and basic input/output operations in Java. |
| **CO 2** | **Implementing** control structures such as conditional statements and loops to perform arithmetic operations and generate sequences. |
| **CO 3** | **Creating** and manipulate arrays and demonstrate basic inheritance, polymorphism, and class hierarchies in Java applications. |
| **CO 4** | **Developing** and test advanced Java applications using multithreading, file handling, collections, and exception handling to solve real-world problems. |

## LAB EXPERIMENTS

| **Ex. No** | **Lab Task** | **MAPPED CO/COS** |
|---|---|---|
| **1** | A Java Application that manages a small library system | **CO1** |
| **2** | A Java Application to manage a simple bank account system | **CO1** |
| **3** | A Java class hierarchy for a simple educational institution system | **CO2** |
| **4** | A system for managing different types of vehicles in a rental service | **CO2** |
| **4** | A Java Application for a basic shape drawing application | **CO3** |
| **5** | A Java multithreaded application that simulates a banking system | **CO3** |
| **6** | A file management system that supports operations such as reading, writing, copying, and navigating files and directories | **CO4** |

| | | |
|---|---|---|
| **7** | A contact management system that utilizes different data structures like Lists, Sets, Queues, and Maps | **CO4** |

# DISCRETE MATHEMATICS

| | | | | |
|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Discrete Mathematics** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCS203** | 3-1-0 | 4 | 40 |
| **Type of Course:** | Major-14 | | | |

**Pre-requisite(s), if any:** Basic of Mathematics

**Course Perspective.** In this comprehensive course on Discrete Mathematics, students will delve into foundational mathematical concepts essential for computer science and problem-solving. They will explore set theory, logic, relations, and graph theory, gaining proficiency in modeling and analyzing discrete structures. By mastering combinatorics, students will tackle counting problems and optimization techniques. Additionally, they will delve into number theory, understanding cryptographic algorithms and their security implications. Overall, this course equips learners with the analytical skills needed to address complex computational challenges and lays a solid groundwork for further studies in computer science and mathematics.

**Course Outcomes (COs).**

| COs | Statements |
|---|---|
| **CO 1** | **Applying** set theory concepts, analyze logical expressions, and use mathematical induction in proofs. |
| **CO 2** | **Understanding** Model relations, understand graph theory basics, and solve problems related to graphs. |
| **CO 3** | **Analyzing** Count combinatorial objects, explore discrete structures, and apply optimization techniques |
| **CO 4** | **Utilizing** number theory concepts and understand cryptographic algorithms. |

**Course Outline:**

**Unit Number: 1** **Title: Set Theory and Logic** **No. of hours: 10**

**Set Theory**

1. **Basic Concepts**:
    - Notations and terminology (union, intersection, complement)
    - Types of sets (finite, infinite, empty, universal)
    - Multisets (elements with multiplicity)
2. **Ordered Pairs and Cartesian Product**:
    - Definition of ordered pairs
    - Properties of Cartesian product
3. **Set Algebra and Proofs**:
    - Set operations (union, intersection, difference)
    - Proofs of set identities (De Morgan's laws, distributive properties)

**Logic**

1. **Propositional Logic**:
    - Syntax and semantics
    - Truth tables for logical connectives (AND, OR, NOT)
    - Tautologies and contradictions
2. **Predicate Logic**:
    - Quantifiers (universal and existential)
    - Predicate calculus
    - Proofs using mathematical induction

**Unit Number: 2** **Title: Relations and Graph Theory** **No. of hours: 10**

**Relations**

1. **Representation and Properties**:
    - Matrices, graphs, and directed graphs
    - Reflexive, symmetric, and transitive relations
2. **Equivalence Relations and Partitions**:
    - Equivalence classes
    - Equivalence partitions
3. **Graph Theory Basics**:
    - Definitions (vertices, edges, degree)
    - Types of graphs (simple, directed, weighted)
    - Graph representations (adjacency matrix, adjacency list)

4. **Graph Algorithms**:
    - Depth-First Search (DFS)
    - Breadth-First Search (BFS)
    - Shortest path algorithms (Dijkstra's, Bellman-Ford)

| | | |
|---|---|---|
| **Unit Number: 3** | **Title: Combinatorics and Discrete Structures** | **No. of hours: 10** |

**Combinatorics**

1. **Counting Principles**:
    - Product rule, sum rule
    - Permutations and combinations
    - Binomial coefficients
2. **Inclusion-Exclusion Principle**:
    - Solving problems with overlapping sets
3. **Generating Functions**:
    - Formal power series for combinatorial problems
    - Applications in counting

**Discrete Structures**

1. **Trees and Recurrence Relations**:
    - Tree properties (rooted, binary, spanning trees)
    - Solving linear recurrence relations
2. **Finite State Machines and Regular Languages**:
    - Deterministic Finite Automata (DFA)
    - Regular expressions
    - Regular languages
3. **Formal Languages and Grammars**:
    - Context-free grammars
    - Chomsky hierarchy

| | | |
|---|---|---|
| **Unit Number: 4** | **Title: Number Theory and Cryptography** | **No. of hours: 10** |

**Number Theory**

1. **Divisibility and Modular Arithmetic**:
    - Greatest common divisor (GCD)

  - Modular inverses
  - Euler’s totient function
2. **Congruences and Fermat’s Little Theorem**:
  - Solving congruences
  - Applications in cryptography
3. **RSA Encryption**:
  - Key generation
  - Encryption and decryption

**Cryptography**

1. **Symmetric-Key Cryptography**:
  - Data Encryption Standard (DES)
  - Advanced Encryption Standard (AES)
2. **Public-Key Cryptography**:
  - RSA algorithm
  - Diffie-Hellman key exchange
3. **Digital Signatures and Hash Functions**:
  - Ensuring data integrity
  - Cryptanalysis techniques

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures**: Introduce key concepts in discrete mathematics using PPTs and examples.
2. **Conceptual Understanding**: Cover topics like logic, set theory, combinatorics, and graph theory.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on solving discrete math problems.
4. **Theory Assignments**: Assign theoretical problems, with solutions discussed in class.
5. **Group Work**: Collaborate on projects involving applications of discrete mathematics.
6. **Case Studies**: Analyze real-world applications of discrete math in computer science and cryptography.
7. **Continuous Feedback**: Implement quizzes and feedback sessions to assess understanding.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home problems emphasizing discrete math concepts.
2. **Lab Projects**: Facilitate hands-on activities applying discrete math to programming or algorithms.
3. **Question Bank**: Provide practice problems and resources for self-assessment.
4. **Online Forums**: Create platforms for discussing discrete math problems and solutions.

5. **Self-Study for Case Studies**: Encourage independent research on applications of discrete mathematics.
6. **Collaborative Projects**: Organize group projects focused on real-world problems using discrete math.

**Text Books:**

1. Discrete Mathematics, Seymour Lipschutz and Marc Lipson, Schaum's Outline Series, 3rd Edition, 2009.
2. Discrete Mathematics, Richard Johnsonbaugh, Pearson, 8th Edition, 2017
3. Discrete Mathematics and Its Applications, Kenneth H. Rosen, McGraw Hill

**References**

**R 1.** Elements of Discrete Mathematics, C. L Liu, McGraw-Hill Inc, 1985. Applied Combinatorics, Alan Tucker.

**R 2.** Concrete Mathematics, Ronald Graham, Donald Knuth, and Oren Patashnik, 2nd Edition **-** Pearson Education Publishers.

# DATA STRUCTURES

| | | | | |
|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Data Structures** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCS205** | 4-0-0 | 4 | 40 |
| **Type of Course:** | Major-15 | | | |

**Pre-requisite(s), if any:** Basics of Computer Programming

**Course Perspective:** This course provides a comprehensive introduction to data structures and algorithms, essential components in the field of computer science that are critical for designing efficient software systems. Data structures serve as the building blocks for data management and organization, crucial for implementing effective algorithms that solve real-world computational problems. The course is structured to not only impart theoretical knowledge but also practical skills through hands-on implementation and problem-solving.

**The Course Outcomes (COs).** On completion of the course the participants will be able to:

| COs | **Statements** |
|---|---|
| **CO 1** | **Understanding** and apply basic and advanced data structures. |
| **CO 2** | **Analyzing** and compare various sorting and searching algorithms. |
| **CO 3** | **Designing** and utilize algorithms for advanced data manipulation. |
| **CO 4** | **Implementing** and evaluate algorithms using hashing and advanced algorithmic techniques. |
| **CO 5** | **Developing** applications that integrate data structures with file I/O operations and handle data dynamically. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

**Unit Number: 1** **Title: Foundations of Data Structures** **No. of hours: 9**

**Introduction:** Abstract Data Type, Elementary Data Organization.

**Measuring efficiency of an Algorithm:** Time and Space Complexity Analysis, Asymptotic notations.

**Arrays:** Single and Multidimensional Arrays, Representation of Arrays: Row Major Order, and Column Major Order, Application of arrays, Sparse Matrices.

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: Linear Data Structures** | **No. of hours: 11** |

**Linked lists:** Array and Dynamic Implementation of Single Linked Lists, Doubly Linked List, Circularly Linked List, Operations on a Linked List. Insertion, Deletion, Traversal, Polynomial Representation, Addition and Multiplication.

**Stacks:** Stack operations: Push & Pop, Array and Linked list implementation of Stack, Applications: Prefix and Postfix Expressions, Evaluation of postfix expression, Recursion.

**Queues:** Queue operations: Create, Add, Delete, full and empty queues, Array and linked implementation of queues, Dequeue, Circular queues and Priority Queue.

| | | |
|---|---|---|
| **Unit Number: 3** | **Title: Trees and Graphs** | **No. of hours: 10** |

**Searching:** Sequential search, Binary Search.

**Sorting:** Insertion Sort, Selection, Bubble Sort, Quick Sort, Merge Sort, Heap Sort, Radix Sort, Bucket Sort, Shell Sort.

**Hashing:** Hash Function, Hash Table, Collision Resolution Strategies.

| | | |
|---|---|---|
| **Unit Number: 4** | **Title: Advanced Sorting, Searching, and Algorithm Techniques** | **No. of hours: 10** |

**Trees:** Basic terminology, Binary Trees, Array and linked list implementation, Types of Binary Tree, Extended Binary Trees, Algebraic Expressions, Tree Traversal algorithms: Inorder, Preorder and Postorder, Threaded Binary trees, Search, Addition and deletion of an element in a binary tree, AVL Trees, Heaps, B Trees, B+ Trees and their applications, Evaluating an expression tree

**Graphs:** Representation (Matrix and Linked), Traversals, Shortest path, Topological sort. Dijkstra's Algorithm, Floyd Warshall's Algorithm, Minimum Spanning Tree Algorithms (Kruskal's Algorithm, Prim's Algorithm).

**L1= Remember, L2= Understand, L3= Apply, L4= Analyze, L5= Evaluate and L6= Create**

**Learning Experiences**

**Classroom Learning Experience**

1. **Interactive Lectures:** Introduce key concepts in data structures using PPTs and coding demonstrations.

2. **Conceptual Understanding**: Cover topics like arrays, linked lists, stacks, queues, trees, and graphs.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on implementing and using various data structures.
4. **Theory Assignments**: Assign theoretical problems that reinforce data structure concepts, discussed in class.
5. **Group Work**: Collaborate on projects that require designing and optimizing data structures.
6. **Case Studies**: Analyze real-world applications of data structures in software development.
7. **Continuous Feedback**: Implement quizzes and peer reviews to assess understanding and coding practices.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home projects that apply data structure concepts to practical problems.
2. **Lab Projects**: Facilitate hands-on programming tasks using data structures in real-world scenarios.
3. **Question Bank**: Provide practice problems and resources for self-assessment on data structures.
4. **Online Forums**: Create platforms for discussing data structure challenges and solutions.
5. **Self-Study for Case Studies**: Encourage independent research on efficient data structure implementations.
6. **Collaborative Projects**: Organize group projects focused on developing applications using various data structures.

**Textbooks**

1. Seymour Lipschutz, “Data Structures”, 2nd Edition, 2015
2. Aaron Tanenbaum, “Data Structures Using C”, 2nd edition, 2016
3. Ellis Horowitz and Sartaj Sahni, “Fundamentals of data structures” 2nd edition, 2017
4. Data Structures Using C (2nd. ed.). Reema Thareja. Oxford University Press, Inc., USA. 2018.

**References**

1. E. Horowitz and S. Sahani, “Fundamentals of Data Structures”, Galgotia Book source Pvt. Ltd.
2. Data Structures & Algorithms in Python by John Canning, Alan Broder, Robert Lafore Addison-Wesley Professional ISBN: 9780134855912.
3. "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.
4. Problem Solving with Algorithms and Data Structures Using Python" by Brad Miller and David Ranum.

**Additional Readings:**

**Online References for Learning Data Structures**

I) **MIT OpenCourseWare - Introduction to Algorithms (6.006)**

  a. Free course materials from MIT's undergraduate course on algorithms, which includes data structures. Lectures, assignments, and exams are available online.

  b. Link: MIT OpenCourseWare - Introduction to Algorithms

II) **LeetCode - Data Structures**

  a. A platform for practicing coding problems. It provides numerous problems related to data structures, complete with solutions and discussions.

  b. Link: LeetCode - Data Structures

# DATA STRUCTURES LAB

| | | | |
|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** **Data Structures lab** | **Course Code** **ENCS253** | **L-T-P** 0-0-2 | **Credits** 1 |
| **Type of Course:** | Major-17 | | |

**Pre-requisite(s), if any:** Basics of Computer Programming with C++

**Defined Course Outcomes**

| COs | |
|---|---|
| CO 1 | **Analyzing** and evaluate the time and space complexity of algorithms for various scenarios, demonstrating an understanding of asymptotic notations. |
| CO 2 | **Implementing** and manipulate single-dimensional and multi-dimensional arrays, including operations like insertion, deletion, and traversal. |
| CO 3 | **Developing** and perform operations on linked lists (single, doubly, and circularly linked), stacks, and queues using both array and linked list representations. |
| CO 4 | **Designing** and analyze the efficiency of different sorting and searching algorithms, as well as implement and compare advanced data structures like binary search trees, AVL trees, and graph algorithms. |

## LAB EXPERIMENTS

| **S.No** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| 1 | Given an array of integers, perform the following operations: reverse the array, find the maximum and minimum elements, and calculate the sum and average of the elements. Implement functions to perform each operation and ensure the time complexity is optimal. | CO1 |
| 2 | Given an array, rotate the array to the right by k steps, where k is non-negative. Implement the rotation in-place with O(1) extra space. | CO1 |
| 3 | Write a function to merge two sorted arrays into a single sorted array. The function should handle arrays of different lengths and ensure the final array is sorted. | CO1 |

| | | |
|---|---|---|
| 4 | Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, find the one that is missing from the array. Implement an efficient algorithm with O(n) time complexity. | CO1 |
| 5 | Find the kth largest element in an unsorted array. Note that it is the kth largest element in sorted order, not the kth distinct element. Implement an efficient algorithm with O(n log n) time complexity. | CO1 |
| 6 | Given an unsorted array of integers, find the length of the longest consecutive elements sequence. Your algorithm should run in O(n) time complexity. | CO1 |
| 7 | Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand. Write a function to search for a target value in the array. If found, return its index; otherwise, return -1. Your algorithm should run in O(log n) time complexity. | CO1 |
| 8 | Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum. Implement an efficient algorithm with O(n) time complexity using Kadane's Algorithm. | CO1 |
| 9 | Write a function to move all zeros to the end of an array while maintaining the relative order of the non-zero elements. Implement the function with O(n) time complexity and O(1) extra space. | CO2 |
| 10 | Write a class to implement a singly linked list with methods to insert an element at the head, insert an element at the tail, delete an element by value, and traverse the list to print all elements. | CO2 |
| 11 | Using linked lists, write a function to add two polynomials. Each node in the linked list represents a term in the polynomial with its coefficient and exponent. Implement the function to handle polynomials of different degrees. | CO2 |
| 12 | Implement a doubly linked list with methods to insert an element at the head, insert an element at the tail, delete an element by value, and reverse the list. Ensure that all operations handle edge cases appropriately. | CO2 |
| 13 | Create a circular linked list with methods to insert an element, delete an element by value, and traverse the list. Ensure that the list maintains its circular nature after each operation. | CO2 |
| 14 | Write a function to evaluate a given postfix expression using a stack. The function should support basic arithmetic operations (+, -, *, /) and handle invalid expressions gracefully. | CO2 |

|  |  |  |
|---|---|---|
| 15 | Implement a stack using a singly linked list with methods for push, pop, and peek operations. Ensure that the stack handles edge cases, such as popping from an empty stack, appropriately. | CO2 |
| 16 | Write a function to convert an infix expression to a postfix expression using a stack. The function should handle parentheses and operator precedence correctly. | CO2 |
| 17 | Create a circular queue using an array with methods for enqueue, dequeue, and checking if the queue is empty or full. Ensure that the circular nature of the queue is maintained after each operation. | CO2 |
| 18 | Given a sorted array that has been rotated at an unknown pivot, write a function to search for a target value in the array. If the target exists, return its index; otherwise, return -1. Implement an efficient algorithm with O(log n) time complexity using binary search. | CO2 |
| 19 | Find the kth largest element in an unsorted array. Note that it is the kth largest element in sorted order, not the kth distinct element. Implement an efficient algorithm with O(n log n) time complexity. | CO2 |
| 20 | Given a collection of intervals, merge all overlapping intervals and return an array of the non-overlapping intervals that cover all the intervals in the input. Implement an efficient algorithm with O(n log n) time complexity. | CO2 |
| 21 | Given a non-empty array of integers, return the k most frequent elements. Implement an efficient algorithm with O(n log k) time complexity. | CO2 |
| 22 | Given an array of integers that is already sorted in ascending order, find two numbers such that they add up to a specific target number. Return the indices of the two numbers (1-indexed) as an integer array. Implement an algorithm with O(n) time complexity. | CO2 |
| 23 | Given an array that has been rotated at an unknown pivot, write a function to search for a target value in the array. Implement the solution using binary search with O(log n) time complexity. | CO3 |
| 24 | Write a function to merge k sorted linked lists and return it as one sorted list. Implement an efficient solution using a min-heap with O(N log k) time complexity, where N is the total number of nodes. | CO3 |
| 25 | Given a non-empty array of integers, return the k most frequent elements. Implement the solution with O(n log k) time complexity using a min-heap and a hash map. | CO3 |

| | | |
|---|---|---|
| 26 | Implement various sorting algorithms including Quick Sort, Merge Sort, Heap Sort, and analyze their performance on different input sizes. Ensure the implementation handles edge cases such as duplicate values and nearly sorted arrays. | CO3 |
| 27 | Given preorder and inorder traversal of a tree, construct the binary tree. Implement an efficient algorithm with O(n) time complexity using a hash map to store the index of elements in the inorder traversal. | CO4 |
| 28 | Implement Dijkstra's algorithm to find the shortest path from a source vertex to all other vertices in a weighted graph. Use both adjacency matrix and adjacency list representations for the graph. Ensure the algorithm handles negative weights appropriately. | CO4 |
| 29 | Implement Kruskal's algorithm to find the minimum spanning tree of a graph. Use a union-find data structure to detect cycles and ensure the algorithm runs in O(E log E) time complexity. | CO4 |
| 30 | Given a binary tree representing an arithmetic expression, write a function to evaluate the expression and return the result. Each leaf node is an operand, and each internal node is an operator. Implement an efficient recursive algorithm. | CO4 |

# VAC II

| | | | | |
|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Community Engagement Service** | **Course Code:** **VAC II** | **L-T-P** 2-0-0 | **Credits** 2 | **Semester** II |
| **Type of Course:** | Value Added Course | | | |
| **Duration** | 30 Hrs | | | |

**Course Objectives:**

- To engage students in meaningful social service activities.
- To develop socially responsible engineers.
- To apply technical and non-technical skills for the benefit of society.
- To foster community engagement and support.

**Course Outline:**

**1. Introduction**

**Overview of the Course:** The **Community Engagement Service (VAC II)** course at K.R. Mangalam University is designed to integrate social responsibility with technical education. This 30-hour value-added course encourages students to engage in meaningful social service activities, applying their technical and non-technical skills to benefit various sections of society. Through hands-on involvement, students will develop a deeper understanding of community needs and contribute positively to societal development.

**Importance of Social Service in Engineering Education:** Incorporating social service into technical education is crucial for nurturing well-rounded professionals who are not only technically proficient but also socially conscious. By participating in community-oriented projects, students can bridge the gap between theory and practice, gaining real-world experience that enhances their problem-solving skills. Engaging in social service fosters empathy, teamwork, and leadership qualities, which are essential attributes for successful engineers dedicated to making a positive impact on society.

**Expectations and Requirements:** Students enrolled in this course are expected to actively participate in chosen social service activities, dedicating at least 30 hours over weekends. They must document their engagement through video clips and photographs, maintaining a detailed logbook of their activities. Additionally, students are required to prepare a comprehensive report and a 10-minute video presentation demonstrating their engagement, learning experiences, and the impact of their initiatives. Evaluation will be based on the quality and relevance of documentation, the depth of the report, and the effectiveness of the video presentation in showcasing their contributions and outcomes.

**2. Possible Engagement Activities**

Students can choose from a variety of activities, including but not limited to:

**Development and Innovation**

**Develop Innovative Tools**: Create solutions such as mobile apps and web-based platforms to address societal needs.

1. **Lever-Powered Wheelchairs**: Develop control applications to enhance mobility for differently-abled individuals.
2. **Assistive Devices**: Design simple devices using basic sensors to improve daily living for people with disabilities.
3. **Environmental Monitoring**: Build introductory systems using Arduino and web dashboards to raise community awareness about air and water quality.
4. **Eco-Friendly Practices**: Create web applications that promote sustainable living and track user participation.
5. **Waste Management**: Implement basic data management systems for efficient waste management in local communities.
6. **Energy Optimization**: Develop algorithms to optimize energy consumption in households and public buildings.
7. **Water Quality Monitoring**: Design systems with sensors and mobile apps to ensure safe drinking water in rural areas.
8. **Smart Agriculture**: Create tools using microcontrollers to support farmers with automated irrigation and soil condition monitoring.
9. **Cybersecurity**: Implement basic practices to protect sensitive data in sustainable technology applications.
10. **Health Tracking**: Develop simple mobile applications to monitor fitness and wellness metrics, benefiting public health initiatives.
11. **Recycling Sorters**: Create introductory computer vision projects for sorting recyclables to aid municipal recycling programs.
12. **Environmental Data Analysis**: Conduct basic projects on environmental data sets to identify trends and propose solutions for urban planning and conservation efforts.
13. **Chemical Analysis Programs**: Create Python programs to support educational institutions.
14. **Electronic Circuits for Physics**: Develop circuits to aid students in experiments.
15. **Engineering Mathematics Tools**: Design simulation tools to assist in academic research.

**Education and Mentorship**

1. **Tutoring and Mentorship**: Provide tutoring and mentorship to underprivileged children.

2. **Day Camps**: Organize and run day camps for low-income children during weekends.
3. **Educational Opportunities for Incarcerated Individuals**: Volunteer to provide educational programs and mentorship to incarcerated individuals.
4. **Skill Development Workshops**: Conduct workshops to teach various skills to children based on students' expertise.

**Community Service and Development**

1. **Local Charities and Community Projects**: Volunteer with local charities to support community development projects.
2. **Entrepreneurship Initiatives**: Help villagers improve their livelihood through entrepreneurship initiatives.
3. **Women Empowerment Programs**: Empower women through skill enhancement, awareness programs, and entrepreneurship training.
4. **Digital Awareness Programs**: Conduct programs on cybersecurity and social media safety to protect against digital frauds.

**Cultural and Traditional Skills**

1. **Traditional Skills Learning**: Spend time with villagers to learn traditional skills such as pottery, carpentry, weaving, etc.
2. **Artisan Marketing Assistance**: Help artisans market their crafts through digital platforms and e-commerce.

**Technology for Social Good**

1. **Problem-Solving with Technology**: Use technology to solve specific problems faced by certain sections of society, such as developing apps for community support.
2. **Community Development Tools**: Create tools and resources to assist in community development and problem-solving.

**Healthcare Domain**

1. **Health Awareness Campaigns**: Organize campaigns to raise awareness about hygiene, nutrition, and preventive healthcare.
2. **Medical Camp Assistance**: Volunteer at medical camps to support healthcare delivery in underserved areas.
3. **Mental Health Support**: Conduct workshops and support groups focusing on mental health awareness and assistance.
4. **Telemedicine Services**: Assist in setting up and running telemedicine services for remote communities.

**Print Media and Social Platforms**

1. **Community Newsletters**: Create and distribute newsletters to share important community news and stories.
2. **Social Media Campaigns**: Run social media campaigns to raise awareness on various social issues and promote community initiatives.

**Other Possible Domains**

1. **Environmental Conservation**: Participate in tree planting drives, clean-up campaigns, and conservation projects.
2. **Disaster Relief Support**: Assist in disaster relief efforts, providing aid and support to affected communities.
3. **Animal Welfare**: Volunteer at animal shelters, support animal rescue operations, and promote animal welfare initiatives.
4. **Cultural Preservation**: Work on projects to preserve and promote local cultural heritage and traditions.

**3. Documentation and Proof of Engagement**

- Students must provide relevant proofs in the form of video clips and day-wise photographs.
- Maintain a logbook detailing the hours spent and activities undertaken.

**4. Reporting and Presentation**

- Prepare a detailed report on the engagement activities.
- Create a 10-minute video demonstrating the overall engagement, learning experiences, and impact.
- The video should include testimonials from beneficiaries showcasing the outcomes and benefits.

**Evaluation Criteria:**

The evaluation of the VAC will be based on the following rubrics, totaling 100 marks:

| **Criteria** | **Marks** |
|---|---|
| Relevant Proofs (video clips, day-wise photographs) | 20 |
| Detailed Report | 30 |
| Video Presentation (10 minutes) | 50 |
| - Demonstration of overall engagement | |
| - Learning experiences | |
| - Initiative impact on society | |
| - Testimonials from beneficiaries | |

**Rubrics for Evaluation:**

| **Evaluation Criteria** | **Excellent (10)** | **Good (7-9)** | **Satisfactory (5-6)** | **Needs Improvement (1-4)** |
|---|---|---|---|---|
| **Relevant Proofs** | Comprehensive and well-documented | Adequately documented | Basic documentation provided | Inadequate or missing documentation |
| **Detailed Report** | Thorough, well-structured, insightful | Clear and informative | Basic structure and content | Lacks detail and structure |
| **Video Presentation** | Highly engaging and impactful | Engaging and informative | Basic engagement and clarity | Lacks engagement and clarity |
| **Demonstration of Engagement** | Clearly demonstrates active and meaningful engagement | Shows active involvement | Demonstrates some involvement | Lacks clear demonstration of involvement |
| **Learning Experiences** | Profound insights and reflections | Clear insights and reflections | Basic reflections | Lacks depth in reflections |
| **Initiative Impact on Society** | Significant positive impact shown | Evident positive impact | Some positive impact | Minimal or unclear impact |
| **Testimonials from Beneficiaries** | Strong and compelling testimonials | Clear and supportive testimonials | Basic testimonials | Lack of or weak testimonials |

**Implementation Plan:**

1. **Orientation Session:** Introduce students to the VAC and explain the objectives and expectations.
2. **Activity Selection:** Students select their preferred engagement activities.
3. **Engagement Phase:** Students actively participate in the chosen activities, documenting their involvement.
4. **Reporting Phase:** Students prepare their detailed report and video presentation.
5. **Evaluation:** Faculty evaluates students based on the provided rubrics.
6. **Feedback Session:** Provide constructive feedback to students for continuous improvement.

**Conclusion:**

This Value-Added Course aims to instill a sense of social responsibility in engineering students, encouraging them to apply their skills for the betterment of society. By engaging in various social service activities, students will gain valuable experiences that complement their technical education, fostering holistic development and community engagement.

**Student Report Template**

Title Page:

- Course Title: Community Engagement Service (VAC-II)
- Student Name:
- Enrollment Number:
- Semester: II
- Program: B.Tech (CSE) including all Specializations, BCA, B.Sc
- Date:

1. Introduction:

- Overview of the Course: Provide a brief overview of the Community Engagement Service (VAC II) course, highlighting its purpose and importance.
- Importance of Social Service in Engineering Education: Discuss why incorporating social service into engineering education is crucial for developing well-rounded professionals.
- Expectations and Requirements: Outline the course expectations, including participation, documentation, and reporting requirements.

2. Chosen Activity:

- Activity Name: State the name of the chosen social service activity.
- Description of the Activity: Provide a detailed description of the activity.
- Objectives and Goals: List the objectives and goals of the activity.

3. Methodology:

- Steps Taken: Describe the steps taken to complete the activity.
- Tools and Techniques Used: Mention any tools or techniques used, such as mobile apps, web-based platforms, etc.
- Duration of Engagement: Specify the duration of the engagement (at least 30 hours).

4. Implementation:

- Detailed Description of Engagement Activities: Provide a detailed log of the engagement activities, including day-wise descriptions.
- Proof of Engagement: Include video clips, photographs, and other relevant proofs of engagement.

5. Impact Analysis:

- Impact on Society: Analyze the impact of the activity on society.
- Benefits to the Community: Discuss the benefits provided to the community.

- Testimonials from Beneficiaries: Include testimonials from beneficiaries showcasing the outcomes and benefits.

6. Learning Experiences:
   - Skills and Knowledge Gained: Detail the skills and knowledge gained through the activity.
   - Reflections on the Experience: Reflect on the overall experience.
   - Challenges Faced and Overcome: Describe any challenges faced and how they were overcome.
7. Ethical Considerations:
   - Ethical Issues Encountered: Discuss any ethical issues encountered during the activity.
   - Solutions and Best Practices: Provide solutions and best practices for addressing these ethical issues.
   - Reflections on Social Responsibility: Reflect on the importance of social responsibility.
8. Conclusions:
   - Summary of the Experience: Summarize the overall experience.
   - Personal Growth and Development: Discuss personal growth and development resulting from the activity.
   - Future Recommendations: Provide recommendations for future engagements.
9. Appendices:
   - Additional Documents and Proofs: Include any additional supporting documents, such as logbook entries and extra photographs.
   - Video Presentation Link: Provide a link to the video presentation.

# Verbal Ability

| | | | |
|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name: Life Skills for Professionals - I** | **Course Code** | **L-T-P** | **Credits** |
| | **AEC006** | 3-0-0 | 3 |

**Type of Course:** AEC-1

**Duration:** 36 hours

**Pre-requisite(s), if any: Nil**

**Course Perspective:** The course aims to improve language proficiency in three key areas: grammar, vocabulary and identification of grammatical errors in writing. Language proficiency enables students to comprehend lectures, understand course materials and enhances students' ability to express themselves clearly and effectively. In many professions, strong language skills are a prerequisite. Whether in business, medicine, law, or science, being able to communicate fluently and accurately is essential for collaboration, negotiation, and advancement. A strong command of verbal abilities can significantly impact job interviews. It allows candidates to answer questions confidently, demonstrate their qualifications effectively and leave a positive impression on potential employers.

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the grammar rules and word meaning (Vocabulary). |
| **CO 2** | **Applying** grammar rules and vocabulary in different context & purpose |
| **CO 3** | **Analyzing** situations/ context of communication and selecting appropriate grammar and words. |
| **CO 4** | **Developing** sentences and paragraphs to describe and narrate a situation. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number: 1** | **Title: Vocabulary Development and Application** | **No. of hours: 10** |

**Content:**

Understanding the concept of root words, Prefix and suffix, Ways to enhance Vocabulary, Crosswords and word quizzes, Confusing words, One word substitution, Odd one out, Synonyms and Antonyms, Commonly misspelt words, Idioms and Phrases.

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: Fundamentals of Grammar and Sentence Structure** | **No. of hours: 8** |

**Content:** Introduction to Parts of Speech, Tenses and its ‘rules, Sentences (Simple, Compound and Complex), Subject Verb Agreement, Pronoun Antecedent agreement, Phrases and Clauses.

| | | |
|---|---|---|
| **Unit Number: 3** | **Title: Mastering Sentence Accuracy and Completion Skills** | **No. of hours: 12** |

**Content:**

Spot the error (grammatical errors in a sentence), Sentence Correction (Improvement of sentences based on Grammar rules), Sentence Completion, Cloze Tests

| | | |
|---|---|---|
| **Unit Number: 4** | **Title: Enhancing Sentence Structure and Reading Comprehension** | **No. of hours: 6** |

**Content:**

Logical Arrangement of Sentences, Comprehending passages, Contextual questions, Anagrams, Analogies

**Learning Experiences**

**Classroom Learning Experience**

1. **Interactive Lectures**: Introduce key life skills concepts using PPTs and real-life examples.
2. **Conceptual Understanding**: Cover topics like communication, teamwork, and problem-solving strategies.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on practical scenarios and decision-making.
4. **Theory Assignments**: Assign reflective essays on personal development and professional growth.
5. **Group Work**: Collaborate on projects that enhance interpersonal skills and teamwork.
6. **Case Studies**: Analyze successful professionals and their life skills in various industries.

7. **Continuous Feedback**: Implement quizzes and peer reviews to assess understanding and application of skills.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home projects focused on applying life skills in real-world contexts.
2. **Workshops**: Facilitate hands-on sessions for practicing communication and leadership skills.
3. **Question Bank**: Provide resources for self-assessment on life skills development.
4. **Online Forums**: Create platforms for discussing life skills challenges and sharing experiences.
5. **Self-Study for Case Studies**: Encourage independent research on effective life skills practices.
6. **Collaborative Projects**: Organize group projects aimed at community engagement and skill application.

**References**

**R1.** Norman Lewis – Word Power Made Easy
**R2.** Wren & Martin – High School English Grammar & Composition
**R3.** R.S. Agarwal & Vikas Agarwal – Quick Learning Objective General English
**R4.** S.P. Bakshi - Objective General English
**R5.** Praxis Groups -Campus Recruitment Complete Reference

**Additional Readings:**

**Communication Resources**

I) **LinkedIn Learning - Communication Foundations**
  a. This course offers foundational knowledge on effective communication strategies, including verbal and non-verbal communication.
  b. Link: LinkedIn Learning - Communication Foundations

II) **Khan Academy - Grammar**
  a. Khan Academy provides a detailed course on grammar which is fundamental to clear and effective communication.
  b. Link: Khan Academy - Grammar

**Non-Verbal Communication Resources**

**R 1. LinkedIn Learning - Developing Your Emotional Intelligence**
- Enhancing emotional intelligence is key to improving non-verbal communication skills. This course covers practical strategies.
- Link: LinkedIn Learning - Developing Your Emotional Intelligence

**R 2. YouTube - TED Talks on Body Language**
- TED Talks provide insightful videos on the importance of body language and how to master it.

- Link: YouTube - TED Talks on Body Language

**Number Systems and Basic Mathematics Resources**

1. **Khan Academy - Arithmetic and Pre-Algebra**
    - Comprehensive lessons on basic arithmetic and pre-algebra, including number systems, divisibility, and more.
    - Link: Khan Academy - Arithmetic and Pre-Algebra
2. **Coursera - Introduction to Mathematical Thinking**
    - This course introduces mathematical thinking, including logic, which is fundamental to understanding number systems.
    - Link: Coursera - Introduction to Mathematical Thinking
3. **edX - Introduction to Algebra**
    - Offers a strong foundation in algebra, covering topics such as factors, LCM, HCF, and simplification.
    - Link: edX - Introduction to Algebra

**Time Management Resources**

1. **Coursera - Work Smarter, Not Harder: Time Management for Personal & Professional Productivity**
    - This course provides practical strategies for effective time management and productivity.
    - Link: Coursera - Time Management
2. **edX - Time Management Strategies for Project Management**
    - Focuses on time management within the context of project management, offering valuable insights and techniques.
    - Link: edX - Time Management Strategies

## SUMMER INTERNSHIP-I

**Program Name** **B. Tech (Computer Science and Engineering)**

| | | | |
|---|---|---|---|
| **Course Name: Summer Internship-I** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSI251** | 0-0-0 | 2 |

**Type of Course:** INT-I

**Pre-requisite(s), if any: NA**

**Duration:**

The internship will last for **six weeks**. It will take place after the completion of the 2nd semester and before the commencement of the 3rd semester.

**Internship Options:**

Students can choose from the following options:

1. **Industry Internship (Offline):**
    1. Students must produce a joining letter at the start and a relieving letter upon completion.
2. **Global Certifications:**
    1. Students can opt for globally recognized certification programs relevant to their field of study.
3. **Research Internship:**
    1. Students can engage in a research internship under the mentorship of a faculty member for six weeks.
4. **On-Campus Industry Internship Programs:**
    1. The university will offer on-campus internships in collaboration with industry partners.
5. **Internships at Renowned Institutions:**
    1. Students can pursue summer internships at esteemed institutions such as IITs, NITs, Central Universities, etc.

**Report Submission and Evaluation:**

1. **Report Preparation:**
    1. Students must prepare a detailed report documenting their internship experience and submit it to the department. A copy of the report will be kept for departmental records.
2. **Case Study/Project/Research Paper:**
    1. Each student must complete one of the following as part of their internship outcome:
        1. A case study

        2. A project
        3. A research paper suitable for publication

3. **Presentation:**
    1. Students are required to present their learning outcomes and results from their summer internship as part of the evaluation process.

**Evaluation Criteria for Summer Internship (Out of 100 Marks)**

**1. Relevance to Learning Outcomes (30 Marks)**

1. **Case Study/Project/Research Paper Relevance (15 Marks):**
    - Directly relates to core subjects: 15 marks
    - Partially relates to core subjects: 10 marks
    - Minimally relates to core subjects: 5 marks
    - Not relevant: 0 marks
2. **Application of Theoretical Knowledge (15 Marks):**
    - Extensive application of theoretical knowledge: 15 marks
    - Moderate application of theoretical knowledge: 10 marks
    - Minimal application of theoretical knowledge: 5 marks
    - No application of theoretical knowledge: 0 marks

**2. Skill Acquisition (30 Marks)**

1. **New Technical Skills Acquired (15 Marks):**
    - Highly relevant and advanced technical skills: 15 marks
    - Moderately relevant technical skills: 10 marks
    - Basic technical skills: 5 marks
    - No new skills acquired: 0 marks
2. **Professional and Soft Skills Development (15 Marks):**
    - Significant improvement in professional and soft skills: 15 marks
    - Moderate improvement in professional and soft skills: 10 marks
    - Basic improvement in professional and soft skills: 5 marks
    - No improvement: 0 marks

**3. Report Quality (20 Marks)**

- **Structure and Organization (10 Marks):**
    - Well-structured and organized report: 10 marks
    - Moderately structured report: 7 marks
    - Poorly structured report: 3 marks
    - No structure: 0 marks

- **Clarity and Comprehensiveness (10 Marks):**
    - Clear and comprehensive report: 10 marks
    - Moderately clear and comprehensive report: 7 marks
    - Vague and incomplete report: 3 marks
    - Incomprehensible report: 0 marks

**4. Presentation (20 Marks)**

- **Content Delivery (10 Marks):**
    - Clear, engaging, and thorough delivery: 10 marks
    - Clear but less engaging delivery: 7 marks
    - Somewhat clear and engaging delivery: 3 marks
    - Unclear and disengaging delivery: 0 marks
- **Visual Aids and Communication Skills (10 Marks):**
    - Effective use of visual aids and excellent communication skills: 10 marks
    - Moderate use of visual aids and good communication skills: 7 marks
    - Basic use of visual aids and fair communication skills: 3 marks
    - No use of visual aids and poor communication skills: 0 marks

**Total: 100 Marks**

**Course Outcomes:**

By the end of this course, students will be able to:

- **Apply Theoretical Knowledge:**
    - Integrate and apply theoretical knowledge gained during coursework to real-world industry or research problems.
- **Develop Technical Skills:**
    - Acquire and demonstrate advanced technical skills relevant to the field of computer science and engineering through practical experience.
- **Conduct Independent Research:**
    - Execute independent research projects, including problem identification, literature review, methodology design, data collection, and analysis.
- **Prepare Professional Reports:**
    - Compile comprehensive and well-structured reports that document the internship experience, project details, research findings, and conclusions.
- **Enhance Problem-Solving Abilities:**
    - Develop enhanced problem-solving and critical thinking skills by tackling practical challenges encountered during the internship.

- **Improve Professional and Soft Skills:**
  - Exhibit improved professional and soft skills, including communication, teamwork, time management, and adaptability in a professional setting.
- **Present Findings Effectively:**
  - Deliver clear and engaging presentations to effectively communicate project outcomes, research findings, and acquired knowledge to peers and faculty members.
- **Pursue Lifelong Learning:**
  - Demonstrate a commitment to lifelong learning by engaging in continuous skill development and staying updated with emerging trends and technologies in the field.

**Learning Experiences**

**Classroom Learning Experience**

1. **Orientation Sessions**: Introduce internship objectives and expectations through interactive presentations.
2. **Skill Development Workshops**: Cover essential skills like communication, teamwork, and time management.
3. **Project Planning**: Guide students in developing project proposals aligned with internship goals.
4. **Group Discussions**: Facilitate discussions on challenges and experiences in workplace settings.
5. **Guest Speakers**: Invite industry professionals to share insights and best practices.
6. **Continuous Feedback**: Implement regular check-ins and peer reviews to assess progress and learning.

**Outside Classroom Learning Experience**

1. **Internship Placement**: Engage students in real-world work environments to apply learned skills.
2. **Reflective Journals**: Encourage students to document their experiences and lessons learned during the internship.
3. **Project Implementation**: Work on assigned projects and tasks within the organization.
4. **Networking Opportunities**: Create platforms for students to connect with industry professionals.
5. **Self-Assessment**: Provide tools for students to evaluate their performance and growth.
6. **Final Presentations**: Organize sessions for students to present their internship experiences and outcomes.

# COMPETITIVE CODING -I

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **COMPETITIVE CODING -I** | | 3-0-0 | 0 |
| **Type of Course:** | AUDIT -II | | |
| **Contact Hours** | 30 | | |

**Course Outcomes**

| | |
|---|---|
| **CO1** | **Understanding** problem-solving strategies and techniques relevant to competitive programming |
| **CO2** | **Analyzing** the efficiency of algorithms in terms of time and space complexity using asymptotic notations |
| **CO3** | **Applying** core programming concepts such as functions, recursion, and dynamic memory allocation to solve computational problems |
| **CO4** | **Implementing** solutions for problems involving arrays and strings, utilizing efficient operations and algorithms |

**Course Outline:**

| **Unit Number: 1** | **Title: Foundations of Competitive Programming** | **No. of hours: 8** |
|---|---|---|
| **Content:**<br>Introduction to Competitive Programming Platforms<br>▪ Overview of major platforms: Codeforces, LeetCode, HackerRank etc.<br>▪ Setting up accounts and environment for competitive programming.<br>▪ Solving introductory problems to get familiar with the platforms.<br>Problem-Solving Strategies<br>▪ Techniques for solving problems<br>▪ Greedy Algorithms: Understanding local optimality leading to global solutions.<br>▪ Divide and Conquer: Solving problems by breaking them into subproblems (with examples like Merge Sort).<br>▪ Brute Force: Iterative approach to solve problems when constraints are small. | | |

| **Unit Number: 2** | **Title:** Time and Space Complexity of Algorithms | **No. of hours: 8** |
|---|---|---|
| **Content:**<br>Time and Space Complexity:<br>▪ Big O Notation: Definition, examples, and practical importance.<br>▪ Common Complexities: O(1), O(log n), O(n), O(n log n), O(n^2), etc.<br>▪ Impact of time and space complexity on algorithm performance.<br>▪ Asymptotic notations<br>▪ Best, Average and worst case analysis of Algorithms | | |
| **Unit Number: 3** | **Title: Core Programming Concepts** | **No. of hours: 8** |
| **Content:**<br>**Functions:** Definition and Declaration, Function Overloading, Recursion and Backtracking<br>**Pointers:** Basics of Pointers and References, Pointer Arithmetic, Dynamic Memory Allocation (malloc, free, new, delete)<br>**Files:** File I/O Operations (Reading/Writing), File Handling in C++/Java/Python, **Vectors (in C++/ArrayLists in Java):** Declaration, Initialization, and Operations, Dynamic Resizing | | |
| **Unit Number: 4** | **Title: Arrays and Strings** | **No. of hours: 6** |
| **Content:**<br>Arrays: Operations, Manipulations<br>Strings: Operations, Substrings, Pattern Matching<br>Operations on arrays: Insertion, deletion, and traversal.<br>String operations: Concatenation, substring search.<br>Key Problems: Rotating arrays, reversing strings, finding longest substrings without repeating characters | | |

## Experiment List

| **Problem Statement** | **Mapped COs** |
|---|---|
| 1. Two Sum: Find two numbers that add up to a specific target. | CO1 |
| 2. Best Time to Buy and Sell Stock: Maximize profit from stock prices. | CO1 |

| **Problem Statement** | **Mapped COs** |
|---|---|
| 3. Valid Parentheses: Check if a string contains valid parentheses. | CO1 |
| 4. Greedy Algorithm: Jump Game - Can you reach the end of the array? | CO1 |
| 5. Divide and Conquer: Merge Sort implementation to sort an array. | CO1 |
| 6. Brute Force: Find all subsets of a given set. | CO1 |
| 7. Greedy Algorithm: Minimum Number of Platforms Required for Trains | CO1 |
| 8. Divide and Conquer: Maximum Subarray (Kadane's Algorithm) | CO1 |
| 9. Brute Force: Count number of occurrences of a substring in a string. | CO1 |
| 10. Greedy Algorithm: Coin Change Problem (Minimum Coins) | CO1 |
| 11. Time Complexity: Check if a number is prime using O(√n) complexity. | CO2 |
| 12. Sorting: QuickSort algorithm with O(n log n) complexity. | CO2 |
| 13. Big O Notation: Analyze time complexity of an algorithm. | CO2 |
| 14. Space Complexity: Fibonacci with O(n) space complexity. | CO2 |
| 15. Time Complexity: Find first duplicate element in an array with O(n) time. | CO2 |
| 16. Time Complexity: Search an element in a rotated sorted array in O(log n) time. | CO2 |
| 17. Complexity Analysis: Binary Search Tree operations with complexity O(log n). | CO2 |
| 18. Analyze best, average, and worst case for Insertion Sort. | CO2 |
| 19. Time and Space Complexity: Check the complexity of an algorithm (recurrences). | CO2 |
| 20. Time Complexity: Compute factorial recursively with complexity analysis. | CO2 |
| 21. Recursion: Generate all permutations of a string. | CO3 |
| 22. Dynamic Memory Allocation: Implement a dynamic array (vector) from scratch. | CO3 |
| 23. Backtracking: Solve the N-Queens problem using recursion. | CO3 |
| 24. Pointers: Swap two numbers using pointers in C++. | CO3 |
| 25. File Handling: Read and write data to a file in Python/C++/Java. | CO3 |
| 26. Function Overloading: Implement overloaded functions for adding integers and floats. | CO3 |
| 27. Dynamic Memory Allocation: Use malloc and free to manage memory in C. | CO3 |
| 28. Recursion: Solve Tower of Hanoi using recursion. | CO3 |

| Problem Statement | Mapped COs |
|---|---|
| 29. Arrays: Rotate an array to the right by k steps. | CO4 |
| 30. Strings: Find the longest substring without repeating characters. | CO4 |

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures**: Introduce competitive coding concepts and strategies using PPTs and coding demos.
2. **Algorithm Workshops**: Cover key algorithms and data structures essential for coding competitions.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on solving competitive coding problems.
4. **Mock Contests**: Organize timed coding contests to simulate competition environments.
5. **Group Discussions**: Facilitate discussions on problem-solving techniques and optimization strategies.
6. **Continuous Feedback**: Implement peer reviews and performance assessments after practice sessions.

**Outside Classroom Learning Experience**

1. **Practice Assignments**: Assign coding problems from various online platforms for independent practice.
2. **Online Competitions**: Encourage participation in external coding competitions and hackathons.
3. **Question Bank**: Provide a repository of practice problems and resources for self-assessment.
4. **Online Forums**: Create platforms for students to discuss coding challenges and share solutions.
5. **Self-Study Resources**: Recommend books and online courses for further learning on algorithms and data structures.
6. **Collaborative Projects**: Organize group projects to develop coding applications or solve larger problems together.

**Textbooks:**

- "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.
- "Algorithm Design" by Jon Kleinberg and Éva Tardos.

**Online Resources:**

- LeetCode (https://leetcode.com/)

- HackerRank (https://www.hackerrank.com/)
- GeeksforGeeks (https://www.geeksforgeeks.org/)

**List of Suggested Competitive Programming Courses:**

1. Algorithms and Data Structures by MIT OpenCourseWare
2. Introduction to Competitive Programming by NPTEL
3. Competitive Programming by HackerRank
4. The Bible of Competitive Programming & Coding Interviews

*All students must complete one online course from the suggested programs.*

**Web References**

- https://www.geeksforgeeks.org/competitive-programming-a-complete-guide/
- https://www.geeksforgeeks.org/must-do-coding-questions-for-companies-like-amazon-microsoft-adobe/
- https://github.com/parikshit223933/Coding-Ninjas-Competitive-Programming
- https://www.hackerearth.com/getstarted-competitive-programming/
- https://www.csestack.org/competitive-coding-questions/

**References to Interview Questions**

- https://www.simplilearn.com/coding-interview-questions-article
- https://www.csestack.org/competitive-coding-questions/
- https://www.geeksforgeeks.org/a-competitive-programmers-interview/
- https://www.geeksforgeeks.org/must-do-coding-questions-for-companies-like-amazon-microsoft-adobe/

# FUNDAMENTALS OF AI & MACHINE LEARNING

| | | | | | |
|---|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | | |
| **Course Name: Fundamentals of AI & Machine Learning** | | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | | **SEC034** | 2-0-0 | 2 | 30 |
| **Type of Course:** | SEC-4 | | | | |

**Pre-requisite(s), if any:** fundamentals of Python programming

**Course Perspective.** This course introduces students to the fundamental concepts and techniques of artificial intelligence (AI) and machine learning (ML), which are pivotal in the development of intelligent systems and data-driven decision-making. The curriculum is structured to provide a comprehensive understanding of various machine learning algorithms, performance metrics, and the principles of designing effective learning systems. The course is designed to bridge the gap between theoretical foundations and practical applications, equipping students with the skills necessary to tackle real-world problems using AI and ML.

**Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Identifying** key components of problem-solving using search algorithms. |
| **CO 2** | Learn to **Applying** supervised learning algorithms to problems. |
| **CO 3** | Learn to **Applying** unsupervised learning algorithms to problems |
| **CO 4** | **Analyzing** Unsupervised Learning and Neural Networks |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

**Unit Number: 1 Title: Introduction to Artificial Intelligence No. of hours: 05**

**Content:**

Introduction to AI: History and Evolution, Basic Concepts, Applications in Various Domains.

Search Algorithms: Uninformed Search (BFS, DFS), Informed Search (A*), Heuristic Functions.

AI Problem Solving: Constraint Satisfaction Problems (CSP).

Hands-on: Implement BFS/DFS and A* Search for a sample problem.

**Unit Number: 2 Title: Supervised Learning in Machine Learning No. of hours: 10**

**Content:**

Regression: Linear regression., Polynomial regression, Model evaluation (MSE, R-squared).

Classification: Logistic regression, Decision trees, Evaluation metrics (accuracy, precision, recall).

**Unit Number: 3 Title: Unsupervised Learning and Neural Networks No. of hours: 10**

Content:

Clustering: K-means clustering, Hierarchical clustering, Applications (customer segmentation, image compression).

Neural Networks: Perceptrons and activation functions, Feedforward neural networks, Backpropagation and gradient descent.

**Unit Number: 4 Title: Reinforcement Learning No. of hours: 05**

Content:

Reinforcement Learning : Introduction to Reinforcement Learning, Markov decision processes (MDPs), Q-learning and policy-based methods, Applications (game playing, robotics).

**Learning Experiences:**

**Inside Classroom Learning Experience:**

1. **Interactive Lectures and Discussions:** Engage in interactive lectures covering key AI and ML concepts such as search algorithms, supervised learning, and unsupervised learning. Participate in classroom discussions to explore the evolution, applications, and impact of AI across different domains.
2. **Algorithmic Demonstrations:** Witness live demonstrations of AI problem-solving approaches like BFS, DFS, and A* search algorithms, including their real-time implementation on whiteboards and teaching screens.
3. **Collaborative Problem Solving:** Work in small groups to solve Constraint Satisfaction Problems (CSP) or design simple neural networks. This fosters teamwork and encourages peer-to-peer learning within the classroom environment.
4. **Hands-On Coding Sessions:** Engage in hands-on coding during classroom hours where the instructor guides students through the step-by-step implementation of supervised learning algorithms (e.g., linear regression, decision trees) and unsupervised learning algorithms (e.g., K-means clustering) using Python.
5. **In-Class Assessments and Quizzes:** Participate in regular in-class assessments such as quizzes and short exercises to test comprehension of topics like heuristic functions, regression models, or neural networks.
6. **Interactive Q&A Sessions:** Participate in frequent Q&A sessions after completing each unit to clarify doubts, review critical concepts, and discuss real-world applications of AI and ML models.
7. **Algorithm Walkthroughs:** Engage in algorithm walkthroughs on the board, where students trace the steps of algorithms like A* or backpropagation in neural networks to understand their working mechanisms.

**Outside Classroom Learning Experience:**

1. **Hands-On Implementation:** Implement AI search algorithms like BFS, DFS, and A* on sample problems or real-world scenarios as part of homework or lab assignments, applying theoretical knowledge in practical projects. Develop supervised and unsupervised learning models using Python libraries (such as Scikit-learn) on real-world datasets for tasks such as customer segmentation or image classification.
2. **Real-World Project Applications:** Work on group projects that apply machine learning techniques to real-world datasets, such as predicting house prices using regression or performing customer segmentation using K-means clustering.

3. **Collaborative Peer Reviews:** Engage in peer reviews and group problem-solving activities outside the classroom to analyze case studies or complete machine learning tasks. This promotes critical thinking, feedback exchange, and collaboration.
4. **Online Learning Platforms (Moodle LMS):** Access additional learning materials, participate in online discussions, and submit assignments through the Moodle LMS platform. Utilize forums for further exploration of topics such as Markov decision processes or reinforcement learning.
5. **Self-Directed Learning with Video Tutorials:** Watch supplementary video tutorials covering advanced AI/ML concepts like neural networks and reinforcement learning to deepen understanding. Use these resources to clarify complex topics and supplement classroom learning.
6. **Data Science Competitions and Challenges:** Participate in online competitions or challenges (such as Kaggle) where students apply machine learning algorithms to new datasets, gaining exposure to diverse problem-solving techniques and real-world applications.
7. **Continuous Feedback through Project Submissions:** Receive continuous feedback from instructors on lab assignments and projects, identifying areas of improvement and providing guidance for refining AI/ML models.
8. **Utilization of ICT Tools:** Make use of ICT tools, such as Jupyter Notebooks and online IDEs, for coding and testing algorithms. Utilize these tools for independent practice or to complete lab-based tasks remotely.

**Text Books**

1. Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). Pearson Education.
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
3. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
4. Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

## SEMESTER: IV

## PROBABILITY AND STATISTICS

| | | | | |
|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Probability and Statistics** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENMA202** | 3-1-0 | 4 | 36 |
| **Type of Course:** | Major-18 | | | |
| **Pre-requisite(s), if any:** | | | | |

**Course Perspective.** This course equips students with the fundamental tools of probability and statistics to analyze and solve real-world engineering problems. This Course Covers Probability, conditional probability, independence, random variables, expected value, moment generating function, probability generating function, characteristic function, specific discrete and continuous distributions, covariance, correlation coefficient, central limit theorem. Sampling distributions, point and interval estimation, testing of hypothesis, goodness of fit and contingency tables, ANOVA, Correlation v/s Causation, linear regression, multiple Regression, Linear and Non-Linear Regression, Parameters estimation via LSE. The course is divided into 5 modules:

a) Foundations of Probability
b) Engineering Applications of Probability Distributions
c) Descriptive Statistics and Regression Analysis
d) Inferential Statistics for Engineers

**Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Applying** probability concepts (conditional probability, Bayes' theorem) to model uncertainties in engineering systems (e.g., system failures, signal noise). |
| **CO 2** | **Utilizing** probability distributions (Bernoulli, Binomial, Poisson, etc.) to analyze random variables encountered in engineering processes (e.g., component lifetimes, machine errors). |

|  |  |
|---|---|
| **CO 3** | **Performing** statistical analysis (descriptive statistics, correlation, regression) to summarize and interpret engineering data. |
| **CO 4** | **Conducting** hypothesis testing (parametric and non-parametric) to draw statistically sound conclusions from engineering experiments (e.g., compare design performance, validate new materials). |
| **CO 5** | **Estimating** unknown parameters in engineering models using techniques like least squares and maximum likelihood estimation. |
| **CO 6** | **Communicating** results effectively using graphical techniques and statistical measures to support engineering decision-making. |

## Course Outline:

| **Unit Number: 1** | **Title: Foundations of Probability** | **No. of hours: 8** |
|---|---|---|

**Content:**

- Basic notions of probability, events, and set operations.
- Conditional probability and independence of events.
- Applications of Bayes' theorem in engineering problems (e.g., system reliability, fault diagnosis).
- Random variables: Discrete and continuous types.
- Cumulative distribution functions (CDF) and probability mass/density functions (PMF/PDF).
- Transformations of random variables and their impact on probability distributions.
- Mathematical expectation, moments, and moment generating functions for characterizing probability distributions.

| **Unit Number: 2** | **Title: Engineering Applications of Probability Distributions** | **No. of hours: 8** |
|---|---|---|

**Content:**

- Joint and marginal distributions for analyzing multiple random variables in engineering systems.
- Discrete distributions: Bernoulli, Binomial, Geometric, and Poisson distributions and their applications in engineering (e.g., defect rates, service arrivals).
- Continuous distributions: Uniform, Exponential, Gamma, and Normal distributions and their applications in engineering (e.g., component lifetimes, sensor measurements).
- Central Limit Theorem and its implications for approximating probability distributions in engineering.

| | | |
|---|---|---|
| • Law of Large Numbers and its role in understanding statistical convergence in engineering experiments. | | |
| **Unit Number: 3** | **Title: Descriptive Statistics and Regression Analysis** | **No. of hours: 8** |
| **Content:**<br>• Descriptive statistics: Measures of central tendency (mean, median, mode) and variability (variance, standard deviation).<br>• Visualization techniques (histograms, scatter plots) to explore and communicate engineering data.<br>• Correlation coefficient and covariance to quantify relationships between engineering variables.<br>• Linear regression for modeling the relationship between a dependent variable and one or more independent variables in engineering contexts.<br>• Least squares method for fitting regression models and interpreting the results in engineering applications. | | |
| **Unit Number: 4** | **Title: Inferential Statistics for Engineers** | **No. of hours: 8** |
| **Content:**<br>• Introduction to statistical inference and the concept of sampling in engineering experiments.<br>• Sampling distributions of mean and variance, and their role in statistical inference.<br>• Estimation techniques: Point estimation and confidence intervals for unknown parameters in engineering models.<br>• Hypothesis testing: Parametric (Z-test, T-test, ANOVA) and non-parametric tests for making data-driven decisions in engineering. | | |

**Learning Experiences:**

**Inside Classroom Learning Experience:**

1. **Interactive Lectures:** Participate in lectures using interactive PPTs and video lectures to grasp fundamental concepts, supplemented by real-world engineering applications of probability and statistics.
2. **Problem-Based Assignments:** Solve problem-based theory assignments to apply probability concepts and statistical techniques to model and analyze engineering problems.
3. **Collaborative Learning:** Collaborate in group activities to tackle complex engineering problems, analyze data, and interpret results, fostering teamwork and peer learning.

4. **Continuous Assessment:** Receive regular feedback through continuous assessments, quizzes, and model question papers, with support available from the course instructor for additional help.

**Outside Classroom Learning Experience:**

1. **Project-Based Labs:** Engage in hands-on lab assignments where students will work on real engineering datasets to perform statistical analysis, regression modeling, and hypothesis testing.
2. **Collaborative Learning:** Continue collaboration with peers outside the classroom to discuss lab projects, assignments, and complex problems related to data analysis and engineering problem-solving.
3. **Use of Technology:** Utilize ICT tools like Moodle LMS for accessing course materials, submitting assignments, and participating in online discussions, enhancing learning efficiency.
4. **Practical Application:** Apply statistical techniques and probability distributions to engineering scenarios through case studies, ensuring practical understanding and real-world relevance.
5. **Continuous Feedback:** Receive continuous feedback and support from the course instructor via email, discussion forums, and office hours. Students can seek additional help outside class through online discussions or by scheduling appointments for one-on-one sessions.

**Text Book**

1. Sheldon M Ross, *Introduction to Probability and Statistics for Engineers and Scientists*, Elsevier

**Reference Books**

*1.* Robert V. Hogg, Joseph W. McKean & Allen T. Craig (2013). *Introduction to Mathematical Statistics* (7th edition), Pearson Education.
2. Irwin Miller &Marylees Miller (2014). *John E. Freund's Mathematical Statistics with Applications* (8thedition). Pearson. Dorling Kindersley Pvt. Ltd. India.
3. Jim Pitman (1993). *Probability*, Springer-Verlag.
4. A. M. Yaglom and I. M. Yaglom (1983). *Probability and Information*. D. Reidel Publishing Company. Distributed by Hindustan Publishing Corporation (India) Delhi

**Additional Readings:**

**Online Learning Resources :**

**I.** NPTL lecture https://archive.nptel.ac.in/courses/111/105/111105090/

II. **E-book:** https://www.utstat.toronto.edu/mikevans/jeffrosenthal/book.pdf

III. **Data Analysis using R**: Students can learn and apply statistical techniques using R, an open-source statistical programming language, to analyze real-world datasets.

a. https://www.coursera.org/learn/data-analysis-r

b. https://www.udemy.com/course/data-analysis-with-r/

IV. **Hypothesis Testing with Excel**: Students can learn how to perform hypothesis testing using Excel's built-in statistical functions and conduct statistical analyses on data sets.

   a. https://www.coursera.org/learn/hypothesis-testing-python-excel

# WEB PROGRAMMING WITH PYTHON AND JAVASCRIPT LAB

| | | | | |
|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** | | **Course Code** | **L-T-P** | **Credits** |
| **Web programming with Python and JavaScript lab** | | **SEC035** | 0-0-4 | 2 |
| **Type of Course:** | SEC-5 | | | |
| **Pre-requisite(s), if any: Basics of Python** | | | | |

**Course Outcomes**

| **COs** | **Statements** |
|---|---|
| CO 1 | Understanding the Fundamentals of Web Development |
| CO 2 | Developing Dynamic Web Pages using JavaScript |
| CO 3 | Building Backend Applications using Python and Flask |
| CO 4 | Integrating Frontend and Backend Technologies to Develop Full-Stack Web Applications |

**Course Objective:** The objective of this course is to provide hands-on experience in web programming using Python for backend development and JavaScript for frontend development. Students will learn how to integrate these technologies to build dynamic, interactive web applications.

**Course Outline:**

**Week 1-2: Introduction to Web Development**

- Lab Activities:
  - Setting up the development environment (VS Code, Python, Node.js)
  - Basics of HTML and CSS
  - Introduction to JavaScript
  - Creating simple static web pages

**Week 3: JavaScript Basics**

- Lab Activities:
  - Variables, data types, and operators
  - Control structures (if-else, loops)

    - Functions and scope
    - DOM manipulation

**Week 4: Introduction to Python and Flask**

- Lab Activities:
    - Python basics (variables, control structures, functions)
    - Introduction to Flask
    - Setting up a Flask project
    - Creating basic routes and templates

**Week 5: Dynamic Web Pages with JavaScript**

- Lab Activities:
    - Working with JSON
    - Making AJAX requests
    - Fetch API
    - Building a dynamic web page that interacts with a server

**Week 6: Database Integration**

- Lab Activities:
    - Introduction to SQL and SQLite
    - Connecting Flask to a SQLite database
    - Performing CRUD operations (Create, Read, Update, Delete)
    - Simple data-driven web application

**Week 7: User Authentication**

- Lab Activities:
    - Implementing user registration and login
    - Hashing passwords
    - Session management in Flask
    - Protecting routes with authentication

**Week 8: Frontend Frameworks Introduction**

- Lab Activities:
    - Introduction to a frontend framework (React.js or Vue.js)
    - Setting up a simple project
    - Creating reusable components

**Week 9: Integrating Frontend with Backend**

- Lab Activities:
    - Connecting the frontend framework with Flask backend

    - Building a single-page application (SPA)
    - Data fetching and state management

**Week 10: Real-Time Features**

- Lab Activities:
    - Introduction to WebSocket
    - Setting up real-time communication with Flask-SocketIO
    - Creating a real-time chat application

**Week 11-12: Final Project Development and Presentation**

- Lab Activities:
    - Planning and developing a final project
    - Implementing features learned throughout the course
    - Testing and debugging the application
    - Preparing a project presentation
    - Demonstrating the project to the class

**Learning Experiences:**

**Inside Classroom Learning Experience:**

1. **Hands-On Lab Activities:** Engage in lab sessions to set up development environments, create static web pages with HTML/CSS, and build dynamic pages using JavaScript, providing a practical foundation in web development tools and technologies.
2. **Interactive JavaScript and Python Sessions:** Gain experience with JavaScript basics, including variables, control structures, and DOM manipulation, and learn Python programming and Flask basics to build and manage web applications.
3. **Dynamic Web Development:** Learn to work with JSON, make AJAX requests, and use the Fetch API to build interactive, data-driven web pages that communicate with servers.
4. **Database Integration:** Explore SQL and SQLite, connect Flask to a database, and perform CRUD operations, allowing students to create simple, data-driven web applications.
5. **Frontend Framework Introduction:** Get introduced to frontend frameworks like React.js or Vue.js, set up projects, and create reusable components to enhance the development of dynamic user interfaces.
6. **User Authentication:** Implement user registration, login functionalities, password hashing, and session management in Flask, focusing on securing web applications.

**Outside Classroom Learning Experience:**

1. **Project Development and Practice:** Practice building static and dynamic web pages using HTML, CSS, JavaScript, and Python outside of lab sessions. Work on projects to reinforce concepts learned in class.
2. **Database Integration Practice:** Independently practice connecting Flask applications to databases like SQLite, performing CRUD operations, and managing database interactions outside the classroom to strengthen the application of database concepts.
3. **Frontend Frameworks Self-Study:** Explore and experiment with frontend frameworks such as React.js or Vue.js by setting up personal projects, applying classroom knowledge to build reusable components and dynamic interfaces on your own.
4. **Final Project:** Apply learned concepts by planning, developing, and presenting a comprehensive final project, including testing, debugging, and demonstrating the application, integrating both frontend and backend skills.
5. **Self-Guided Learning of AJAX and Fetch API:** Independently study and apply AJAX requests and the Fetch API to communicate with servers and dynamically update web pages, reinforcing these skills through self-directed coding exercises.
6. **User Authentication Practice:** Practice implementing secure web applications using Flask by setting up user authentication systems, including session management and password security, as part of outside lab assignments and independent coding practice.

## Lab Experiments

| Course Outcome | Project | Problem Statement |
|---|---|---|
| Apply | Project 1: Personal Portfolio Website | Develop a personal portfolio website using HTML, CSS, and basic JavaScript. The website should showcase the user's skills, projects, and contact information. |
| | Project 2: Basic Company Landing Page | Create a landing page for a fictitious company using HTML and CSS. The landing page should include sections such as an introduction, services, testimonials, and contact information. |
| Analyze | Project 4: Dynamic To-Do List | Create a dynamic to-do list application where users can add, edit, and delete tasks. |

| Course Outcome | Project | Problem Statement |
|---|---|---|
| | Project 3: Interactive Quiz Application | Develop an interactive quiz application where users can answer multiple-choice questions. |
| Create | Project 5: Simple Blog Platform | Develop a simple blog platform using Flask where users can create, read, update, and delete blog posts. |
| | Project 6: Contact Management System | Create a contact management system using Flask where users can store and manage their contacts. |
| Evaluate | Project 7: E-Commerce Website | Develop an e-commerce website where users can browse products, add items to a shopping cart, and complete purchases. |

# ANALYSIS AND DESIGN OF ALGORITHMS

| | | | | | |
|---|---|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | | |
| **Course Name:** | | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| **Analysis and Design of Algorithms** | | **ENCS202** | 4-0-0 | 4 | 40 |
| **Type of Course:** | **Major-**19 | | | | |

**Pre-requisite(s), if any: -** Introduction to Data Structures

**Course Perspective** The course provides a comprehensive introduction to the fundamental concepts of algorithm analysis and design, essential for various fields such as computer science, engineering, data science, and artificial intelligence. This course equips students with the tools to understand, analyze, and develop efficient algorithms for solving complex computational problems. By covering both theoretical foundations and practical applications, the course ensures a balanced approach to learning. The course is divided into 5 modules:

a) Introduction and Complexity Analysis
b) Divide and Conquer, Greedy Algorithms, and Dynamic Programming
c) Graph Algorithms
d) Advanced Algorithms and Techniques
e) Advanced Topics and Implementation Techniques

**The Course Outcomes (COs).**

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** fundamental algorithmic concepts and analyze their complexities. |
| **CO 2** | **Analyzing and evaluating** the performance of various algorithms. |
| **CO 3** | **Designing** efficient algorithms considering both time and space complexities. |
| **CO 4** | **Applying** algorithmic problem-solving strategies to a variety of computational problems. |
| **CO 5** | **Developing** skills to implement and optimize algorithms for real-world applications. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number: 1** | **Title:** Introduction and Complexity Analysis | **No. of hours: 10** |

**Content:**

**Introduction to Algorithms**: Definition, importance, specification and role in problem-solving.

**Algorithm Analysis**: RAM computational models, Time and space complexity, Asymptotic Notations, best, average, and worst-case analysis, Performance measurement of algorithms, rate of growth of algorithms

**Recurrence Relations**: Solving recurrences using substitution, recursion tree, and master theorem.

| | | |
|---|---|---|
| **Unit Number: 2** | **Title:** Divide and Conquer, Greedy Algorithms, and Dynamic Programming | **No. of hours: 10** |

**Content:**

**Divide and Conquer**: General method, Merge Sort, Quick Sort, Binary Search, Strassen's Matrix Multiplication, finding maximum and minimum.

**Greedy Algorithms**: Concept and characteristics, Fractional Knapsack, Activity Selection, Huffman Coding.

**Dynamic Programming**: General Method, Longest Common Subsequence, 0/1 Knapsack problem, Matrix Chain Multiplication, Travelling salesman problem.

| | | |
|---|---|---|
| **Unit Number: 3** | **Title:** Graph Algorithms | **No. of hours: 10** |

**Content:**

**Graph Representation**: Adjacency matrix, adjacency list.

**Graph Traversal Algorithms**: Depth First Search (DFS), Breadth First Search (BFS), Applications of graph (Topological sorting).

**Shortest Path Algorithms**: Dijkstra's algorithm, Bellman-Ford algorithm, Floyd-Warshall algorithm.

**Minimum Spanning Tree Algorithms**: Kruskal's algorithm, Prim's algorithm.

| | | |
|---|---|---|
| **Unit Number: 4** | **Title**: Advanced Algorithms and Techniques | **No. of hours: 10** |

**Content:**

**Backtracking**: Concept, examples (N-Queens problem, Sum of subsets).

**Branch and Bound**: Concept, examples (Traveling Salesman Problem, 0/1 Knapsack Problem).

**String Matching Algorithms**: Naive algorithm, Rabin-Karp algorithm, String matching with finite automata, Knuth-Morris-Pratt (KMP) algorithm.

**Introduction to NP-Completeness:** The class P and NP, Polynomial time, NP-complete and NP-hard.

Introduction to Approximation Algorithms and Randomized Algorithms

**Learning Experiences**

**Inside Classroom Learning Experience**

1. **Interactive Lectures**: Introduce key concepts in algorithm design and analysis using PPTs and examples.
2. **Conceptual Understanding**: Cover fundamental topics like complexity analysis, recursion, and algorithmic paradigms.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on designing and analyzing various algorithms.
4. **Case Studies**: Analyze real-world algorithms and their applications in different fields.
5. **Group Work**: Collaborate on projects that involve implementing and optimizing algorithms.
6. **Continuous Feedback**: Implement quizzes and peer reviews to assess understanding and application of concepts.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home projects requiring analysis and design of algorithms for practical problems.
2. **Lab Projects**: Facilitate hands-on programming tasks to implement and test algorithms.
3. **Question Bank**: Provide practice problems and resources for self-assessment on algorithm concepts.
4. **Online Forums**: Create platforms for discussing algorithm challenges and sharing solutions.
5. **Self-Study for Case Studies**: Encourage independent research on advancements in algorithm design and analysis.
6. **Collaborative Projects**: Organize group projects focused on solving complex problems using algorithms.

**Additional Readings:**

**Online Learning Resources:**

I) **MIT OpenCourseWare - Introduction to Algorithms (6.006)**
   a. A comprehensive resource from MIT covering fundamental and advanced algorithms.
   b. Link: MIT OpenCourseWare - Introduction to Algorithms

II) **HackerRank - Algorithms Practice**
   a. Provides a platform to practice and compete in coding challenges related to algorithms.
   b. Link: HackerRank - Algorithms Practice

III) **LeetCode - Algorithm Problems**

a. A platform offering a vast array of problems to practice algorithms and data structures.

b. Link: LeetCode - Algorithm Problems

# ANALYSIS AND DESIGN OF ALGORITHMS LAB

| | | | |
|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** **Analysis and Design of Algorithms Lab** | **Course Code** **ENCS256** | **L-T-P** 0-0-2 | **Credits** 1 |
| **Type of Course:** | Major-22 | | |

**Pre-requisite(s), if any: -** Data Structure

**Defined Course Outcomes**

| COs | Statements |
|---|---|
| CO 1 | **Analyzing** the time and space complexity of algorithms, demonstrating an understanding of asymptotic notations and performance metrics. |
| CO 2 | **Implementing** and compare sorting algorithms, such as bubble sort and insertion sort, and apply the divide and conquer technique to algorithms like merge sort and quick sort |
| CO 3 | **Solving** optimization problems using greedy and dynamic programming algorithms, such as the fractional knapsack problem and longest common subsequence |
| CO 4 | **Developing** graph algorithms for traversal, shortest path, and minimum spanning tree, applying techniques like DFS, BFS, Dijkstra's, and Kruskal's algorithms |
| CO 5 | **Implementing** advanced algorithms for problems like N-Queens, traveling salesman, and string matching using backtracking, branch and bound, and pattern matching techniques |

## Lab Experiments

| ***S.No*** | ***Lab Task*** | ***Mapped CO/COs*** |
|---|---|---|

|  |  |  |
|---|---|---|
| *1* | Conduct a case study on the efficiency of different sorting algorithms (e.g., Insertion Sort, Bubble Sort, Merge Sort, Quick Sort, Counting Sort, Radix sort, Bucket sort). | CO1 |
| *2* | Develop a tool in your preferred programming language to measure the performance of various algorithms. | CO1 |
| *3* | Develop a resource allocation system for a fictional company using greedy algorithms. | CO2 |
| *4* | Build a web application that solves dynamic programming problems. Implement solutions for the Longest Common Subsequence, 0/1 Knapsack Problem, and Matrix Chain Multiplication. | CO2 |
| *5* | Create a file compression tool using the Huffman Coding algorithm. Allow users to input a text file and generate the corresponding Huffman tree and encoded output | CO2 |
| *6* | Create a program to represent a social network using both adjacency matrix and adjacency list representations. | CO3 |
| *7* | Develop a web crawler simulation that uses Depth First Search (DFS) and Breadth First Search (BFS) algorithms to traverse a given website's pages. | CO3 |
| *8* | Design a city navigation system that calculates the shortest path between locations using Dijkstra’s algorithm, Bellman-Ford algorithm, and Floyd-Warshall algorithm. | CO3 |
| *9* | Implement a solution to the N-Queens problem using backtracking. Allow the user to input the size of the chessboard (N) and display all possible solutions | CO4 |
| *10* | Write a program to find all subsets of a given set of positive integers that sum up to a given value using the backtracking technique. | CO4 |
| *11* | Develop the simulation of various string matching algorithms and compare their runtime complexities. Display their time complexity graphs | CO4 |

# DATABASE MANAGEMENT SYSTEMS

| **Program Name** | **B. Tech (Computer Science and Engineering)** | | | |
|---|---|---|---|---|
| **Course Name:** **Database Management System** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | ENCS204 | 3-1-0 | 4 | 40 |
| **Type of Course:** | Major-20 | | | |
| Pre-requisite(s), if any: **Basics of Computer Fundamentals** | | | | |

**Course Perspective:** This course provides a comprehensive introduction to the fundamental concepts and advanced techniques of database management systems (DBMS). It is designed to equip students with the knowledge and skills required to design, implement, and manage databases effectively. The course covers a broad range of topics, including database architecture, data models, SQL, transaction management, concurrency control, database recovery, and security. The course is divided into 4 modules:

a) Introduction

b) Relational Query Languages

c) Transaction Processing and Storage Strategies

d) Advanced Topics and Database Security

**The Course Outcomes (COs).** On completion of the course the participants will be able to:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the fundamental concepts and architecture of database management systems, including data models and ER modeling. |
| **CO 2** | **Utilizing** Structured Query Language (SQL) and relational algebra for effective database querying and manipulation. |
| **CO 3** | **Applying** database design principles, including normalization and integrity constraints, to develop well-structured databases. |
| **CO 4** | **Analyzing** storage structures, transaction processing, concurrency control, and recovery protocols in databases. |

|  |  |
|---|---|
| **CO 5** | **Implementing** security measures and explores advanced database concepts such as distributed databases, data warehousing, and data mining. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| **Unit Number: 1** | **Title: Introduction** | **No. of hours: 12** |
|---|---|---|
| **Content:**<br>**Introduction to DBMS**: Overview, benefits, and applications.<br>**Database System Architecture**: Schemas, Instances, Data abstraction, data models (network model, relational model, object-oriented data model), Three schema architecture and data independence<br>**Entity-Relationship Model**: Entity Types, Entity Sets, Attributes, and Keys, Relationship Types, Relationship Sets, ER diagrams, Naming Conventions, Design issues.<br>**Integrity Constraints**: Primary key, foreign key, unique, not null, check constraints. | | |
| **Unit Number: 2** | **Title: Relational Query Languages** | **No. of hours: 8** |
| **Content:**<br>Relational Database Design, Relational query languages, Relational algebra, Tuple and domain relational calculus.<br>**SQL**: DDL (Data Definition Language), DML (Data Manipulation Language), DCL (Data Control Language).<br>**Query Processing and Optimization**: Evaluation of relational algebra expressions, query equivalence, join strategies, query optimization algorithms.<br>**Database Design**: Functional dependencies, normalization (1NF, 2NF, 3NF, BCNF, 4NF), dependency preservation, lossless decomposition.<br>**Open Source and Commercial DBMS**: Overview of MySQL, Oracle, DB2, SQL Server. | | |
| **Unit Number: 3** | **Title: Transaction Processing and Storage Strategies** | **No. of hours: 12** |
| **Content:**<br>**Transaction Management**: ACID properties, transaction states, serializability, conflict and view serializability. | | |

| | | |
|---|---|---|
| **Concurrency Control**: Lock-based protocols, timestamp-based protocols, multi-version concurrency control, deadlock handling.<br>**Database Recovery**: Recovery concepts, recovery techniques (log-based recovery, shadow paging), checkpoints.<br>**Storage Strategies**: File organization, indexing (single-level, multi-level), B-tree, B+ tree, hashing (static and dynamic). | | |
| **Unit Number: 4** | **Title: Advanced Topics and Database Security** | **No. of hours: 8** |
| **Content:**<br>**Database Security**: Authentication, authorization, access control, DAC (Discretionary Access Control), MAC (Mandatory Access Control), RBAC (Role-Based Access Control).<br>**Intrusion Detection**: Techniques and tools, SQL injection prevention.<br>**Advanced Database Topics**: Object-oriented databases, object-relational databases, logical databases, web databases.<br>**Distributed Databases**: Concepts, architecture, data fragmentation, replication, distributed query processing.<br>**Data Warehousing and Data Mining**: Concepts, architecture, OLAP, data preprocessing, data mining techniques. | | |

**Learning Experiences**

**Classroom Learning Experience**

1. **Interactive Lectures**: Introduce key concepts in database management using PPTs and case studies.
2. **Conceptual Understanding**: Cover fundamental topics like data modeling, normalization, and SQL.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on database design and query optimization.
4. **Case Studies**: Analyze real-world database systems and their architectures.
5. **Group Work**: Collaborate on projects that involve designing and implementing databases.
6. **Continuous Feedback**: Implement quizzes and peer reviews to assess understanding of database concepts.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home projects requiring the application of database management principles.

2. **Lab Projects**: Facilitate hands-on tasks to create, manipulate, and query databases using DBMS software.
3. **Question Bank**: Provide practice problems and resources for self-assessment on database topics.
4. **Online Forums**: Create platforms for discussing database challenges and sharing solutions.
5. **Self-Study for Case Studies**: Encourage independent research on current trends and technologies in database management.
6. **Collaborative Projects**: Organize group projects focused on developing database solutions for real-world problems.

**Text Books**

1. R. Elmasri and S.B. Navathe, 2000, Fundamentals of Database Systems, 3rd Ed, AW.
2. C.J. Date, 2000, An Introduction to Database Systems, 7th ED., Addison-Wesley.
3. Database System Concepts", 6th Edition by Abraham Silberschatz, Henry F. Korth, S. Sudarshan, McGraw-Hill.

**Additional Readings:**

**Online Learning Resources for "Database Management Systems"**

1. NPTEL-Database Management System
2. **MIT OpenCourseWare - Database Systems (6.830)**
    - Advanced course materials from MIT covering database system internals and advanced topics.
    - Link: MIT OpenCourseWare - Database Systems
3. **Oracle - Database 2-Day Developer's Guide**
    - Official documentation and guide for Oracle database developers.
    - Link: Oracle - Database 2-Day Developer's Guide
4. **SQLBolt - Learn SQL with interactive exercises**
    - Interactive SQL tutorials and exercises to practice database querying.
    - Link: SQLBolt - Learn SQL

# DATABASE MANAGEMENT SYSTEMS LAB

| | | | |
|---|---|---|---|
| **Program Name** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name: Database Management System Lab** | **Course Code** | **L-T-P** | **Credits** |
| | **ENCS254** | 0-0-2 | 1 |
| **Type of Course:** | Major-21 | | |

**Defined Course Outcomes**

| COs | |
|---|---|
| **CO 1** | **Designing and implementing** database schemas using both open-source and commercial DBMS, defining tables, relationships, and integrity constraints |
| **CO 2** | **Developing and analyzing** Entity-Relationship diagrams, relational schemas, and enforce normalization techniques to ensure database efficiency and integrity |
| **CO 3** | **Understanding** the Write and execute SQL queries for data definition, manipulation, and complex data retrieval, demonstrating proficiency in relational algebra and transaction processing |
| **CO 4** | **Implementing** advanced database concepts including indexing, concurrency control, recovery techniques, security features, and distributed database processing |

## Lab Experiments

| ***Ex. No*** | ***Lab Task*** | ***Mapped CO/COs*** |
|---|---|---|

| | | |
|---|---|---|
| 1 | Analyze and document the benefits of using a DBMS over a traditional file system for managing data. Use a case study of a small retail business to highlight the advantages of a DBMS in handling inventory, sales, and customer data. | CO1 |
| 2 | Design a three-schema architecture for a university management system. Create the internal schema, conceptual schema, and external schema.<br><br>Illustrate how data independence is achieved and provide examples of each schema with specific details. | CO1 |
| 3 | Design and implement an ER model for a university course registration system. The system should include entities such as Students, Courses, Professors, and Enrollments. Define relationships, attributes, and keys | CO1 |
| 4 | Design a relational database schema for an e-commerce platform that manages Customers, Products, Orders, Order Details, and Payments. Define the entities, their attributes, and the relationships between them, ensuring the schema is normalized to at least 3NF.<br><br>Use this schema to create an ER diagram and specify primary and foreign keys. | CO2 |
| 5 | Write SQL scripts to create the e-commerce platform database schema using Data Definition Language (DDL). Create tables for Customers, Products, Orders, Order Details, and Payments, and enforce primary keys, foreign keys, unique constraints, not null constraints, and check constraints. Ensure the database structure supports data integrity and consistency | CO2 |
| 6 | Populating, Querying, and Securing the E-commerce Database using SQL DML, DCL, and Relational Algebra/Calculus | CO2 |
| 7 | Design and implement a banking transaction management system that demonstrates the ACID properties. The system should handle various transaction states, ensuring serializability and data integrity.<br><br>Implement features for depositing, withdrawing, and transferring funds, and simulate scenarios to showcase conflict and view serializability. | CO3 |
| 8 | Develop a concurrency control mechanism for an e-commerce platform to manage simultaneous transactions, such as placing orders and updating inventory. | CO3 |

| | | |
|---|---|---|
| | Implement lock-based protocols, timestamp-based protocols, and multi-version concurrency control. Simulate scenarios where deadlock handling techniques are required to ensure smooth operation | |
| 9 | Design and implement a data warehousing solution for a financial analytics platform. Create a data warehouse to store historical financial data and perform OLAP operations for data analysis.<br><br>Implement data preprocessing techniques and apply data mining algorithms to discover patterns and insights from the financial data. Simulate various analytical queries and demonstrate how the data warehouse and mining techniques enhance decision-making and business intelligence. | CO4 |

## Communication & Personality Development

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **Communication & Personality Development** | **AEC007** | 3-0-0 | 3 |
| **Type of Course:** | AEC-2 | | |
| **Contact Hours** | 36 | | |
| **Pre-requisite(s), if any:** | | | |

**Course Perspective.** The course enhances public speaking and presentation skills, helps students confidently convey ideas, information & build self-reliance and competence needed for career advancement. Personality assessments like the Johari Window and Myers & Briggs Type Indicator (MBTI) provide frameworks to enhance self-understanding, helps people increase their self-awareness, understand and appreciate differences in others and apply personality insights to improve their personal and professional effectiveness. Interpersonal skills included in the course deal with important topics like communication, teamwork and leadership, vital for professional success.

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Improving** public speaking and presentation abilities to confidently convey ideas and information. |
| **CO 2** | **Understanding** the framework of Communication to augment oratory skills and written English communication, professional writing, and persuasive communication. |
| **CO 3** | **Cultivating** essential soft skills required at the different workplaces. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number: 1** | **Title: Developing self and others** | **No. of hours: 8** |

**Content Summary:** Self Awareness, Personality Concepts (Personality Assessments -Johari Window, Myers & Brigg), Self-Management, Self Esteem, Self-Efficacy, Interpersonal skills, mindset, grit and working in teams.

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: Enhancing Reading and Writing Skills** | **No. of hours: 6** |

**Content Summary:** Speed reading and its importance in competitive examinations, techniques for speed reading, note-taking, and critical analysis. Paragraph Writing, Essay and Summary writing, Business Letter, Email writing

| | | |
|---|---|---|
| **Unit Number: 3** | **Title: Effective Communication and Public Speaking** | **No. of hours: 7** |

**Content Summary:** Communication Framework, barriers & overcoming these barriers, Group Discussions, Extempore & Public Speaking drills, to manage stage fright and anxiety. Structuring and organizing a presentation (Oral & PPT), Etiquettes, Grooming, Body Language and Conversation starters, TMAY.

| | | |
|---|---|---|
| **Unit Number: 4** | **Title: Career Guide and readiness** | **No. of hours: 15** |

**Content Summary:** Cover Letter, ATS friendly resume, Elevator Pitch, Video Resume (Visume), Networking, Group Discussion, Mock Interviews. Capstone Project

**Learning Experiences**

1. **Interactive Lectures**: Introduce advanced life skills concepts using PPTs and real-life scenarios.
2. **Conceptual Understanding**: Cover topics like emotional intelligence, conflict resolution, and leadership.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on real-world workplace challenges.
4. **Group Discussions**: Facilitate discussions on ethics, diversity, and effective communication in professional settings.
5. **Guest Speakers**: Invite industry leaders to share insights and experiences related to life skills.
6. **Continuous Feedback**: Implement quizzes and peer reviews to assess application of life skills.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign reflective essays on personal development and career aspirations.
2. **Workshops**: Facilitate hands-on sessions for practicing negotiation, networking, and public speaking skills.
3. **Question Bank**: Provide resources for self-assessment on advanced life skills development.
4. **Online Forums**: Create platforms for discussing personal growth and professional challenges.
5. **Self-Study for Case Studies**: Encourage independent research on successful professionals and their life skills.
6. **Collaborative Projects**: Organize group projects focused on community engagement and leadership initiatives.

**Textbooks**

I) Aggarwal, R. S. (2014). Quantitative aptitude (Revised edition).

II) Gladwell, M. (2021). Talking to strangers.

III) Scott, S. (2004). Fierce conversations.

**References**

- "The 7 Habits of Highly Effective People" by Stephen R. Covey
- "Presentation Skills: The Essential Guide for Students" by Joan van Emden and Lucinda Becker
- "Emotional Intelligence: Why It Can Matter More Than IQ" by Daniel Goleman
- "Arithmetic for Competitive Examinations" by R.S. Aggarwal
- "The Art of Public Speaking" by Dale Carnegie

# MINOR PROJECT-II

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name: Minor Project-II** | | **Course Code** | **L-T-P** | **Credits** |
| | | **ENSI252** | **---** | 2 |
| **Type of Course:** | Proj-2 | | | |

**Pre-requisite(s), if any: Fundamentals subjects of Computer Engineering**

**Duration:**

The minor project will last for **three** months.

**Project Requirements:**

1. **Understanding of Societal Problems:**
    - Students must have a basic understanding of societal problems, the concerned domain, and relevant issues.
2. **Critical Thinking and Problem Formulation:**
    - Students are expected to think critically about formulated problems and review existing solutions.
3. **Data Gathering and ETL Activities:**
    - Students should gather relevant data and perform ETL (Extract, Transform, Load) activities to prepare the data for analysis.
4. **Innovation and Entrepreneurship Focus:**
    - Students should develop innovative ideas or entrepreneurial solutions to address the identified problems.
5. **Implementation (Optional):**
    - While implementation of the proposed solutions is encouraged, it is not strictly required. The focus should be on idea development.

**Guidelines:**

1. **Project Selection:**
    - Choose a societal problem relevant to the field of computer science and engineering.
    - Ensure the problem is specific and well-defined.
2. **Literature Review:**
    - Conduct a thorough review of existing literature and solutions related to the problem.
    - Identify gaps in existing solutions and potential areas for further investigation.

3. **Data Gathering and ETL:**
   - Collect relevant data from various sources.
   - Perform ETL activities to clean, transform, and load the data for analysis.
4. **Analysis and Critical Thinking:**
   - Analyze the problem critically, considering various perspectives and implications.
   - Evaluate the effectiveness and limitations of current solutions.
5. **Innovation and Idea Development:**
   - Develop innovative ideas or entrepreneurial solutions to address the identified problem.
   - Focus on the feasibility, impact, and potential of the proposed solutions.
6. **Documentation:**
   - Document the entire process, including problem identification, literature review, data gathering, ETL activities, analysis, and ideas.
   - Use appropriate formats and standards for documentation.
7. **Presentation:**
   - Prepare a presentation summarizing the problem, existing solutions, data analysis, and proposed ideas.
   - Ensure the presentation is clear, concise, and well-structured.

**Evaluation Criteria for Minor Project (Out of 100 Marks):**

1. **Understanding of Societal Problems (15 Marks):**
- Comprehensive understanding of the problem: 15 marks
- Good understanding of the problem: 12 marks
- Basic understanding of the problem: 9 marks
- Poor understanding of the problem: 5 marks
- No understanding of the problem: 0 marks

2. **Critical Thinking and Analysis (20 Marks):**
- Exceptional critical thinking and analysis: 20 marks
- Good critical thinking and analysis: 15 marks
- Moderate critical thinking and analysis: 10 marks
- Basic critical thinking and analysis: 5 marks
- Poor critical thinking and analysis: 0 marks

3. **Data Gathering and ETL Activities (20 Marks):**
- Comprehensive and effective ETL activities: 20 marks
- Good ETL activities: 15 marks
- Moderate ETL activities: 10 marks

- Basic ETL activities: 5 marks
- Poor ETL activities: 0 marks

4. **Innovation and Idea Development (25 Marks):**

- Highly innovative and feasible ideas: 25 marks
- Good innovative ideas: 20 marks
- Moderate innovative ideas: 15 marks
- Basic innovative ideas: 10 marks
- Poor innovative ideas: 5 marks
- No innovative ideas: 0 marks

5. **Documentation Quality (10 Marks):**

- Well-structured and detailed documentation: 10 marks
- Moderately structured documentation: 7 marks
- Poorly structured documentation: 3 marks
- No documentation: 0 marks

6. **Presentation (10 Marks):**

- Clear, concise, and engaging presentation: 10 marks
- Clear but less engaging presentation: 7 marks
- Somewhat clear and engaging presentation: 3 marks
- Unclear and disengaging presentation: 0 marks

**Total: 100 Marks**

**Course Outcomes:**

By the end of this course, students will be able to:

1. **Understand Societal Issues:**
    - Demonstrate a basic understanding of societal problems and relevant issues within the concerned domain.
2. **Critical Thinking:**
    - Think critically about formulated problems and existing solutions.
3. **Data Management:**
    - Gather relevant data and perform ETL activities to prepare the data for analysis.
4. **Innovation and Entrepreneurship:**
    - Develop innovative ideas or entrepreneurial solutions to address identified problems.
5. **Literature Review:**
    - Conduct comprehensive literature reviews and identify gaps in existing solutions.
6. **Documentation:**

- Document findings and analysis in a well-structured and appropriate format.

7. **Presentation Skills:**
   - Present findings and analysis effectively, using clear and concise communication skills.
8. **Problem Analysis:**
   - Analyze problems from various perspectives and evaluate the effectiveness of existing solutions.
9. **Professional Development:**
   - Develop skills in research, analysis, documentation, and presentation, contributing to overall professional growth.

**Learning Experiences**

**Classroom Learning Experience**

1. **Project Kickoff**: Introduce project objectives and expectations through orientation sessions.
2. **Research Methodology**: Cover essential techniques for conducting research and project planning.
3. **Problem-Solving Sessions**: Conduct workshops focused on overcoming project-related challenges.
4. **Progress Presentations**: Facilitate sessions for students to present their project updates and receive feedback.
5. **Group Collaboration**: Encourage teamwork to enhance project development and idea exchange.
6. **Continuous Feedback**: Implement peer reviews and mentor check-ins to assess progress and learning.

**Outside Classroom Learning Experience**

1. **Independent Research**: Assign tasks that require in-depth research and exploration of project topics.
2. **Hands-On Implementation**: Facilitate practical application of project concepts in real-world scenarios.
3. **Documentation**: Encourage students to maintain detailed project logs and documentation.
4. **Online Collaboration Tools**: Create platforms for students to communicate and share resources effectively.
5. **Self-Assessment**: Provide tools for students to evaluate their contributions and project outcomes.
6. **Final Presentation**: Organize sessions for students to present their completed projects to peers and faculty.

# COMPETITIVE CODING -II

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **COMPETITIVE CODING -II** | | 2-0-0 | 0 |
| **Type of Course:** | AUDIT-2 | | |
| **Contact Hours** | 30 | | |

**Course Outcomes**

| | |
|---|---|
| **CO1** | **Understanding** fundamental tree structures, including AVL trees, and their balancing mechanisms. |
| **CO2** | **Applying** graph representations (adjacency matrix and adjacency list) to solve basic graph traversal problems. |
| **CO3** | **Implementing** shortest path algorithms such as Dijkstra's algorithm and Bellman-Ford. |
| **CO4** | **Exploring** dynamic programming concepts, including memoization and tabulation, to solve classic problems. |

| **Unit Number: 1** | **Title: Object-Oriented Programming Concepts** | **No. of hours: 8** |
|---|---|---|
| **Content:**<br>**OOP Basics:** Encapsulation, Inheritance, Polymorphism, Class Design and Object Creation<br>**C++ OOP Concepts:** Classes and Objects, Constructors/Destructors, Operator Overloading, Inheritance, Virtual Functions<br>**Java OOP Concepts:** Classes and Objects, Constructors, Method Overloading, Inheritance, Polymorphism, Abstract Classes, Interfaces<br>**Python OOP Concepts:** Classes and Objects, Constructors, Method Overloading (via default arguments), Inheritance, Polymorphism, Multiple Inheritance | | |

| | | |
|---|---|---|
| **Unit Number: 2** | **Title:** Linked Lists, Stacks and Queues | **No. of hours: 8** |
| **Content:**<br>Linked Lists<br>▪ Singly and doubly linked lists: Creation, insertion, deletion, traversal.<br>▪ Key Problems: Reversing a linked list (iterative and recursive), detecting cycles using Floyd's cycle-finding algorithm.<br>Stacks and Queues :<br>▪ Stack operations: Push, pop, top, isEmpty.<br>▪ Queue operations: Enqueue, dequeue, front, isEmpty.<br>▪ Applications: Parentheses matching, queue-based problems (LeetCode challenge - sliding window problems). | | |
| **Unit Number: 3** | **Title: Sorting & Searching** | **No. of hours: 8** |
| **Content**<br>Basic Sorting Algorithms<br>• Implementing Bubble Sort, Selection Sort, Insertion Sort.<br>• Understanding the time complexities and use cases of each algorithm.<br>• Key Problems: Sorting small arrays, finding the median, custom sorting based on conditions (frequent LeetCode challenge).<br>Advanced Sorting Algorithms<br>• Implementing Merge Sort, Quick Sort, Heap Sort.<br>Binary Search<br>• Implementing binary search for sorted arrays.<br>• Applications: Finding an element in a sorted array, finding the position to insert an element, LeetCode challenges like searching for ranges. | | |
| **Unit Number: 4** | **Title: Trees** | **No. of hours: 6** |
| **Content:**<br>**Basic Tree Concepts** | | |

- Introduction to tree terminology and operations.
- Tree Traversals: Preorder, inorder, postorder.
- Key Problems: Printing all elements in a tree, finding the depth of a tree.

**Binary Trees**

- Basic operations on binary trees: Insertion, deletion, searching.
- Key Problems: Finding the height of a binary tree, counting leaf nodes, lowest common ancestor (common LeetCode challenges).

**Binary Search Trees**

Understanding BST properties: Every left subtree is smaller, and every right subtree is larger.

**Learning Experiences:**

**Classroom Learning Experience**

1. **Advanced Lectures**: Introduce complex algorithms and techniques using PPTs and coding demonstrations.
2. **Algorithm Workshops**: Cover advanced topics like dynamic programming, graph algorithms, and optimization strategies.
3. **Intensive Problem-Solving Sessions**: Conduct in-class exercises focused on challenging competitive coding problems.
4. **Mock Contests**: Organize timed coding competitions to simulate real contest environments.
5. **Group Strategy Discussions**: Facilitate discussions on effective problem-solving strategies and approaches.
6. **Continuous Feedback**: Implement performance assessments and code reviews to enhance skills.

**Outside Classroom Learning Experience**

1. **Practice Assignments**: Assign challenging coding problems for independent practice from various platforms.
2. **Online Competitions**: Encourage participation in external coding contests and hackathons.
3. **Question Bank**: Provide a repository of advanced practice problems for self-assessment.
4. **Online Collaboration**: Create forums for students to discuss problems and share solutions.
5. **Self-Study Resources**: Recommend books and online courses focused on advanced algorithms and techniques.
6. **Collaborative Projects**: Organize group projects that involve developing coding solutions for complex problems.

**Textbooks:**

- “Introduction to Algorithms” by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
- “Data Structures and Algorithms Made Easy” by Narasimha Karumanchi

**Online References:**

1. **GeeksforGeeks**:
   - Offers articles on advanced data structures like self-balancing trees, segment trees, tries, and more[1].
   - Link to GeeksforGeeks
2. **Coursera**:
   - Various data structures and algorithms courses available online.
   - Examples include “Data Structures and Algorithms” from the University of California San Diego and “Algorithms, Part I” from Princeton University[3].
   - Link to Coursera
3. **Princeton University References**:
   - Provides a list of seminal papers and advanced resources.
   - Includes textbooks like “Algorithms, 4th Edition” by Robert Sedgewick and Kevin Wayne[4].

## Lab Experiments

| **Problem Statement** | **Mapped COs** |
|---|---|
| **Object-Oriented Programming Concepts** | |
| 1. Design a Parking Lot System using OOP concepts (Classes, Objects, Inheritance, Polymorphism). | CO1 |
| 2. Implement a Student Management System with Classes and Objects. | CO1 |
| 3. Create a Banking System with Constructors and Destructors. | CO1 |
| 4. Implement Method Overloading and Overriding in a chosen language. | CO1 |
| 5. Demonstrate Multiple Inheritance with a practical example. | CO1 |
| 6. Design a Library Management System with OOP principles. | CO1 |
| 7. Use Virtual Functions to implement polymorphism. | CO1 |
| 8. Implement Abstract Classes and Interfaces for a Payment System. | CO1 |

| **Problem Statement** | **Mapped COs** |
|---|---|
| 9. Create a simple calculator with Operator Overloading. | CO1 |
| 10. Build a Polymorphic class hierarchy (e.g., Shapes) to showcase polymorphism. | CO1 |
| **Linked Lists, Stacks, and Queues** | |
| 11. Reverse a Linked List (Iterative and Recursive). | CO2 |
| 12. Detect a cycle in a Linked List using Floyd's Cycle-Finding Algorithm. | CO2 |
| 13. Implement basic operations on a Singly Linked List (Insertion, Deletion). | CO2 |
| 14. Implement and traverse a Doubly Linked List. | CO2 |
| 15. Implement Stack operations (Push, Pop, Top) using arrays or linked lists. | CO2 |
| 16. Implement Queue operations (Enqueue, Dequeue, Front) using arrays or linked lists. | CO2 |
| 17. Solve the Parentheses Matching problem using Stack. | CO2 |
| 18. Implement Sliding Window Maximum using Deque. | CO2 |
| 19. Check for balanced parentheses using Stack. | CO2 |
| 20. Design a Circular Queue using linked list or array. | CO2 |
| **Sorting & Searching** | |
| 21. Implement Bubble Sort and analyze its time complexity. | CO3 |
| 22. Implement Merge Sort to sort an array of integers. | CO3 |
| 23. Find the Kth largest element in an array using Quick Sort. | CO3 |
| 24. Perform Binary Search to find an element in a sorted array. | CO3 |
| 25. Implement Heap Sort to sort a list of elements. | CO3 |
| 26. Find the position to insert an element in a sorted array using Binary Search. | CO3 |
| 27. Implement a custom sort based on frequency of elements. | CO3 |
| 28. Compare sorting results using Insertion Sort and Bubble Sort. | CO3 |
| **Trees** | |
| 29. Perform Preorder, Inorder, and Postorder Traversal on a Binary Tree. | CO4 |
| 30. Find the Lowest Common Ancestor in a Binary Search Tree. | CO4 |

| Problem Statement | Mapped COs |
|---|---|
| **Problem Statement** | **Mapped COs** |
| **Trees** | |
| 31. Implement an algorithm to check if a Binary Tree is balanced. | CO4 |
| 32. Determine if two Binary Trees are identical. | CO4 |
| 33. Find the maximum path sum in a Binary Tree. | CO4 |
| 34. Convert a Binary Search Tree to a Greater Tree (where each node's value is replaced by the sum of all greater values). | CO4 |
| 35. Count the number of nodes in a complete Binary Tree. | CO4 |
| 36. Flatten a Binary Tree to a linked list using preorder traversal. | CO4 |
| 37. Serialize and deserialize a Binary Tree. | CO4 |
| 38. Find the diameter of a Binary Tree (the longest path between any two nodes). | CO4 |
| 39. Check if a Binary Tree is a subtree of another Binary Tree. | CO4 |
| 40. Find the level order traversal of a Binary Tree (Breadth-First Search). | CO4 |

## Semester: V

# THEORY OF COMPUTATION

|  |  |  |  |  |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name: Theory of Computation** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCS301** | 3-1-0 | 4 | 40 |
| **Type of Course:** | Major-23 | | | |

**Pre-requisite(s), if any: Basic Mathematics and Programming Concepts**

**Course Perspective:** The course provides a comprehensive foundation in the theoretical aspects of computer science, essential for understanding the underlying principles of various computational processes and languages. This course delves into the formalization and analysis of computation, encompassing finite automata, pushdown automata, context-free grammars, Turing machines, and the Chomsky hierarchy. The course is divided into 4 modules:

a) Introduction to Finite Automata
b) Pushdown Automata and Context-Free Languages
c) Chomsky Hierarchy and Turing Machines
d) Code Generation and Optimization

**The Course Outcomes (COs).** On completion of the course the participants will be able to:

| COs | Statements |
|---|---|
| **CO 1** | **Remembering** the fundamental concepts and terminology of automata theory. |
| **CO 2** | **Understanding** the relationships and equivalences between various computational models. |
| **CO 3** | **Applying** conversion techniques between different forms of automata and grammars. |
| **CO 4** | **Analyzing** the properties and limitations of formal languages using theoretical tools. |
| **CO 5** | **Evaluating** the decidability and complexity of computational problems. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number: 1** | **Title: Introduction to Finite automata** | **No. of hours: 10** |

**Content:**

**Finite Automata**: Review of Automata, Description of Finite automata, representation of FA, Deterministic Finite Automata(DFA), Non-deterministic Finite Automata(NFA),Equivalence of NFA and DFA Finite Automata with Epsilon Transitions, Minimization of Deterministic Finite Automata

**Finite Automata with output: -** Moore machine and Mealy Machine, Conversion of Moore machine to Mealy Machine & Vice-Versa

Applications of Finite Automata

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: Regular Expression and Languages** | **No. of hours: 10** |

**Content:**

**Regular Expressions:** Introduction, Identities of Regular Expressions, Arden's theorem state and prove

**Finite Automata and Regular Expressions:** Converting from DFA's to Regular Expressions and Vive-Versa

**Pumping Lemma for Regular Sets**: Introduction, Applications of the pumping lemma- Proving languages not to be regular, Closure properties of regular languages

**Introduction to Formal languages:** Definition of a Grammar, Derivations and the Language Generated by a Grammar, Chomsky Classification of Languages

| | | |
|---|---|---|
| **Unit Number: 3** | **Title: Context-Free Languages and Pushdown Automata (PDA)** | **No. of hours: 12** |

**Content:**

**Context Free Grammar (CFG):** Properties of context free grammar, Derivations using a grammar, Parse Trees, Ambiguity in context free grammar

**Simplification of Context Free grammar:** Reduced grammar, Removal of useless Symbols and unit production

**Normal Forms of CFG:** Chomsky Normal Form (CNF), Greibach Normal Form (GNF)

Pumping lemma for CFG.

**Push down Automata (PDA):** Definition, acceptance by PDA, Types of PDA: Deterministic PDA, Non-Deterministic PDA

Equivalence of CFL and PDA, interconversion

**Unit Number: 4 Title: Turing Machine and Undecidability No. of hours: 8**

**Content Summary:**

**Turing Machines:** Definition, types, and language acceptors, Design of Turing Machines

Universal Turing Machine and its implications

Decidability and Undecidability

Halting problem of Turing Machine, Post-Correspondence Problem.

Properties of Recursive and Recursively Enumerable Languages

**Learning Experience**

**Classroom Learning Experience**

1. **Interactive Lectures**: Introduce key concepts in the theory of computation using PPTs and examples.
2. **Conceptual Understanding**: Cover topics like automata theory, formal languages, and Turing machines.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on designing automata and proving language properties.
4. **Case Studies**: Analyze real-world applications of computational theory in computer science.
5. **Group Discussions**: Facilitate discussions on complexity theory and computability.
6. **Continuous Feedback**: Implement quizzes and peer reviews to assess understanding of theoretical concepts.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign take-home projects that require applying theoretical concepts to practical problems.
2. **Lab Projects**: Facilitate hands-on tasks involving simulations of automata and formal languages.
3. **Question Bank**: Provide practice problems and resources for self-assessment on computation theory topics.
4. **Online Forums**: Create platforms for discussing challenges and sharing solutions related to computation theory.

5. **Self-Study for Case Studies**: Encourage independent research on advancements and applications in computation theory.
6. **Collaborative Projects**: Organize group projects focused on exploring complex theoretical concepts and their implications.

**Text Books:**

1. Hopcroaft J.E., Ullman, J.D., and Rajiv Motwani, 2001, Introduction to Automata Theory, Language & Computations, 3rdEd.
2. Mishra K.L.P.& N. Chandrasekaran, 2000, Theory of Computer Science Automata, Languages and Computation,5th Ed. , 2000, PHI

**Reference Books**

1. H.R. Lewis and C.H. Papadimitriou, "Elements of the theory of Computation", Second Edition, Pearson Education.
2. Peter Linz, 2001, Introduction to formal Languages & Automata, 3rd Ed., NarosaPubl.
3. J. Martin, "Introduction to Languages and the Theory of computation" Third Edition, Tata Mc Graw Hill.

**Additional Readings:**

**Online Learning Resources for "Theory of Computation"**

1. **NPTEL - Theory of Computation**
   - Online course by IITs on the fundamentals of the theory of computation.
   - Link: NPTEL-Theory of Computation
2. **Coursera - Automata Theory by Stanford University**
   - This course covers the fundamentals of automata theory, including finite automata, regular expressions, and Turing machines.
   - Link: Coursera - Automata Theory
3. **MIT OpenCourseWare - Theory of Computation**
   - Advanced course materials from MIT covering various topics in the theory of computation.
   - Link: MIT OpenCourseWare - Theory of Computation

# OPERATING SYSTEMS

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name: OPERATING SYSTEMS** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCS303** | 4-0-0 | 4 | 40 |
| **Type of Course:** | Major-24 | | | |

**Pre-requisite(s), if any:** Basics of programming and Computer Fundamentals

**Course Perspective.** This course provides a comprehensive introduction to the fundamental principles and practices of operating systems. It covers essential concepts such as process management, memory management, file systems, and I/O systems, as well as more advanced topics like distributed operating systems and concurrent systems. Through this course, students will gain a deep understanding of how operating systems function, how they manage hardware resources, and how they provide services to applications. The course also emphasizes practical skills in implementing and managing operating system components and handling challenges such as process synchronization, deadlocks, and system security. By the end of the course, students will be well-equipped to apply these concepts in designing and optimizing operating systems in various computing environments. The course is divided into 4 modules:

a) Introduction to Operating Systems and Process

b) Memory & File Management

c) Process Synchronization, Deadlocks & I/O Systems

d) Distributed Operating Systems & Concurrent Systems

**The Course Outcomes (COs).** On completion of the course the participants will be able to:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the fundamental concepts of operating systems, including their structure and types. |
| **CO 2** | **Analyzing** process scheduling algorithms and their impact on system performance. |
| **CO 3** | **Implementing** and manage memory allocation, paging, and virtual memory techniques. |

| | |
|---|---|
| **CO 4** | **Examining** process synchronization mechanisms and handle deadlocks in an operating system environment. |
| **CO 5** | **Developing** distributed operating systems and concurrent systems with a focus on fault tolerance and recovery mechanisms. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number: 1** | **Title: Introduction to Operating System, Process and CPU Scheduling** | **No. of hours: 10** |

**Introduction:** Definition, Role, Types of Operating System, Batch Systems, multi programming, time–sharing, parallel, distributed and real-time systems, Operating system structure, Operating system components and services, System calls, System programs, Virtual machines.

**Processes:** Process Concept, Process Scheduling, Operation on Processes, Cooperating Processes, Threads.

**CPU Scheduling:** Basic Concepts, Scheduling Criteria, Scheduling Algorithms, Multiple Processor Scheduling, Real-Time Scheduling.

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: Threads, Synchronization, Deadlock and Memory Management** | **No. of hours: 10** |

**Threads:** overview, Benefits of threads, User and kernel threads, Multithreaded Models, Precedence Graph, Fork-Join, Cobegin-Coend construct.

**Inter-process Communication and Synchronization**: Background, The Critical-Section Problem, Synchronization Hardware, Semaphores, Classical Problems of Synchronization, Critical Regions, Monitors, Message Passing.

**Deadlocks:** System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock.

**Memory Management:** Background, Logical vs. Physical Address space, swapping, Contiguous allocation, Paging, Segmentation, Segmentation with Paging.

| | | |
|---|---|---|
| **Unit Number: 3** | **Title: Virtual Memory, Device Management and Secondary-Storage Structure** | **No. of hours: 10** |

**Virtual Memory:** Demand Paging and its performance, Page-replacement Algorithms, Allocation of Frames, Thrashing, page size and other Considerations, Demand Segmentation.

**Device Management:** Techniques for Device Management, Dedicated Devices, Shared Devices, Virtual Devices, Independent Device Operation, Buffering, Device Allocation Consideration.

**Secondary-Storage Structure:** Disk Structure, Disk Scheduling, Disk Management, Swap Space Management, Disk Reliability.

| | | |
|---|---|---|
| **Unit Number: 4** | **Title: File-System Interface, implementation and Security** | **No. of hours: 10** |

**File-System Interface:** File Concept, Access Methods, Directory Structure.

**File-System Implementation:** Introduction, File-System Structure, Basic File System, Allocation Methods, Free-Space Management, Directory Implementation.

**Security:** Security problems, Goals of protection, Access matrix, Authentication, Program threats, System threats, Intrusion detection.

## Learning Experiences

- **Interactive Lectures**: Introduce key concepts of operating systems using PPTs and real-world examples.
- **Conceptual Understanding**: Cover topics like process management, memory management, and file systems.
- **Problem-Solving Sessions**: Conduct in-class exercises focused on system calls and scheduling algorithms.
- **Case Studies**: Analyze real-world operating systems and their architectures.
- **Group Work**: Collaborate on projects that involve designing and implementing operating system components.
- **Continuous Feedback**: Implement quizzes and peer reviews to assess understanding of operating system principles.

- **Outside Classroom Learning Experience**
- **Theory Assignments**: Assign take-home projects requiring application of operating system concepts to practical scenarios.
- **Lab Projects**: Facilitate hands-on tasks involving system programming and OS simulations.
- **Question Bank**: Provide practice problems and resources for self-assessment on operating system topics.
- **Online Forums**: Create platforms for discussing operating system challenges and sharing solutions.

- **Self-Study for Case Studies**: Encourage independent research on current trends and technologies in operating systems.
- **Collaborative Projects**: Organize group projects focused on solving real-world problems using operating system concepts.

**Textbooks**

1. Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Wiley
2. Modern Operating Systems, Andrew S. Tanenbaum and Herbert Bos, Pearson, 4th Edition, 2014.
3. Operating Systems: Internals and Design Principles, William Stallings, Pearson, 9th Edition, 2017.
4. Operating Systems: Three Easy Pieces, Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau Arpaci-Dusseau Books, 1st Edition, 2018

**References**

1. MukeshSinghal and N. G. Shivaratri, “Advanced Concepts in Operating Systems”, McGrawHill, 2000
2. Abraham Silberschatz, Peter B. Galvin, G. Gagne, “Operating System Concepts”, Sixth Addison Wesley Publishing Co., 2003.
3. Andrew S. Tanenbaum, “Modern Operating Systems”, Second Edition, Addison Wesley, 2001.
4. Tannenbaum, “Operating Systems”, PHI, 4th Edition.

**Additional Readings:**

**Online Learning References :**

I) **MIT OpenCourseWare - Operating System Engineering**
   a. Advanced course materials from MIT covering various topics in operating system design and implementation.
   b. Link: MIT OpenCourseWare - Operating System Engineering

II) **NPTEL - Operating System by IITs**
   a. Online course by IITs providing in-depth coverage of operating system principles and practices.
   b. Link: NPTEL - Operating System

# OPERATING SYSTEM LAB

**Program Name:** **B. Tech (Computer Science and Engineering)**

| | | | |
|---|---|---|---|
| **Course Name: OPERATING SYSTEMS LAB** | **Course Code** | **L-T-P** | **Credits** |
| | **ENCS351** | 0-0-2 | 1 |

**Type of Course:** Major-25

**Pre-requisite(s), if any:** Basics of programming

**Defined Course Outcomes**

| **COs** | **Statements** |
|---|---|
| CO 1 | **Implementing** and analyze process creation, management, and CPU scheduling algorithms, demonstrating the ability to simulate an operating system environment. |
| CO 2 | **Developing** and evaluate multithreaded applications, demonstrating synchronization, deadlock handling, and memory management techniques. |
| CO 3 | **Simulating** virtual memory management, device management, and disk scheduling algorithms, showcasing the application of operating system concepts. |
| CO 4 | **Designing** and implement secure file systems, demonstrating file operations, directory management, and access control mechanisms. |

## List of Experiments

| **Ex No** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| 1 | Implement a program that simulates system calls for basic operations such as process creation, file manipulation, and device management. Demonstrate how system calls interact with the operating system components and services. | CO1 |
| 2 | Develop a process scheduling simulation that demonstrates different CPU scheduling algorithms (FCFS, SJF, Round Robin, Priority Scheduling). Compare the performance of each algorithm based on scheduling criteria such as turnaround time, waiting time, and response time. | CO1 |
| 3 | Create a multi-threaded application to illustrate process operations, including creation, termination, and inter-process communication. Implement thread | CO1 |

| | | |
|---|---|---|
| | management to demonstrate the concept of cooperating processes and the benefits of threading. | |
| 4 | Implement a multi-threaded program to demonstrate the benefits of threads over single-threaded processes. Use different multithreading models such as user-level and kernel-level threads and simulate various thread operations. | CO2 |
| 5 | Design and implement solutions for classical synchronization problems such as the Producer-Consumer problem, Readers-Writers problem, and Dining Philosophers problem using semaphores, critical regions, and monitors. | CO2 |
| 6 | Create a simulation to detect and handle deadlocks in a system. Implement deadlock prevention, avoidance, and detection algorithms. Demonstrate recovery from deadlock scenarios. | CO2 |
| 7 | Develop a memory management simulator that demonstrates different memory allocation techniques such as contiguous allocation, paging, and segmentation. Implement swapping and address translation between logical and physical address spaces. | CO2 |
| 8 | Implement a demand paging system to simulate virtual memory management. Evaluate the performance of different page-replacement algorithms (FIFO, LRU, Optimal) and analyze the effects of thrashing. | CO3 |
| 9 | Create a simulation for device management that includes buffering, device allocation, and handling dedicated, shared, and virtual devices. Demonstrate techniques for independent device operation and management. | CO3 |
| 10 | Develop a disk scheduling simulator to compare the performance of different disk scheduling algorithms (FCFS, SSTF, SCAN, C-SCAN). Implement disk management techniques and swap space management. | CO3 |
| 11 | Implement a file system simulator to demonstrate different file access methods (sequential, direct, indexed). Design a directory structure and simulate file operations such as creation, deletion, reading, and writing. | CO4 |
| 12 | Develop a program to simulate file system implementation techniques, including different file allocation methods (contiguous, linked, indexed) and free-space management techniques. Implement a basic file system and directory structure. | CO4 |

## (DEPARTMENT ELECTIVE-I)
## SECURE CODING AND VULNERABILITIES

**Program Name:** **B. Tech (Computer Science and Engineering)**

| **Course Name:** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
|---|---|---|---|---|
| **Secure Coding & Vulnerabilities** | **ENSP301** | 4-0-0 | 4 | 40 |

**Type of Course:** DSE-1

**Pre-requisite(s), if any: Fundamentals of Programming and Networks**

**Course Perspective:** This course provides an in-depth exploration of secure coding practices and the identification and mitigation of common vulnerabilities in software development. Students will gain a solid foundation in security concepts, secure application design, and the implementation of security best practices throughout the software development lifecycle. By understanding the principles of secure coding and the types of vulnerabilities that can compromise applications, students will be equipped to develop robust, secure software. The course covers essential topics such as input validation, authentication, cryptography, buffer overflows, SQL injection, and application security testing. The course is divided into four modules:

a) Introduction to Coding and Security
b) Secure Application Design and Architecture
c) Secure Coding Practices and Vulnerabilities
d) Application Security Testing and Deployment

**The Course Outcomes (COs).** On completion of the course the participants will be able to:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** different types of application security threats and their potential impact. |
| **CO 2** | **Applying** secure design principles and architectures to develop robust and secure applications. |
| **CO 3** | **Implementing** secure coding practices for input validation, authentication, cryptography, session management, and error handling. |
| **CO 4** | **Conducting** static and dynamic application security testing to identify vulnerabilities and implement secure deployment and maintenance practices. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

**Unit Number: 1 Title: Introduction to coding and Security No. of hours: 10**

**Content**

Introduction-security concepts-CIA Triad, Viruses, Trojans, and Worms, threat, vulnerability, risk, attack. Coding Standards: Dirty Code and Dirty Compiler, Dynamic Memory Management functions, Common memory management Errors (Initialization Errors, Forget to Check Return Values, accessing already freed memory, Freeing the same memory multiple times, Forget to free the allocated memory), Integer Security –Introduction to integer types: Integer Data Types, data type conversions, Integer vulnerabilities and mitigation strategies

**Unit Number: 2 Title: Secure Application Design and Architecture No. of hours: 10**

**Content** Security requirements gathering and analysis, Secure software development life cycle (SSDLC), Security issues while writing SRS, Design phase security, Development Phase, Test Phase, Maintenance Phase, Writing Secure Code – Best Practices SD3 (Secure by design, default and deployment), Security principles and Secure Product Development Timeline.

**Unit Number: 3 Title: Secure Coding Practices and Vulnerabilities No. of hours: 10**

**Content** Input validation Techniques-whitelist validation, regular expressions, authentication and authorization, Cryptography, buffer overflows, Session management and protection against session-related attacks, Secure error handling and logging practices, SQL Injection Techniques and Remedies, Race conditions

| | | |
|---|---|---|
| **Unit Number: 4** | **Title: Application Security Testing and Deployment** | **No. of hours: 10** |

**Content** Security code overview, Secure software installation. The Role of the Security Tester, Building the Security Test Plan. Testing HTTP-Based Applications, Testing File-Based Applications, Testing Clients with Rogue Servers, Static and Dynamic Application Security Testing (SAST & DAST), Secure Deployment and Maintenance, Patch management and software updates, Vulnerability scanning and penetration testing.

**Learning Experiences:**

**Classroom Learning Experience**

1. **Hands-on Vulnerability Testing**: Practice identifying and mitigating common software vulnerabilities through hands-on exercises.
2. **Code Review Sessions**: Conduct peer code reviews to spot potential security flaws and enhance secure coding practices.
3. **Case Studies**: Analyze real-world security breaches to understand the exploitation of vulnerabilities.
4. **Interactive Labs**: Implement secure coding techniques such as input validation and buffer overflow protection.
5. **Security Audits**: Perform security audits on sample applications to assess vulnerabilities like SQL injection and session hijacking.
6. **Role Play**: Simulate attacker and defender roles in vulnerability exploitation and mitigation scenarios.

**Outside Classroom Learning Experience**

1. **Project-Based Learning**: Develop secure applications by applying best practices in secure design, coding, and testing.
2. **Tools Exploration**: Learn to use static and dynamic application security testing tools (SAST & DAST) for real-world applications.
3. **Collaborative Learning**: Work in groups to design security testing plans and assess security risks in various application environments.

4. **Real-World Simulations**: Conduct vulnerability scanning and penetration testing in simulated deployment environments.

**References**

1. Secure Coding: Principles and Practices, Mark G. Graff, Kenneth R. Van Wyk, O'Reilly Media
2. Writing Secure Code, Michael Howard and David LeBlanc, Microsoft Press, 2nd Edition, 2004
3. Buffer Overflow Attacks: Detect, Exploit, Prevent by Jason Deckard ,Syngress,1st Edition, 2005
4. Threat Modeling, Frank Swiderski and Window Snyder,Microsoft Professional, 1st Edition ,2004
5. Secure Coding: Principles and Practices by Mark G. Graff, Kenneth R. van Wyk, Publisher(s): O'Reilly Media, Inc., 2003
6. The Software Vulnerability Guide (Programming Series) by H. Thompson (Author), Scott G. Chase, 2005

**Additional Readings:**

**Online Learning References for "Secure Coding and Vulnerabilities"**

1. **OWASP - Secure Coding Practices - Quick Reference Guide**
   - This guide provides a quick reference to secure coding practices based on OWASP's recommendations for secure software development.
   - Link: OWASP - Secure Coding Practices
2. **NPTEL - Secure Coding**
   - Offered by IITs through NPTEL, this course covers secure coding practices and principles for writing secure software.
   - Link: NPTEL - Secure Coding
3. **Mozilla Developer Network (MDN) - Web Security**
   - Comprehensive documentation on web security principles, secure coding practices, and common vulnerabilities in web applications.
   - Link: MDN - Web Security
4. **Google Code University - Web Security**
   - Learn about web security from Google, including secure coding practices and how to protect web applications from common threats.
   - Link: Google Code University - Web Security

# SECURE CODING AND VULNERABILITIES LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **Secure Coding & Vulnerabilities Lab** | **ENSP351** | 0-0-2 | 1 |
| **Type of Course:** | DSE-2 | | |

**Pre-requisite(s), if any: Fundamentals of Programming and Networks**

**Defined Course Outcomes**

| COs | **Statements** |
|---|---|
| CO 1 | **Implementing** fundamental security concepts such as the CIA Triad (Confidentiality, Integrity, and Availability) and demonstrate secure coding practices to prevent common vulnerabilities. |
| CO 2 | **Analyzing** and fix memory management errors and integer vulnerabilities, applying mitigation strategies to enhance software security. |
| CO 3 | **Developing** secure software by following the Secure Software Development Life Cycle (SSDLC), incorporating security principles and best practices throughout the development process. |
| CO 4 | **Designing** and test secure applications, performing vulnerability scanning, penetration testing, and implementing security measures to protect against attacks such as SQL injection and buffer overflow. |

**Lab Experiments**

| **Ex. No** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| P1 | **Project Title: Secure Memory Management System**<br><br>**Problem Statement:** Develop a secure memory management system for a critical application such as a healthcare management system. This system should handle dynamic memory allocation and de-allocation securely, preventing common memory management vulnerabilities. | CO1 |

| | | |
|---|---|---|
| P2 | **Project Title: Secure E-commerce Platform Design**<br><br>**Problem Statement:** Design and implement a secure e-commerce platform that ensures data security throughout the software development life cycle (SDLC). The platform should handle sensitive user information securely and provide a robust security architecture. | CO2 |
| P3 | **Project Title: Secure Banking Application**<br>**Problem Statement:** Develop a secure online banking application that ensures the protection of user data and prevents common vulnerabilities such as SQL injection, buffer overflow, and session hijacking. | CO3 |
| P4 | **Project Title: Comprehensive Security Testing and Deployment for a Social Media Platform**<br><br>**Problem Statement:** Develop a social media platform with a focus on security testing and secure deployment. The platform should protect user data and provide a secure environment for social interactions. | CO4 |

# CYBER CRIME INVESTIGATION & DIGITAL FORENSICS

**Program Name:** **B. Tech (Computer Science and Engineering)**

| **Course Name:** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
|---|---|---|---|---|
| **Cyber Crime Investigation & Digital Forensics** | **ENSP303** | 4-0-0 | 4 | 40 |

**Type of Course:** DSE-1

**Pre-requisite(s), if any: Basics of Cybersecurity**

**Course Perspective**: The course offers an in-depth exploration of the methodologies and techniques employed in identifying, investigating, and prosecuting cybercrimes. As digital technologies permeate every aspect of modern life, understanding how to safeguard and investigate electronic evidence becomes crucial for ensuring security and justice. This course covers the foundational concepts of digital forensics, types of cybercrimes, investigation procedures, and the utilization of forensic tools. It prepares students to handle and analyze digital evidence proficiently, contributing to the effective enforcement of cyber laws. The course is divided into four comprehensive units:

a) Introduction
b) Types of Cyber Crimes
c) Investigation of Cyber Crimes
d) Forensic Tools and Processing of Electronic Evidence

**The Course Outcomes (COs).** On completion of the course the participants will be able to:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the nature and classification of conventional and cyber-crimes. |
| **CO 2** | **Analyzing** various types of cyber-crimes and their modes of operation. |
| **CO 3** | **Evaluating** the impact of cyber-crimes on individuals, organizations, and society. |
| **CO 4** | **Developing** an understanding of digital forensics and the investigative procedures used in cyber-crime cases. |

| | |
|---|---|
| **CO 5** | **Applying** forensic tools and techniques to retrieve and analyze digital evidence. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

**Unit Number: 1 Title: Title: Introduction No. of hours: 10**

**Content:** Introduction to Digital Forensics, Definition and types of cybercrimes, electronic evidence and handling, electronic media, collection, searching and storage of electronic media, introduction to internet crimes, hacking and cracking, credit card and ATM frauds, web technology, cryptography, emerging digital crimes and modules.

**Unit Number: 2 Title: Types of Cyber Crimes No. of hours: 10**

**Content:** Crimes targeting Computers: Unauthorized Access Packet Sniffing Malicious Codes including Trojans, Viruses, Logic Bombs, etc. Online based Cyber Crimes: Phishing and its variants Web Spoofing and E-mail Spoofing Cyber Stalking Web defacement financial crimes, ATM and Card Crimes etc. Spamming Commercial espionage and Commercial Extortion online Software and Hardware Piracy Money Laundering Fraud& Cheating Other Cyber Crimes.

**Unit Number: 3 Title: Investigation of Cyber Crimes No. of hours: 10**

**Content:** Investigation of malicious applications Agencies for investigation in India, their powers and their constitution as per Indian Laws Procedures followed by First Responders; Evidence Collection and Seizure Procedures of Digital mediums Securing the Scene, Documenting the Scene, Evidence Collection and Transportation Data Acquisition Data Analysis Reporting

**Unit Number: 4 Title: Forensic Tools and Processing of Electronic Evidence No. of hours: 10**

**Content:**Introduction to Forensic Tools, Usage of Slack space, tools for Disk Imaging, Data Recovery, Vulnerability Assessment Tools, Encase and FTK tools, Anti Forensics and probable counters, retrieving information, process of computer forensics and digital investigations, processing of digital evidence, digital images, damaged SIM and data recovery, multimedia evidence, retrieving deleted data: desktops, laptops and mobiles, retrieving data from slack space, renamed file, ghosting, compressed files.

**L1= Remember, L2= Understand, L3= Apply, L4= Analyze, L5= Evaluate and L6= Create**

**Learning Experiences**

**Classroom Learning Experience**

1. **Interactive Lectures and Video Sessions**: Engage with interactive presentations and videos on cyber crime and digital forensics.
2. **Problem-Based Theory Assignments**: Analyze real-world cyber crime scenarios to encourage complex problem-solving.
3. **Project-Based Lab Assignments**: Use forensic tools in hands-on labs to investigate simulated cyber crimes.
4. **Collaborative Group Work**: Work in groups on case studies to promote teamwork and peer learning.
5. **Continuous Assessment and Feedback**: Monitor progress through assessments with regular instructor feedback.

**Outside Classroom Learning Experience**

1. **Use of ICT Tools and Moodle LMS**: Access course materials via Moodle and use interactive boards for discussions.
2. **Engagement with a Question Bank and Model Papers**: Utilize a question bank and model papers for exam preparation.
3. **Application of Theoretical Knowledge to Practical Scenarios**: Apply theory to practical cases in the full cycle of cyber crime investigations.

**References**

1. Moore, Robert, (2011). Cybercrime, investigating high-technology computer crime(2nd Ed.). Elsevie
2. C. Altheide& H. Carvey Digital Forensics with Open Source Tools, Syngress, 2011.
3. Majid Yar, "Cybercrime and Society", SAGE Publications Ltd, Hardcover, 2nd Edition, 2013.
4. Robert M Slade, "Software Forensics: Collecting Evidence from the Scene of a Digital Crime", Tata McGraw Hill, Paperback, 1st Edition, 2004.

**Additional Readings:**

**Online Learning References:**

I) **Cybrary - Digital Forensics**
   a. A free online course that covers various aspects of digital forensics, including tools, techniques, and procedures for investigating cybercrimes.
   b. Link: Cybrary - Digital Forensics

II) **Pluralsight - Digital Forensics Fundamentals**

   a. This course offers a thorough understanding of digital forensics, covering the fundamentals, tools, and techniques used in the field.
   b. Link: Pluralsight - Digital Forensics Fundamentals

III) **SANS Institute - Digital Forensics and Incident Response Blog**
   a. A blog providing insights, case studies, and updates on the latest in digital forensics and incident response.
   b. Link: SANS Institute - Digital Forensics Blog

IV) **OWASP - Open Web Application Security Project**
   a. Provides resources on web security, including best practices for secure coding and tools for vulnerability assessment, which are essential for investigating cybercrimes.
   b. Link: OWASP - Open Web Application Security Project

# CYBER CRIME INVESTIGATION & DIGITAL FORENSICS LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** **Cyber Crime Investigation & Digital Forensics Lab** | **Course Code** **ENSP353** | **L-T-P** 0-0-2 | **Credits** 1 |
| **Type of Course:** | DSE-2 | | |
| **Pre-requisite(s), if any: Basics of Cybersecurity** | | | |

**Defined Course Outcomes**

| COs | **Statements** |
|---|---|
| CO 1 | **Understanding** the fundamental concepts and principles of digital forensics and cybercrimes. |
| CO 2 | **Applying** the knowledge of digital forensics techniques and procedures to collect, analyse, and preserve electronic evidence in various types of cybercrimes. |
| CO 3 | **Evaluating** and utilize forensic tools and technologies for data acquisition, analysis, and recovery in the investigation of cybercrimes. |
| CO 4 | **Analyzing** and interpret digital evidence obtained from different sources, such as electronic media, internet crimes, malicious applications, and various forms of cybercrimes. |

## LAB EXPERIMENTS

| **Ex. No** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| 1 | **Project Title: Comprehensive Study on Cybercrime and Digital Forensics** **Problem Statement:** Conduct a comprehensive study on various types of cybercrimes and the role of digital forensics in investigating these crimes. The project will involve collecting electronic evidence, understanding cybercrime techniques, and applying digital forensics methodologies. | CO1 |
| 2 | **Project Title: Simulation and Prevention of Cyber Crimes** **Problem Statement:** Develop a comprehensive simulation and prevention strategy for various types of cybercrimes. The project will involve creating scenarios for | CO2 |

| | | |
|---|---|---|
| | unauthorized access, phishing, and malware attacks, and implementing preventive measures. | |
| 3 | **Project Title: Investigation and Reporting of Cyber Crime Incidents**<br>**Problem Statement:** Investigate a simulated cybercrime incident, collect and analyze digital evidence, and report the findings. The project will cover the entire investigation process from securing the scene to data analysis and reporting. | CO3 |
| 4 | **Project Title: Advanced Digital Forensics and Evidence Processing**<br>**Problem Statement:** Develop a system for advanced digital forensics and processing of electronic evidence. The project will involve using forensic tools for data recovery, vulnerability assessment, and processing digital evidence from various devices. | CO4 |

## AI IN CYBER SECURITY

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| **AI in Cyber Security** | **ENSP305** | 4-0-0 | 4 | 40 |
| **Type of Course:** | DSE-1 | | | |

**Pre-requisite(s), if any:** Basic understanding of web development technologies such as HTML, CSS, and JavaScript. Additionally, students should have some familiarity with networking concepts, operating systems, and databases.

**Course Perspective.** The course delves into the integration of Artificial Intelligence (AI) techniques within the realm of cyber security, highlighting the transformative potential of AI in detecting, preventing, and responding to cyber threats. As cyber threats evolve in complexity and scale, AI offers advanced methodologies to enhance security measures and mitigate risks effectively. This course provides a comprehensive understanding of the applications of AI in cyber security, from fundamental machine learning and deep learning techniques to their practical implementations in threat detection and prevention. Students will explore the history, evolution, and current trends of AI in cyber security, gaining insights into the ethical considerations and challenges associated with the adoption of AI technologies in this critical field. Through detailed case studies and practical examples, the course bridges theoretical concepts with real-world applications, equipping students with the skills necessary to leverage AI for robust cyber defense strategies. The course is structured into four modules:

a) Introduction to AI and Cyber Security:

b) Machine Learning Techniques for Cyber Security:

c) Deep Learning Techniques for Cyber Security:

d) AI for Cyber Security: Threat Detection and Prevention

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the concepts and applications of AI in the field of cyber security. |

| | |
|---|---|
| **CO 2** | **Expressing** the ethical and legal considerations associated with the use of AI in cyber security. |
| **CO 3** | **Determining** emerging trends and technologies in AI for cyber security, and their potential impact on the field. |
| **CO 4** | **Identifying** strategies for integrating AI-driven solutions into existing cyber security frameworks, policies, and practices. |
| **CO 5** | **Articulating** critical thinking and problem-solving skills to address real-world cyber security challenges using AI techniques. |
| **CO 6** | **Designing** machine learning techniques for threat detection and prevention in cyber security, including supervised and unsupervised algorithms. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number: 1** | **Title: Introduction to AI and Cyber Security** | **No. of hours: 10** |

**Content:**

Overview of Artificial Intelligence and its applications in Cyber Security History and evolution of AI in cyber security, Understanding of the Cyber Security threats landscape, Familiarization with the latest trends and techniques of AI in Cyber Security, Basic principles of Machine Learning and Deep Learning in Cyber Security, Ethical considerations and challenges of using AI in cyber security.

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: Machine Learning Techniques for Cyber Security** | **No. of hours: 10** |

**Content:**

An introduction to Machine Learning techniques, Supervised and unsupervised Machine Learning models in Cyber Security, feature engineering and data preparation for Machine Learning models, Case studies demonstrating the application of Machine Learning to Cyber Security problems.

| | | |
|---|---|---|
| **Unit Number: 3** | **Title: Deep Learning Techniques for Cyber** | **No. of hours: 10** |

**Content:**

Introduction to Deep Learning techniques, Convolutional Neural Networks (CNNs) and their application in Cyber Security, Recurrent Neural Networks (RNNs) and their application in Cyber Security, GANs and their application in Cyber Security, Case studies demonstrating the application of Deep Learning to Cyber Security problems.

| **Unit Number: 4** | **Title: AI for Cyber Security: Threat Detection and Prevention** | **No. of hours: 10** |
|---|---|---|

**Content:**

Introduction to AI and its applications in threat detection and prevention ,Overview of different types of threats in cyber security and their characteristics ,Understanding the limitations of traditional threat detection and prevention methods ,Fundamentals of machine learning and deep learning for threat detection and prevention ,Supervised machine learning algorithms for threat detection, such as decision trees, support vector machines, and random forests ,Unsupervised machine learning algorithms for anomaly detection, such as clustering and outlier detection ,Deep learning techniques for threat detection, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) ,Feature selection and feature engineering for machine learning in threat detection, Emerging trends and challenges in AI for threat detection and prevention, including adversarial machine learning, explainable AI, and privacy concerns.

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures and Video Sessions**: Engage with presentations and videos on AI in cyber security.
2. **Problem-Based Theory Assignments**: Analyze real-world threats and AI solutions to enhance critical thinking.
3. **Project-Based Lab Assignments**: Implement AI algorithms in labs to detect and mitigate cyber threats.
4. **Collaborative Group Work**: Work in teams on case studies at the intersection of AI and cyber security.
5. **Continuous Assessment and Feedback**: Monitor progress through assessments and receive regular feedback.

**Outside Classroom Learning Experience**

1. **Use of ICT Tools and Moodle LMS**: Access course materials via Moodle LMS for flexible learning.
2. **Engagement with a Question Bank and Model Papers**: Utilize a question bank and model papers for exam prep.
3. **Application of Theoretical Knowledge to Practical Scenarios**: Apply AI concepts to real-world cyber security cases.

**Text Books:**

1. Artificial Intelligence for Cybersecurity" by Bhaskar Sinha (Auerbach Publications)
2. Machine Learning and Security: Protecting Systems with Data and Algorithms" by Clarence Chio and David Freeman (O'Reilly Media)

**Additional Readings:**

**Online Learning Resources:**

I) **Cybrary - Introduction to Artificial Intelligence for Cyber Security**
   a. This course offers insights into how AI can be applied to cyber security, including threat detection and response.
   b. Link: Cybrary - Introduction to Artificial Intelligence for Cyber Security

II) **Pluralsight - Machine Learning and AI for Cybersecurity**
   a. This course provides an in-depth look at how machine learning and AI can be used to enhance cyber security measures.
   b. Link: Pluralsight - Machine Learning and AI for Cybersecurity

III) **FutureLearn - Artificial Intelligence for Cyber Security by Coventry University**
   a. This course explores the application of AI in cyber security, covering topics like threat detection, response, and mitigation.
   b. Link: FutureLearn - Artificial Intelligence for Cyber Security

IV) **MIT OpenCourseWare - Artificial Intelligence**
   a. Lecture notes, assignments, and exams from MIT's course on Artificial Intelligence, providing a deep dive into AI concepts applicable to cyber security.
   b. Link: MIT OpenCourseWare - Artificial Intelligence

V) **IBM - Introduction to Cyber Security Tools & Cyber Attacks**
   a. A course that covers various cyber security tools and techniques, including the use of AI and machine learning for threat detection and prevention.
   b. Link: IBM - Introduction to Cyber Security Tools & Cyber Attacks

# AI IN CYBER SECURITY LAB

**Program Name:** **B. Tech (Computer Science and Engineering)**

| **Course Name:** **AI in Cyber Security Lab** | **Course Code** | **L-T-P** | **Credits** |
|---|---|---|---|
| | **ENSP355** | 0-0-2 | 1 |

**Type of Course:** DSE-2

**Pre-requisite(s), if any:** Basic understanding of web development technologies such as HTML, CSS, and JavaScript. Additionally, students should have some familiarity with networking concepts, operating systems, and databases.

**Defined Course Outcomes**

| COs | Statement |
|---|---|
| CO 1 | **Analyzing** the history, evolution, and ethical considerations of AI in cyber security, documenting key milestones and advancements, and discussing the implications of AI applications. |
| CO 2 | **Implementing** and evaluate machine learning models for classifying and detecting cyber threats, using various datasets and techniques such as supervised and unsupervised learning, deep learning, and anomaly detection. |
| CO 3 | **Developing** and apply feature engineering, data preparation, and model training techniques to enhance the performance and accuracy of cyber security models. |
| CO 4 | **Conducting** comprehensive case studies and surveys on the application of AI in cyber security, identifying emerging trends, challenges, and documenting methodologies and findings. |

## LAB EXPERIMENTS

| **Ex. No** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| P1 | **Project Title: Comprehensive Analysis of AI in Cyber Security** **Problem Statement:** Conduct a comprehensive analysis of the role of AI in cyber security. The project will involve studying the history, evolution, and current | CO1 |

|  |  |  |
|---|---|---|
|  | trends in AI applications for cyber security, and understanding the basic principles of machine learning and deep learning in this context. |  |
| P2 | **Project Title: Machine Learning Models for Cyber Threat Detection** **Problem Statement:** Develop and evaluate different machine learning models for detecting cyber threats. The project will involve implementing supervised and unsupervised learning techniques, performing feature engineering, and analyzing case studies. | CO2 |
| P3 | **Project Title: Deep Learning Models for Advanced Cyber Threat Detection** **Problem Statement:** Develop and evaluate deep learning models for advanced cyber threat detection. The project will involve implementing CNNs, RNNs, and GANs, and analyzing their applications in cyber security. | CO3 |
| P4 | **Project Title: AI-Based Comprehensive Threat Detection System** **Problem Statement:** Develop a comprehensive AI-based system for threat detection and prevention in cyber security. The project will involve implementing machine learning and deep learning models and addressing the challenges of traditional threat detection methods. | CO4 |

## SOCIAL MEDIA SECURITY

**Program Name:** **B. Tech (Computer Science and Engineering)**

| **Course Name:** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
|---|---|---|---|---|
| **Social Media Security** | **ENSP307** | 4-0-0 | 4 | 40 |

**Type of Course:** DSE-1

**Pre-requisite(s), if any: Fundamentals of Cyber and Digital Media**

**Course Perspective.** This course introduces students to the critical concepts of social media security, addressing the growing need to understand and manage security and privacy issues in the digital age. Social media platforms have become integral to personal, professional, and commercial interactions, creating a complex landscape of potential security threats and privacy concerns. This course aims to equip students with the knowledge and skills required to navigate and mitigate these risks effectively. Students will explore the technical, legal, and social dimensions of social media security, developing strategies to safeguard personal information, ensure user trust, and comply with legal standards. The course is divided into four modules:

a) Social Media Overview

b) Security Issues in Social Media

c) Privacy Issues in Social Media

d) Social Media Security: Laws, Best Practices, and Case Studies

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Demonstrating** an understanding of the different types of social media platforms, their features, and their impact on communication, marketing, and society. |
| **CO 2** | **Acquiring** knowledge and skills in social media monitoring techniques, including data collection, analysis, and the use of relevant tools and technologies. |
| **CO 3** | **Developing** the ability to analyze and evaluate viral content on social media, understand the factors contributing to its spread, and recognize its implications for marketing and online engagement. |
| **CO 4** | **Identifying** the challenges, opportunities, and pitfalls associated with social media marketing, and formulate strategies for effective audience targeting, engagement, and brand promotion. |
| **CO 5** | **Designing** strategies to safeguard personal information, foster user trust, and mitigate associated risks. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

**Unit Number: 1 Title: Social Media Overview No. of hours: 10**

**Content Summary:**

Introduction to Social media. Types of social media, Social media platforms, Social media monitoring, Hashtag, Viral content, Social media marketing, challenges, opportunities, and pitfalls in online social networks, APIs, Collecting data from Online Social Media, Social Media Content Analysis - BoW Model, TF-IDF; Network Analysis - Node Centrality Measures, Degree Distribution, Average Path Length, Clustering Coefficient, Power Law; Synthetic Networks - Random Graphs, Preferential Attachment Model.

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: S**ocial Media Management and Marketing | **No. of hours: 10** |

**Content Summary:**

Strategies for using social media for employment screening and recruitment, Customer engagement and content management, Analysis of effective versus ineffective social media campaigns, Ethical considerations and privacy issues in crowdsourcing, Managing and promoting social media presence

| | | |
|---|---|---|
| **Unit Number: 3** | **Title: Privacy Issues in Social Media** | **No. of hours: 10** |

**Content Summary:**

Overview, Privacy Settings, PII Leakage, Identity vs Attribute Disclosure Attacks, Inference Attacks, De-anonymization Attacks, Privacy Metrics - k-anonymity, l-diversity, Personalization vs Privacy, Differential Privacy, Social Media and User Trust.

| | | |
|---|---|---|
| **Unit Number: 4** | **Title: Social Media Security: Laws, Best Practices, and Case Studies** | **No. of hours: 10** |

**Content Summary:**

Laws regarding posting of inappropriate content, Best practices for the use of Social media, Content Moderation and Removal Policies, User Authentication and Access Control, Security Awareness and Education, Social media Case studies-Facebook, Twitter, Instagram, YouTube, LinkedIn, StackOverflow, GitHub, Quora, SnapChat, Reddit, FourSquare, Yelp.

**Learning Experiences**

**Classroom Learning Experience**

1. **Interactive Lectures and Video Sessions**: Learn about social media security threats through engaging presentations.
2. **Problem-Based Theory Assignments**: Analyze real incidents to enhance critical thinking on social media security.
3. **Project-Based Lab Assignments**: Implement security measures in labs to protect social media accounts.
4. **Collaborative Group Work**: Explore security risks through team case studies on social media platforms.
5. **Continuous Assessment and Feedback**: Receive ongoing assessments and instructor feedback on progress.

**Outside Classroom Learning Experience**

1. **Use of ICT Tools and Moodle LMS**: Access course materials anytime via Moodle LMS.
2. **Engagement with a Question Bank and Model Papers**: Prepare for exams using a question bank and model papers.
3. **Application of Theoretical Knowledge to Practical Scenarios**: Apply concepts to real-world social media security cases.

**References**

1. Mastering Social Media Mining, Bonzanini Marco, Packt Publishing Limited
2. Mining the Social Web, Mikhail Klassen and Matthew A. Russell, O'Reilly Media, Inc
3. Social media mining: an introduction, Zafarani, Reza, Mohammad Ali Abbasi, and Huan Liu, Cambridge University Press
4. Social Media Security: Leveraging Social Networking While Mitigating Risk, Michael Cross, Syngress
5. Social Media and the Law: A Guidebook for Communication Students and Professionals, Daxton R. Stewart, Taylor & Francis Ltd
6. Security in the Digital Age: Social Media Security Threats and Vulnerabilities by Henry A. Oliver, Create Space Independent Publishing Platform.

**Additional Readings:**

**Online Learning Resources for Social Media Security**

**R 1. Coursera - Social Media Marketing Specialization**

- **Provider:** Northwestern University
- **Description:** This specialization covers the major social media platforms, marketing strategies, and data analysis tools.
- **Link:** Coursera Social Media Marketing Specialization

**R 2. edX - Cybersecurity Fundamentals**

- **Provider:** Rochester Institute of Technology
- **Description:** This course offers foundational knowledge in cybersecurity, including threats, vulnerabilities, and defense strategies.
- **Link:** edX Cybersecurity Fundamentals

**R 3. Udemy - The Complete Cyber Security Course: Network Security!**

- **Instructor:** Nathan House
- **Description:** This comprehensive course covers network security, including how to secure your network, protect your devices, and more.

- **Link:** Udemy Cyber Security Course

# SOCIAL MEDIA SECURITY LAB

**Program Name:** **B. Tech (Computer Science and Engineering)**

| **Course Name: Social Media Security Lab** | **Course Code** | **L-T-P** | **Credits** |
|---|---|---|---|
| | **ENSP357** | 0-0-2 | 1 |

**Type of Course:** DSE-2

**Pre-requisite(s), if any: Fundamentals of Cyber and Digital Media**

**Course Outcomes (CO)**

| COs | Statements |
|---|---|
| CO1 | **Analyzing** different social media platforms, their features, and the ethical and privacy considerations in crowdsourcing and data handling. |
| CO2 | **Implementing** data collection, text analysis, and network analysis techniques on social media datasets, demonstrating proficiency in using APIs and various models. |
| CO3 | **Developing** strategies and plans for using social media in various contexts such as employment screening, customer engagement, and small business promotion. |
| CO4 | **Evaluating** privacy settings, metrics, and security incidents on social media platforms, applying best practices for user authentication, access control, and content moderation. |

**LAB EXPERIMENTS**

| **Ex. No** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| P1 | **Project Title: Comprehensive Analysis of Social Media Platforms**<br>**Problem Statement:** Conduct a comprehensive analysis of different social media platforms, their features, and the data they generate. The project will involve collecting and analyzing data from social media, performing content analysis, and understanding network properties. | CO1 |
| P2 | **Project Title: Effective Social Media Management and Marketing Strategy**<br>**Problem Statement:** Develop an effective social media management and marketing strategy for a small business. The project will involve | CO2 |

| | | |
|---|---|---|
| | analyzing customer engagement, creating marketing strategies, and addressing ethical considerations in social media use. | |
| P3 | **Project Title: Privacy Protection in Social Media**<br>**Problem Statement:** Develop strategies and tools to protect user privacy on social media platforms. The project will involve analyzing privacy settings, simulating privacy attacks, and evaluating privacy metrics. | CO3 |
| P4 | **Project Title: Enhancing Security and Compliance on Social Media Platforms**<br>**Problem Statement:** Develop a comprehensive approach to enhance security and ensure compliance with laws on social media platforms. The project will involve researching laws, developing best practices, and analyzing case studies of security incidents. | CO4 |

# Summer Internship-II

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name: Summer Internship-II** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSI351** | 0-0-0 | 2 |
| **Type of Course:** | INT-2 | | |
| **Pre-requisite(s), if any: NA** | | | |

**Duration:**

The internship will last for six weeks. It will take place after the completion of the 4th semester and before the commencement of the 5th semester.

**Internship Options:**

Students can choose from the following options:

- **Industry Internship (Offline) or Internship in Renowned Institutions (Offline):**
    - Students must produce a joining letter at the start and a relieving letter upon completion.

**Report Submission and Evaluation:**

1. **Report Preparation:**
    - Students must prepare a detailed report documenting their internship experience and submit it to the department. A copy of the report will be kept for departmental records.
2. **Case Study/Project/Research Paper:**
    - Each student must complete one of the following as part of their internship outcome:
        1. A case study
        2. A project
        3. A research paper suitable for publication
3. **Presentation:**
    - Students are required to present their learning outcomes and results from their summer internship as part of the evaluation process.

**Evaluation Criteria for Summer Internship (Out of 100 Marks):**

1. **Relevance to Learning Outcomes (30 Marks)**
    - **Case Study/Project/Research Paper Relevance (15 Marks):**
        1. Directly relates to core subjects: 15 marks
        2. Partially relates to core subjects: 10 marks

        3. Minimally relates to core subjects: 5 marks
        4. Not relevant: 0 marks
    - **Application of Theoretical Knowledge (15 Marks):**
        1. Extensive application of theoretical knowledge: 15 marks
        2. Moderate application of theoretical knowledge: 10 marks
        3. Minimal application of theoretical knowledge: 5 marks
        4. No application of theoretical knowledge: 0 marks

2. **Skill Acquisition (40 Marks)**
    - **New Technical Skills Acquired (20 Marks):**
        1. Highly relevant and advanced technical skills: 20 marks
        2. Moderately relevant technical skills: 15 marks
        3. Basic technical skills: 10 marks
        4. No new skills acquired: 0 marks
    - **Professional and Soft Skills Development (20 Marks):**
        1. Significant improvement in professional and soft skills: 20 marks
        2. Moderate improvement in professional and soft skills: 15 marks
        3. Basic improvement in professional and soft skills: 10 marks
        4. No improvement: 0 marks
3. **Report Quality (15 Marks)**
    - **Structure and Organization (8 Marks):**
        1. Well-structured and organized report: 8 marks
        2. Moderately structured report: 6 marks
        3. Poorly structured report: 3 marks
        4. No structure: 0 marks
    - **Clarity and Comprehensiveness (7 Marks):**
        1. Clear and comprehensive report: 7 marks
        2. Moderately clear and comprehensive report: 5 marks
        3. Vague and incomplete report: 2 marks
        4. Incomprehensible report: 0 marks
4. **Presentation (15 Marks)**
    - **Content Delivery (8 Marks):**
        1. Clear, engaging, and thorough delivery: 8 marks
        2. Clear but less engaging delivery: 6 marks
        3. Somewhat clear and engaging delivery: 3 marks

        4. Unclear and disengaging delivery: 0 marks

  - **Visual Aids and Communication Skills (7 Marks):**
    1. Effective use of visual aids and excellent communication skills: 7 marks
    2. Moderate use of visual aids and good communication skills: 5 marks
    3. Basic use of visual aids and fair communication skills: 2 marks
    4. No use of visual aids and poor communication skills: 0 marks

**Total: 100 Marks**

**Course Outcomes:**

By the end of this course, students will be able to:

1. **Apply Theoretical Knowledge:**
   - Integrate and apply theoretical knowledge gained during coursework to real-world industry or research problems.
2. **Develop Technical Skills:**
   - Acquire and demonstrate advanced technical skills relevant to the field of computer science and engineering through practical experience.
3. **Conduct Independent Research:**
   - Execute independent research projects, including problem identification, literature review, methodology design, data collection, and analysis.
4. **Prepare Professional Reports:**
   - Compile comprehensive and well-structured reports that document the internship experience, project details, research findings, and conclusions.
5. **Enhance Problem-Solving Abilities:**
   - Develop enhanced problem-solving and critical thinking skills by tackling practical challenges encountered during the internship.
6. **Improve Professional and Soft Skills:**
   - Exhibit improved professional and soft skills, including communication, teamwork, time management, and adaptability in a professional setting.
7. **Present Findings Effectively:**
   - Deliver clear and engaging presentations to effectively communicate project outcomes, research findings, and acquired knowledge to peers and faculty members.

**Learning Experiences**

**Classroom Learning Experience**

1. **Orientation Sessions**: Introduce internship objectives, expectations, and assessment criteria.

2. **Skill Development Workshops**: Cover essential professional skills such as communication, teamwork, and time management.
3. **Project Planning Guidance**: Assist students in developing project proposals aligned with internship goals.
4. **Progress Presentations**: Facilitate sessions for students to present updates and receive constructive feedback.
5. **Group Discussions**: Encourage sharing of challenges and solutions experienced during internships.
6. **Continuous Feedback**: Implement regular check-ins and mentor evaluations to assess student progress.

**Outside Classroom Learning Experience**

1. **Internship Placement**: Engage students in real-world work environments to apply learned skills.
2. **Reflective Journals**: Encourage students to document experiences, challenges, and lessons learned throughout the internship.
3. **Project Implementation**: Work on assigned projects and tasks within the organization, applying theoretical knowledge.
4. **Networking Opportunities**: Create opportunities for students to connect with industry professionals and peers.
5. **Self-Assessment**: Provide tools for students to evaluate their performance and contributions during the internship.

- **Final Presentations**: Organize sessions for students to showcase their internship experiences and project outcomes to faculty and peers.

# SOFTWARE ENGINEERING

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Software Engineering** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCS305** | 4-0-0 | 4 | 40 |
| **Type of Course:** | Major-26 | | | |
| **Pre-requisite(s), if any: Fundamentals of Computers** | | | | |

**Course Perspective.** The course offers a comprehensive overview of the fundamental principles, methodologies, and tools used in the development and maintenance of software systems. This course emphasizes the importance of a disciplined approach to software engineering, covering various lifecycle models, requirements engineering, software design, metrics, and testing methodologies. Students will learn how to apply these principles to develop high-quality, maintainable software systems.

**The Course Outcomes (COs).** On completion of the course the participants will be able to :

| **COs** | **Statements** |
|---|---|
| **CO 1** | **Understand** software engineering as an iterative and systematic process and the software development lifecycle models. |
| **CO 2** | **Design** the software development process to complement technical understanding of software products using UML diagram |
| **CO 3** | Estimate the cost/effort of the software development processes using different metrics |
| **CO 4** | **Generate** test case specifications and test cases from given requirements. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| **UNIT WISE DETAILS** | | |
|---|---|---|
| **Unit Number: 1** | **Title: Introduction to Software Engineering** | **No. of hours: 10** |

| | | |
|---|---|---|
| **Content Summary:**<br>Introduction to Software Engineering, Software Evolution, Software Characteristics, Software Crisis: Problem and Causes, Software process models (Waterfall, Incremental, and Evolutionary process models and Agile), Software quality concepts, process improvement, software process capability maturity models, Personal Software process and Team Software Process, Overview of Agile Process. | | |
| **Unit Number: 2** | **Title: Software Requirement analysis, Design & Construction** | **No. of hours: 10** |
| **Content Summary:**<br>Problem Analysis, Requirement elicitation and Validation, Requirements modeling: Scenarios, Information and analysis classes, flow and behavioral modeling, documenting Software Requirement Specification (SRS). System design principles: levels of abstraction, separation of concerns, information hiding, coupling and cohesion, Structured design, object-oriented design, event driven design, component-level design, test driven design, aspect oriented design, Design patterns, Coding Practices: Techniques, Refactoring | | |
| **Unit Number: 3** | **Title: Software Project Management and UML** | **No. of hours: 10** |
| **Content Summary:**<br>**Software Project Management:** SP Estimation of scope (LOC, FP etc), time (PERT/CPM Networks), and cost (COCOMO models), Quality Management, Plan for software Quality Control and Assurance, Earned Value Analysis.<br>**UML:** UML Structural Diagrams, UML Behavioural Diagrams. | | |
| **Unit Number: 4** | **Title: Software Testing & Maintenance:** | **No. of hours: 10** |
| **Content Summary:**<br>**Testing:** Levels of Testing, Functional Testing, Structural Testing, Test Plan, Test Case Specification, Software Testing Strategies, Verification & Validation, Unit, Integration Testing, Top Down and Bottom-Up Integration Testing, Alpha & Beta Testing, White box and black box testing techniques, System Testing and Debugging.<br><br>**Software Maintenance:** Maintenance Process, Maintenance Models, Reverse Engineering, Software RE-engineering. | | |

**Learning Experiences:**

**Inside Classroom Learning Experience:**

1. **Interactive Lectures with PPTs:** Students will engage in dynamic classroom sessions using Lecture PPTs and interactive teaching boards, fostering a deeper understanding of software engineering concepts.

2. **Problem-Based Theory Assignments:** Regular assignments will focus on real-world software problems, enhancing analytical thinking and critical problem-solving skills.
3. **Project-Based Lab Assignments:** Hands-on projects will encourage students to apply theoretical concepts in practical scenarios, reinforcing their learning through experience.
4. **Continuous Assessment & Feedback:** Ongoing assessments will track progress, and timely feedback will be provided to support student growth. Additional support will be available for those seeking extra guidance.

**Outside Classroom Learning Experience:**

1. Problem-Based Theory Assignments: Students will work on regular assignments outside the classroom, solving real-world software problems. These assignments will enhance critical problem-solving abilities and deepen their understanding of concepts.
2. **Project-Based Lab Assignments:** Outside the classroom, students will continue to work on project-based lab assignments, applying theoretical knowledge to practical projects and software development tasks.
3. **Comprehensive Question Bank:** Students will have access to a detailed question bank covering the entire syllabus. They can use this resource to practice and revise outside class, preparing thoroughly for assessments.
4. **Model Question Papers for Exam Practice:** Students can work on model question papers at home to become familiar with exam patterns and improve their confidence, time management, and preparation for final assessments.
5. **Use of ICT Tools:** Students will have access to Moodle LMS outside class, where all learning materials, assignments, video lectures, and resources will be available. This ensures continuous learning and easy access to materials anytime, anywhere.
6. **Continuous Assessment & Additional Support:** Students can seek additional support and feedback outside the classroom through one-on-one sessions, emails, or discussion forums on Moodle LMS. This helps ensure continuous learning and provides extra guidance for those in need.

**Text books:**

T1: R. S. Pressman, “Software Engineering – A practitioner’s approach”, 7th Ed., McGraw Hill Int. Ed., 2010.

T2: K. K. Aggarwal & Yogesh Singh, “Software Engineering”, 4th Ed, New Age International, 2022.

T3: Rajib Mall, Fundamentals of Software Engineering, PHI Learning, 5th Edition, 2018.

**Additional Readings:**

**R 1. MIT OpenCourseWare - Software Engineering for Web Applications**

- **Link:** MIT OCW Software Engineering for Web Applications
- **Description:** This course explores software engineering principles with a focus on web applications, including requirements analysis, design, testing, and maintenance.

**R 2. edX - Software Development Fundamentals**

- **Link:** edX Software Development Fundamentals
- **Description:** This professional certificate program covers fundamental software development concepts, including requirements engineering, software design, and testing.

# COMPETITIVE CODING-III

|  |  |  |  |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** **COMPETITIVE CODING -III** | **Course Code** | **L-T-P** 2-0-0 | **Credits** 0 |
| **Type of Course:** | AUDIT-3 | | |
| **Contact Hours** | 30 | | |

**Course Outcomes**

|  |  |
|---|---|
| **CO1** | **Analyzing** and write SQL queries to retrieve, modify, and optimize data in relational databases. |
| **CO2** | **Designing** and **implementing** efficient tree-based data structures like AVL, B Trees, and Splay Trees to solve computational problems. |
| **CO3** | **Developing** solutions for optimization problems using greedy algorithms and dynamic programming approaches. |
| **CO4** | **Implementing** and evaluate graph algorithms for traversing, searching, and finding shortest paths in complex graph structures. |

| **Unit Number: 1** | **Title: SQL & PL/SQL** | **No. of hours: 8** |
|---|---|---|
| **Content:**<br>**Introduction to Databases and SQL**:<br>o Understand relational databases, tables, and SQL queries.<br>o Practice SELECT, INSERT, UPDATE, DELETE statements.<br>**Joins and Subqueries**:<br>o Master INNER JOIN, LEFT JOIN, RIGHT JOIN, and self-joins.<br>o Learn about subqueries and correlated subqueries.<br>**Indexes and Query Optimization**:<br>o Explore indexing techniques (B-tree, hash indexes).<br>o Optimize SQL queries for performance. | | |

|  |  |  |
|---|---|---|
| **PL/SQL Basics**:<br>o Introduce PL/SQL (Procedural Language/Structured Query Language).<br>o Write basic PL/SQL blocks, loops, and conditional statements | | |
| **Unit Number: 2** | **Title:** Height Balanced Tree Concepts | **No. of hours: 8** |
| **Content:**<br>**AVL Trees:** Definition and Properties, Rotations , AVL Tree Operations (Insertion, Deletion, Lookup), Complexity Analysis<br><br>**B Trees :** Definition and Properties, B Tree Operations, Complexity Analysis, Applications in Databases and File Systems<br>**B+ Trees:** Definition and Properties, B+ Tree Operations, Complexity Analysis<br><br>**Splay Trees:** Definition and Properties, Splaying Operation, Splay Tree Operations (Insertion, Deletion, Lookup), Complexity Analysis<br><br>**Applications of Height Balanced Trees:** Use in Databases (Indexing), Use in Memory Management (Allocators) | | |
| **Unit Number: 3** | **Title: Greedy Design Strategy and Dynamic Programming** | **No. of hours: 8** |
| **Content:**<br>**Greedy Algorithms:** Definition and Characteristics, Greedy Choice Property, Optimal Substructure<br>**Dynamic Programming:** Definition and Characteristics, Optimal Substructure, Overlapping Subproblems, Comparison with Greedy Algorithms<br>**Greedy Algorithms**<br>**Basic Greedy Algorithms:** Activity Selection Problem, Huffman Coding, Kruskal’s Algorithm, Prim’s Algorithm, Fractional Knapsack Problem<br>Complexity Analysis: Time Complexity, Proof of Optimality<br><br>**Dynamic Programming** | | |

| | | |
|---|---|---|
| **Basic Dynamic Programming Problems:** Fibonacci Sequence (Memoization vs. Tabulation), 0/1 Knapsack Problem, Longest Common Subsequence (LCS), Matrix Chain Multiplication | | |
| **Unit Number: 4** | **Title: Graph Algorithms** | **No. of hours: 6** |
| **Content:**<br>**Graph Representations:**<br>▪ Representing graphs using adjacency matrix and adjacency list.<br>▪ Solving basic graph traversal problems.<br>**Breadth-First Search (BFS):**<br>▪ Implementing BFS for finding the shortest path in unweighted graphs.<br>▪ Applications include finding connected components.<br>**Depth-First Search (DFS):**<br>▪ Implementing DFS for tasks like topological sorting and cycle detection.<br>**Shortest Path Algorithms**<br>**Dijkstra's Algorithm:**<br>▪ Implementing Dijkstra's algorithm for finding shortest paths in weighted graphs.<br>▪ Using priority queues for efficient computation.<br>**Bellman-Ford Algorithm:**<br>▪ Handling negative weights with the Bellman-Ford algorithm.<br>▪ Detecting negative weight cycles. | | |

## Lab Experiments

| **Problem Statement** | **Mapped COs** |
|---|---|
| **SQL & PL/SQL** | |
| 1. Write an SQL query to find the department with the highest average salary. | CO1 |
| 2. Write a query to find all employees who earn more than their managers. | CO1 |
| 3. Retrieve the top three salaries from the "employees" table. | CO1 |

| **Problem Statement** | **Mapped COs** |
|---|---|
| 4. Write an SQL query to find employees who have been in the company for more than 5 years. | CO1 |
| 5. Write a query to delete duplicate rows from a table without using temporary tables. | CO1 |
| 6. Fetch all records where the customer ordered more than once from the "orders" table. | CO1 |
| 7. Find the name of departments with more than 10 employees using a JOIN between "employees" and "departments". | CO2 |
| 8. Write a query to retrieve all customers who ordered more than the average number of orders using subqueries. | CO2 |
| 9. Write a query to find the second highest salary of employees using a subquery. | CO2 |
| 10. Optimize the performance of a query that fetches all orders placed in the last 30 days from the "orders" table using indexing. | CO3 |
| 11. Use indexing to speed up searches on the "products" table and compare the execution time before and after indexing. | CO3 |
| 12. Write a PL/SQL block to display the Fibonacci sequence up to a given number using loops. | CO4 |
| 13. Create a PL/SQL block that calculates the factorial of a number using recursion. | CO4 |
| **Height Balanced Tree Concepts** | |
| 14. Implement an AVL Tree and insert a series of elements into it. Ensure the tree remains balanced after each insertion. | CO5 |
| 15. Write a function to check if a given AVL Tree is height-balanced. | CO5 |
| 16. Perform AVL Tree deletion and ensure rebalancing using rotations. | CO5 |
| 17. Implement an AVL Tree lookup operation and calculate its time complexity. | CO5 |
| 18. Implement insertion operations in a B Tree and verify the tree’s structure after each insertion. | CO6 |

| **Problem Statement** | **Mapped COs** |
|---|---|
| 19. Write a function to search for an element in a B Tree and trace the steps of the search. | CO6 |
| 20. Implement deletion operations in a B Tree and verify rebalancing after each deletion. | CO6 |
| 21. Perform insertion and deletion operations in a B+ Tree and trace the changes in the tree structure. | CO6 |
| 22. Demonstrate the application of B Trees in database indexing with a small dataset. | CO6 |
| 23. Implement a splay tree and observe the behavior of nodes being splayed to the root after lookups. | CO6 |
| 24. Compare the performance of AVL Trees, B Trees, and Splay Trees for a series of random insertions. | CO6 |
| **Greedy Design Strategy and Dynamic Programming** | |
| 25. Solve the Activity Selection Problem using a greedy algorithm. | CO7 |
| 26. Write a function to implement Huffman coding for a string of characters and display the encoded output. | CO7 |
| 27. Implement Kruskal's algorithm for finding the minimum spanning tree of a graph. | CO7 |
| 28. Solve the Fractional Knapsack Problem using a greedy approach. | CO7 |
| 29. Solve the 0/1 Knapsack Problem using dynamic programming. | CO8 |
| 30. Write a program to find the nth Fibonacci number using memoization and compare it with the iterative approach. | CO8 |
| 31. Implement dynamic programming to solve the Longest Common Subsequence (LCS) problem. | CO8 |
| 32. Solve the Matrix Chain Multiplication problem using dynamic programming and analyze the time complexity. | CO8 |
| **Graph Algorithms** | |

| **Problem Statement** | **Mapped COs** |
|---|---|
| 33. Implement a graph using an adjacency list and perform Depth-First Search (DFS) to detect cycles. | CO9 |
| 34. Implement Breadth-First Search (BFS) to find all connected components in an unweighted graph. | CO9 |
| 35. Solve a shortest-path problem in an unweighted graph using BFS. | CO9 |
| 36. Write a program to implement Dijkstra's algorithm to find the shortest path in a weighted graph. | CO10 |
| 37. Implement Bellman-Ford algorithm to find shortest paths in a graph with negative weights. | CO10 |
| 38. Detect negative weight cycles in a graph using the Bellman-Ford algorithm. | CO10 |
| 39. Perform topological sorting of a directed graph using DFS. | CO9 |
| 40. Solve a shortest-path problem using Dijkstra's algorithm with priority queues and analyze the time complexity. | CO10 |

## Arithmetic & Reasoning Skills-I

|  |  |  |  |  |
|---|---|---|---|---|
| **Programme Name:** | **B. Tech (Computer Science and Engineering)** |  |  |  |
| **Course Name: Arithmetic & Reasoning Skills-I** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
|  | **AEC008** | **3-0-0** | **3** | **24** |
| **Type of Course:** | **AEC-3** |  |  |  |
| **Pre-requisite(s), if any:** |  |  |  |  |

**Course Perspective:** The course aims to improve basic arithmetic skills, speed, and accuracy in mental calculations, and logical reasoning. These abilities are essential for a strong math foundation, helping students succeed in academics and various practical fields.

**The Course Outcomes (COs):** On completion of the course the participants will be able to:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** arithmetic algorithms required for solving mathematical problems. |
| **CO 2** | **Applying** arithmetic algorithms to improve proficiency in calculations. |
| **CO 3** | **Analyzing** cases, scenarios, contexts and variables, and understanding their inter-connections in a given problem. |
| **CO 4** | **Evaluating** & deciding approaches and algorithms to solve mathematical & reasoning problems. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

**Unit Number: 1** **Title:** Mathematical Essentials **No. of hours: 15**

**Content:** Table chart, Line graph, Bar graph, Pie chart

**Unit Number: 2** **Title:** Fundamentals of Logical Reasoning **No. of hours: 6**

**Content:** Blood Relations, Direction Sense, Coding Decoding

**Unit Number: 3** **Title:** Elementary Quantitative Skills **No. of hours: 18**

**Content:** Simple and Compound Interest, Average, Partnership, Time and Work, Time Speed & Distance

**Unit Number: 4** **Title:** Advanced Quantitative Skills **No. of hours: 6**

**Content:**

Permutation & Combination, Probability

**Learning Experiences**

**Classroom Learning Experience**

1. **Interactive Lectures**: Introduce advanced life skills concepts using PPTs and real-life scenarios.
2. **Conceptual Understanding**: Cover topics like strategic thinking, resilience, and ethical decision-making.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on real-world workplace dilemmas and solutions.
4. **Group Discussions**: Facilitate discussions on leadership styles, team dynamics, and conflict management.
5. **Guest Speakers**: Invite industry experts to share insights on personal and professional development.
6. **Continuous Feedback**: Implement quizzes and peer reviews to assess application of life skills in professional contexts.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign reflective essays on personal experiences and career aspirations.
2. **Workshops**: Facilitate hands-on sessions for practicing advanced communication and negotiation skills.
3. **Question Bank**: Provide resources for self-assessment on life skills development and application.
4. **Online Forums**: Create platforms for discussing personal growth challenges and sharing experiences.
5. **Self-Study for Case Studies**: Encourage independent research on successful professionals and their life skills.

6. **Collaborative Projects**: Organize group projects focused on community engagement and leadership initiatives.

**Evaluation Scheme (Theory):**

| **Evaluation Components** | **Weightage** |
|---|---|
| **Internal Marks (Theory)** | |
| **I) Continuous Assessment (30 Marks)** | |
| **(All the components to be evenly spaced)** | |
| Project/ Quizzes/ Assignments and Essays/ Presentations/ Participation/ Case Studies/ Reflective Journals (minimum of five components to be evaluated) | **30 Marks** |
| **II) Internal Marks (Theory) – Mid Term Exam** | **20 Marks** |
| **External Marks (Theory): -** | **50 Marks** |
| End term Examination | |
| **Total** | **100 Marks** |

**Note:** (It is compulsory for a student to secure 40% marks in Internal and End Term Examination separately to secure minimum passing grade).

**References**

**R 3.** Aggarwal, R. S. (2014). Quantitative aptitude (Revised edition).

**R 4.** Gladwell, M. (2021). Talking to strangers.

**R 5.** Scott, S. (2004). Fierce conversations.

## Semester: VI

# COMPUTER ORGANIZATION & ARCHITECTURE

**Program Name:** **B. Tech (Computer Science and Engineering)**

| **Course Name:** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
|---|---|---|---|---|
| **Computer Organization & Architecture** | **ENCS302** | 3-1-0 | 4 | 40 |

**Type of Course:** Major

**Pre-requisite(s), if any: Concepts of Digital Electronics**

**Course Perspective:** The course provides an in-depth understanding of the structural and operational principles of computer systems. It explores the intricate details of computer architecture, bridging the gap between high-level programming and hardware design. Students will learn about the functional units of a computer, instruction set architectures, processor design, memory hierarchy, and I/O systems. This knowledge is crucial for designing efficient, high-performance computing systems and understanding the trade-offs in hardware and software design. The course is divided into 4 units:

1. Introduction
2. Instruction Set Architecture (RISC-V)
3. The Processor
4. Memory hierarchy, Storage, and I/O

**Course Outcomes (COs):** On completion of the course the participants will be able to:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the basics of instructions sets and their impact on processor design. |
| **CO 2** | **Demonstrating** an understanding of the design of the functional units of a digital computer system |

| | |
|---|---|
| **CO 3** | **Evaluating** cost performance and design trade-offs in designing and constructing a computer processor including memory. |
| **CO 4** | **Designing** a pipeline for consistent execution of instructions with minimum hazards |
| **CO 5** | **Manipulating** representations of numbers stored in digital computers using I/O devices and store them into memory |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number: 1** | **Title: Introduction** | **No. of hours: 10** |

**Content:**

**Introduction to Computer Architecture:** Definitions and Concepts, Levels of abstraction, Von Neumann Architecture.

**Functional Blocks of a Computer:** CPU, memory, input-output subsystems, control unit.

**Instruction Set Architecture (ISA) of CPU:** Registers, instruction execution cycle, RTL (Register Transfer Language) interpretation of instructions, addressing modes, instruction set.

**Types of Instruction Set Architectures:** Reduced Instruction Set Computer (RISC) and Complex Instruction Set Computer (CISC).

**Data Representation:** Number Systems (binary, octal, decimal, hexadecimal), Arithmetic Operations (addition, subtraction, multiplication, division), Floating Point Representation (IEEE 754 standard).

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: Memory hierarchy, Storage and I/O** | **No. of hours: 10** |

**Content:**

**Memory Hierarchy:** Types of memory: RAM (Random Access Memory), ROM (Read-Only Memory), Cache, and Secondary Storage, SRAM (Static RAM) vs. DRAM (Dynamic RAM), Locality of reference: spatial and temporal locality.

**Caching:** Different indexing mechanisms: direct-mapped, set-associative, fully associative, Trade-offs related to block size, associativity, and cache size, Processor-cache interactions for read/write requests, Basic optimizations: write-through vs. write-back caches, Average memory access time (AMAT), Cache replacement policies: Least Recently Used (LRU), First-In-First-Out (FIFO).

**Storage:** Introduction to magnetic disks: notion of tracks, sectors, Flash memory: NAND and NOR flash, I/O Data Transfer Techniques: programmed I/O, interrupt-driven I/O, Direct Memory Access (DMA).

| | | |
|---|---|---|
| **Unit Number: 3** | **Title: The Processor** | **No. of hours: 10** |

**Content:**

**Building a Datapath:** Introduction, Logic Design Conventions, Building a Datapath: A Simple Implementation scheme, Overview of Pipelining: Pipelined Datapath and Control, Data Hazards: Forwarding versus Stalling, Control Hazards and their mitigation, Handling Exceptions, Parallelism via Instructions.

**Clocking Methodology:** Revisiting clocking methodology, Amdahl's Law and its implications.

**Processor Design:** Single cycle processor design, Multi-cycle processor design, Instruction pipelining: stages and performance considerations, Notion of Instruction Level Parallelism (ILP), Data and control hazards: identification and mitigation strategies.

| | | |
|---|---|---|
| **Unit Number: 4** | **Input/Output Systems and Advanced Topics** | **No. of hours: 10** |

**Content:**

**I/O Systems: I**/O Mapped vs. Memory-Mapped I/O, I/O Data Transfer Techniques: Programmed I/O, Interrupt-Driven I/O, Direct Memory Access (DMA)

**Advanced Memory Concepts:** Memory Interleaving: Concepts and Benefits, Processor-Cache Interactions: Handling Read/Write Requests, Basic Cache Optimizations: Write-Through vs. Write-Back Caches

**Storage Technologies:** Introduction to Magnetic Disks: Tracks, Sectors, Disk Scheduling Algorithms, Flash Memory Technology: Structure and Performance Characteristics

**Cache Memory:** Different Indexing Mechanisms: Direct-Mapped, Set-Associative, Fully Associative Caches, Trade-offs related to Block Size, Associativity, and Cache Size, Processor-Cache Interactions for Read/Write Requests, Basic Optimizations: Write-Through vs. Write-Back Caches, Average Memory Access Time (AMAT), Cache Replacement Policies: Least Recently Used (LRU), First-In-First-Out (FIFO).

**Learning Experiences**

**Classroom Learning Experience**

1. **Interactive Lectures**: Introduce key concepts in computer organization using PPTs and diagrams.

2. **Conceptual Understanding**: Cover topics like CPU architecture, memory hierarchy, and I/O systems.
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on assembly language and instruction set architecture.
4. **Case Studies**: Analyze real-world computer architectures and their performance metrics.
5. **Group Work**: Collaborate on projects that involve designing and simulating computer systems.
6. **Continuous Feedback**: Implement quizzes and peer reviews to assess understanding of organizational concepts.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign projects that require application of concepts in computer organization and architecture.
2. **Lab Projects**: Facilitate hands-on tasks involving hardware simulations and assembly programming.
3. **Question Bank**: Provide practice problems and resources for self-assessment on architecture topics.
4. **Online Forums**: Create platforms for discussing challenges and sharing solutions related to computer architecture.
5. **Self-Study for Case Studies**: Encourage independent research on current trends in computer organization.
6. **Collaborative Projects**: Organize group projects focused on building and optimizing computer architectures.

**Text Books:**

1. David A. Patterson and John L. Hennessy ,"Computer Organization and Design: The Hardware/Software Interface",5th Edition, Elsevier
2. Mano M. Morris, "Computer System Architecture", Pearson.

**Reference Books:**

1. CarlHamache, "Computer Organization and Embedded Systems", 6th Edition, McGraw Hill Higher Education
2. "Computer Architecture and Organization", 3rd Edition by John P. Hayes, WCB/McGraw-Hill
3. William Stallings "Computer Organization and Architecture: Designing for Performance", 10th Edition, Pearson Education

**Additional Readings:**

**Online Learning References**

a) **MIT OpenCourseWare - Computer System Engineering**
    a. **Link:** MIT OCW
    b. **Description:** This course provides a deep dive into computer system architecture, exploring processor design, memory systems, and parallel processing.

b) **GeeksforGeeks - Computer Organization and Architecture**
    a. **Link:** GeeksforGeeks
    b. **Description:** GeeksforGeeks provides detailed tutorials on various topics in computer organization and architecture, such as instruction sets, pipelining, and memory hierarchy.

c) **NPTEL - Computer Architecture**
    a. **Link:** NPTEL
    b. **Description:** This course from NPTEL covers the principles of computer architecture, including instruction sets, CPU design, and memory systems, with a focus on practical applications.

# COMPUTER NETWORKS

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Computer Networks** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCS304** | 4-0-0 | 4 | 40 |
| **Type of Course:** | Major-28 | | | |
| **Pre-requisite(s), if any: Basics of Communication Engineering and Data structures** | | | | |

**Course Perspective:** The course provides a comprehensive introduction to the fundamental concepts and principles of computer networking. Students will gain an in-depth understanding of how data communication works, the different types of network topologies, and the protocols that govern data exchange over networks. The course covers key aspects such as network hardware and software components, network administration, and protocol selection for various communication services. By exploring topics such as the OSI model, error detection and correction, IP addressing, routing protocols, and network applications, students will develop the skills necessary to design, implement, and manage computer networks effectively. The course is divided into 4 modules:

Evolution of Computer Networking

a) Data Link Layer Design
b) Introduction to Network Layer and Transport Services
c) Application Layer

**Course Outcomes (COs).** On completion of the course the participants will be able to:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the fundamental concepts and principles of computer networks. |
| **CO 2** | **Applying** knowledge of network hardware and software components. |
| **CO 3** | **Developing** skills in network administration and management. |
| **CO 4** | **Identifying** appropriate protocol for desired communication service. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| **Unit Number: 1** | **Title: Evolution of Computer Networking** | **No. of hours: 10** |
|---|---|---|
| **Content Summary:** Data communication Components: Representation of data and its flow Networks, Various Connection Topology, Protocols and Standards, OSI model, Access networks, physical media, Forwarding, routing; packet switching; circuit switching; a network of network, packet delay and loss, end to end throughput. | | |
| **Unit Number: 2** | **Title: Data Link Layer** | **No. of hours: 10** |
| **Content Summary:** Data Link Layer and Medium Access Sub Layer: Error Detection and Error Correction - Fundamentals, Block coding, Hamming Distance, CRC; Flow Control and Error control protocols - Stop and Wait, Go back – N ARQ, Selective Repeat ARQ, Sliding Window, Piggybacking, Random Access, Multiple access protocols -Pure ALOHA, Slotted ALOHA, CSMA/CD,CDMA/CA. | | |
| **Unit Number: 3** | **Title: Introduction to Network Layer and Transport Services** | **No. of hours: 10** |
| **Content Summary:** Network Layer: Switching, Logical addressing – IPV4, IPV6; Address mapping – ARP, RARP, BOOTP and DHCP–Delivery, Forwarding and Unicast Routing protocols. Transport Layer: Process to Process Communication, User Datagram Protocol (UDP), Transmission Control Protocol (TCP), SCTP Congestion Control; Quality of Service, QoS improving techniques: Leaky Bucket and Token Bucket algorithm. | | |
| **Unit Number: 4** | **Title: Application Layer** | **No. of hours: 10** |
| **Content Summary:** Application Layer: Domain Name Space (DNS), DDNS, TELNET, EMAIL, File Transfer Protocol (FTP), WWW, HTTP, SNMP, Bluetooth, Firewalls, Basic concepts of Cryptography. | | |

**Learning Experiences**

**Classroom Learning Experience**

1. **Interactive Lectures**: Introduce key concepts in computer networks using PPTs and network diagrams.
2. **Conceptual Understanding**: Cover topics like network protocols, topologies, and architecture models (OSI, TCP/IP).
3. **Problem-Solving Sessions**: Conduct in-class exercises focused on troubleshooting network issues and configuring devices.
4. **Case Studies**: Analyze real-world network designs and their performance implications.

5. **Group Work**: Collaborate on projects that involve designing and simulating network setups.
6. **Continuous Feedback**: Implement quizzes and peer reviews to assess understanding of networking concepts.

**Outside Classroom Learning Experience**

1. **Theory Assignments**: Assign projects that require practical application of networking principles.
2. **Lab Projects**: Facilitate hands-on tasks involving network configuration and management tools.
3. **Question Bank**: Provide practice problems and resources for self-assessment on networking topics.
4. **Online Forums**: Create platforms for discussing networking challenges and sharing solutions.
5. **Self-Study for Case Studies**: Encourage independent research on current trends and technologies in computer networking.
6. **Collaborative Projects**: Organize group projects focused on developing efficient network solutions for real-world scenarios.

**Textbooks:**

T1: " Data Communication and Networking", 5th Edition, Behrouz A. Forouzan, McGraw-Hill, 2012.

T2: "Computer Networks", Andrew S. Tanenbaum and David J. Wetherall, Pearson, 5th Edition, 2010.

T3: "Data and Computer Communication", 8th Edition, William Stallings, Pearson Prentice Hall Indi

**Reference Books**

R1: "Computer Networking A Top-Down Approach". 5th Edition, James F. Kurose-Keith W. Ross (Pearson).

R3. "Computer Networks – Protocols, Standards and Interfaces". 2nd Edition –Uyless Black (Prentice Hall of India Pvt. Ltd.

**Additional Readings:**

**Online Learning References**

I) **MIT OpenCourseWare - Computer Networks**
   a. **Link:** MIT OCW
   b. **Description:** This course provides a thorough exploration of computer networks, focusing on network design, protocol layers, and network management.

II) **GeeksforGeeks - Computer Networks**
   a. **Link:** GeeksforGeeks
   b. **Description:** GeeksforGeeks provides detailed tutorials on various aspects of computer networks, such as the OSI model, data link layer, network layer, and transport layer protocols.

**III) NPTEL - Computer Networks**

a. **Link:** NPTEL

b. **Description:** This course from NPTEL provides a comprehensive overview of computer networking, including topics like error detection, IP addressing, and routing protocols.

# COMPUTER NETWORKS LAB

**Program Name:** **B. Tech (Computer Science and Engineering)**

| **Course Name:** **Computer Networks Lab** | **Course Code** | **L-T-P** | **Credits** |
|---|---|---|---|
| | **ENCS352** | 0-0-2 | 1 |

**Type of Course:** Major-30

**Pre-requisite(s), if any: Basics of Networking**

**Defined Course Outcomes**

| COs | Statements |
|---|---|
| CO 1 | **Applying** fundamental networking concepts and techniques to develop and analyze network topologies, protocols, and error detection mechanisms. |
| CO 2 | **Designing** and implement network protocols and architectures for efficient data communication and management in various environments. |
| CO 3 | **Utilizing** advanced networking techniques to implement, monitor, and optimize communication systems for real-time and multimedia applications. |
| CO 4 | **Integrating** IoT devices and develop smart systems using networking principles for automation and efficient data management. |

**Lab Experiments**

| **Ex. No** | **Experiment Title** | **Mapped CO/COs** |
|---|---|---|
| 1 | Design and simulate a simple computer network using various connection topologies (bus, star, ring, mesh). Compare the advantages and disadvantages of each topology in terms of data flow and network efficiency. | CO 1 |
| 2 | Create a network simulation to demonstrate packet switching and circuit switching. Compare the performance and efficiency of both methods by simulating a series of data transmission scenarios. | CO 1 |
| 3 | Develop a network simulator to analyze packet delay, loss, and end-to-end throughput. Implement various routing algorithms and measure their impact on network performance under different traffic conditions. | CO1 |

| | | |
|---|---|---|
| 4 | Implement error detection and correction mechanisms using block coding and CRC. Simulate a communication system that demonstrates how errors are detected and corrected during data transmission. | CO2 |
| 5 | Design and simulate flow control and error control protocols such as Stop and Wait, Go-Back-N ARQ, and Selective Repeat ARQ. Compare their performance in terms of throughput and efficiency under varying network conditions. | CO2 |
| 6 | Develop a simulation to demonstrate multiple access protocols such as Pure ALOHA, Slotted ALOHA, CSMA/CD, and CSMA/CA. Analyze the performance of each protocol in handling network collisions and maximizing data transmission efficiency | CO2 |
| 7 | Implement a sliding window protocol with piggybacking for efficient data transmission and error control. Simulate data transfer between two nodes and visualize the window movements and acknowledgments. | CO2 |
| 8 | Create a simulation to demonstrate logical addressing using IPv4 and IPv6. Implement address mapping techniques such as ARP, RARP, BOOTP, and DHCP to show how devices acquire and resolve network addresses. | CO3 |
| 9 | Implement a transport layer simulation to demonstrate process-to-process communication using UDP, TCP, and SCTP. Compare the protocols in terms of connection establishment, data transmission, and congestion control. | CO3 |
| 10 | Implement a DNS and DDNS simulation to demonstrate domain name resolution and dynamic updates. Create a simple client-server application that queries and updates the DNS records. | CO4 |
| 11 | Create a web server simulation to demonstrate the workings of HTTP and WWW. Implement basic HTTP request and response handling, and simulate a simple web browsing session. | CO4 |

## INTRODUCTION OF NEURAL NETWORK AND DEEP LEARNING

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name: Introduction to Neural Network and Deep learning** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENCS306** | 4-0-0 | 4 | 40 |
| **Type of Course:** | Major-29 | | | |
| **Pre-requisite(s), if any: Fundamentals of AI and Machine Learning, Programming knowledge** | | | | |

**Course Perspective.** This course provides a comprehensive introduction to the fundamental concepts of neural networks and deep learning, which are crucial for the development of intelligent systems and advanced data analysis. Students will explore core topics including the basic structure and function of neural networks, feedforward neural networks, deep learning techniques, and probabilistic neural networks. The course emphasizes both theoretical understanding and practical implementation, preparing students to tackle real-world problems using advanced neural network models.

**The Course Outcomes (COs).** On completion of the course the participants will be able to:

| COs | **Statements** |
|---|---|
| **CO 1** | **Understanding** key concepts of neural networks and deep learning, including ANNs and their biological equivalents. |
| **CO 2** | **Implementing** basic neural network models and training algorithms using popular deep learning frameworks |
| **CO 3** | **Comparing** and **contrasting** different deep learning architectures, such as CNNs, RNNs, and GANs, and their applications in various domain |
| **CO 4** | **Assessing and optimizing** deep learning models by applying regularization techniques, tuning hyperparameters, and evaluating performance metrics. |

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number: 1** | **Title: Introduction to Neural Networks** | **No. of hours: 10** |
| **Content:**<br>▪ Human Brain and Artificial Neuron Models<br>▪ Biological vs. Artificial Neural Networks<br>▪ Evolution and Characteristics of Neural Networks<br>▪ Learning Methods: Supervised, Unsupervised, Reinforcement<br>▪ Taxonomy of Neural Network Architectures | | |
| **Unit Number: 2** | **Title: Supervised and Unsupervised Neural Networks** | **No. of hours: 10** |
| **Content:**<br>**Supervised learning:** - Supervised Learning Networks, Perceptron Networks, Adaptive Linear Neuron, Back-propagation Network. Associative Memory Networks. Training Algorithms for pattern association.<br>**Unsupervised learning**: - Introduction, Fixed Weight Competitive Nets, Maxnet, Hamming Network, Kohonen Self-Organizing Feature Maps, Learning Vector Quantization, Counter Propagation Networks, Adaptive Resonance Theory Networks. | | |
| **Unit Number: 3** | **Title: Deep learning and Regularization for Deep Learning** | **No. of hours: 10** |
| **Content:**<br>**Introduction to Deep Learning**: Historical Trends and Development, Deep Feed-Forward Networks, Gradient-Based Learning, Architecture Design and Hidden Units, Back-Propagation and Differentiation Algorithms<br>**Regularization Techniques**: Parameter Norm Penalties, Data Augmentation and Noise Robustness, Semi-Supervised Learning and Multi-Task Learning, Early Stopping, Sparse Representations, Bagging and Ensemble Methods | | |
| **Unit Number: 4** | **Title: Optimization for Train Deep Models** | **No. of hours: 10** |
| **Content:**<br>**Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM), Generative Adversarial Networks (GANs);** | | |

**Optimization Techniques:** Parameter Initialization Strategies, Adaptive Learning Rates (Adam, RMSprop), Regularization Techniques (Dropout, Batch Normalization), Advanced Optimization Algorithms (Stochastic Gradient Descent, Mini-Batch Gradient Descent)

**Applications and Trends:** Large-Scale Deep Learning Applications, Computer Vision, Speech Recognition, Natural Language Processing, Recent Trends and Research in Deep Learning

**Learning Experiences:**

**Inside Classroom Learning Experience:**

1. **Interactive Lectures and PPTs:** Students will engage with concept-focused lectures, supported by clear and concise PowerPoint presentations to aid understanding of neural network and deep learning fundamentals.
2. **Collaborative Learning:** Group projects and peer-to-peer learning will foster teamwork and collaborative problem-solving, with opportunities for peer reviews and group discussions.
3. **Continuous Assessment & Feedback:** Regular quizzes, assignments, and lab assessments will provide timely feedback, with additional support available from the course instructor as needed.

**Outside Classroom Learning Experience:**

1. **Hands-on Lab Assignments:** Through project-based lab assignments, students will implement key algorithms like CNNs, RNNs, and GANs, enhancing their practical skills in deep learning.
2. **Problem-Based Assignments:** Theory assignments will challenge students to solve real-world problems, encouraging critical thinking and the application of theoretical knowledge to practical scenarios.
3. **Collaborative Learning:** Students will continue to collaborate on group projects outside of class, working together to solve problems and review each other's work, encouraging peer-to-peer learning beyond the classroom.
4. **Use of Technology & ICT Tools:** Moodle LMS will be used for uploading lecture notes, assignments, and additional learning resources, allowing students to access material anytime and stay organized.
5. **Case Studies and Recent Trends:** Students will analyze case studies on neural network applications, staying updated with the latest research and industry trends in deep learning, making the learning experience industry-relevant.
6. **Continuous Assessment & Feedback:** Students can access additional support and feedback from the instructor through email, online forums, or scheduled one-on-one sessions. This provides extra guidance for those who need it outside regular class hours.

**Textbooks:**

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
- Aggarwal, C. C. (2018). Neural Networks and Deep Learning: A Textbook. Springer.
- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
- Shanmugamani, R. (2018). Deep Learning for Computer Vision. Packt Publishing.
- Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.

**Online Learning References**

1. **Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville**
    - Deep Learning Book
2. **Neural Networks and Deep Learning by Michael Nielsen**
    - Neural Networks and Deep Learning
3. **CS231n: Convolutional Neural Networks for Visual Recognition by Stanford University**
    - CS231n Course
4. **Deep Learning Specialization by Andrew Ng on Coursera**
    - Deep Learning Specialization
5. **Introduction to Deep Learning by MIT OpenCourseWare**
    - MIT OpenCourseWare

# INTRODUCTION TO NEURAL NETWORKS & DEEP LEARNING LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name: Introduction to Neural Network and Deep learning Lab** | **Course Code** | **L-T-P** | **Credits** |
| | **ENCS354** | 0-0-2 | 1 |
| **Type of Course:** | Major-31 | | |

**Pre-requisite(s), if any: fundamentals of AI and programming.**

**Defined Course Outcomes**

| COs | |
|---|---|
| CO 1 | **Demonstrating** the ability to implement and analyze basic artificial neural network models and classical machine learning algorithms. |
| CO 2 | **Developing** and evaluate advanced neural network architectures and unsupervised learning models for complex data analysis. |
| CO 3 | **Applying** deep learning techniques to real-world applications, including image and speech recognition, and optimize model performance. |
| CO 4 | **Implementing** and optimize neural network models using advanced techniques such as adaptive learning rates, regularization, and meta-optimization for enhanced performance. |

## LAB EXPERIMENTS

| Experiment No. | Problem Statement | Mapped COs |
|---|---|---|
| 1 | Basic Neural Network Implementation: Implement a simple feedforward neural network to classify handwritten digits from the MNIST dataset. | CO 1, CO 2 |
| 2 | Perceptron Algorithm: Develop and train a Perceptron to perform binary classification on a given dataset, and visualize the decision boundary. | CO 1, CO 2 |
| 3 | Backpropagation Network: Implement a neural network with backpropagation to predict house prices based on features like size, location, etc. | CO 1, CO 2, CO 3 |

| Experiment No. | Problem Statement | Mapped COs |
|---|---|---|
| 4 | Kohonen Self-Organizing Maps (SOM): Apply a Self-Organizing Map to cluster and visualize a dataset with multiple features. | CO 1, CO 2 |
| 5 | Convolutional Neural Network (CNN): Build and train a CNN for image classification on the CIFAR-10 dataset. | CO 2, CO 3, CO 4 |
| 6 | Recurrent Neural Network (RNN) for Text Prediction: Develop an RNN to predict the next word in a sentence using the IMDB movie reviews dataset. | CO 2, CO 3 |
| 7 | Regularization Techniques: Implement dropout and L2 regularization in a neural network to improve model performance on the MNIST dataset. | CO 3, CO 4 |
| 8 | Optimization Algorithms: Compare the performance of different optimization algorithms (SGD, Adam, RMSprop) in training a CNN on a benchmark dataset. | CO 4 |
| 9 | Generative Adversarial Network (GAN): Create and train a GAN to generate new images similar to a given dataset (e.g., faces, landscapes). | CO 2, CO 3 |
| 10 | Transfer Learning: Use a pre-trained deep learning model (e.g., VGG16, ResNet) and fine-tune it for a new image classification task with a smaller dataset. | CO 2, CO 4 |

**Online Learning References**

1. Deep Learning Specialization by Andrew Ng on Coursera
   - Deep Learning Specialization
2. Neural Networks and Deep Learning by Michael Nielsen
   - Neural Networks and Deep Learning
3. CS231n: Convolutional Neural Networks for Visual Recognition by Stanford University
   - CS231n Course
4. Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
   - Deep Learning Book
5. MIT OpenCourseWare - Introduction to Deep Learning
   - MIT OpenCourseWare

## (DEPARTMENT ELECTIVE -II)

## NATURAL LANGUAGE PROCESSING

**Program Name:** **B. Tech (Computer Science and Engineering)**

| **Course Name:** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
|---|---|---|---|---|
| **NATURAL LANGUAGE PROCESSING** | **ENSP302** | 4-0-0 | 4 | 40 |

**Type of Course:** DSE-3

**Pre-requisite(s), if any:** Strong programming skills, particularly in Python.

**Course Perspective.** This course offers a comprehensive introduction to the principles and techniques of Natural Language Processing (NLP). It bridges the gap between theoretical linguistics and practical implementation using machine learning and deep learning algorithms. NLP is pivotal in developing applications that can interpret and respond to human language, impacting areas such as artificial intelligence, data analysis, and information retrieval. The course emphasizes hands-on learning through real-world applications, ensuring that students gain practical skills in processing and analyzing natural language. The course is divided into 4 modules:

- Foundations of Natural Language Processing
- Machine & Deep Learning in Natural Language Processing
- Advanced Text and Speech Processing
- Ethics, Fairness, and Practical Applications in NLP

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Grasping** NLP fundamentals, resolve linguistic ambiguities, and employ ML/DL in NLP for both text and speech. |
| **CO 2** | **Applying** techniques for processing and analyzing natural language to develop intelligent interpretation systems. |
| **CO 3** | **Designing** and assess advanced NLP models for essential tasks, utilizing transformers, GRUs, and more. |

| | |
|---|---|
| **CO 4** | **Innovating** with NLP and speech recognition for real-world applications, incorporating cutting-edge concepts. |
| **CO 5** | **Evaluating** NLP and speech technologies' fairness and ethics, striving for inclusive and meaningful solutions. |

**Course Outline:**

**Unit Number: 1 Title: Foundations of Natural Language Processing No. of hours: 10**

**Content Summary:**

- Introduction to NLP and its applications
- Basic text processing and analysis
- Python programming for NLP
- Language Modeling:
  - N-grams
  - Smoothing techniques
- Morphology and Parts of Speech (PoS) Tagging
- Syntax:
  - Probabilistic Context-Free Grammars (PCFGs)
  - Dependency Parsing
- Lexical Semantics and Word Sense Disambiguation
- Distributional Semantics:
  - Vector Space Models
  - Word2Vec
  - Introduction to GloVe
- Topic Models:
  - Latent Dirichlet Allocation (LDA)

**Unit Number: 2 Title: Machine & Deep Learning in Natural Language Processing No. of hours: 12**

**Content Summary:**

- Neural Networks for NLP
- Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) Networks
- Gated Recurrent Units (GRUs) and comparison with LSTMs

- Deep Learning Architectures for NLP
- Neural and Pre-Trained Language Models:
  - Introduction and Applications
  - Advanced Pre-Training Techniques
- Generative Models:
  - Few-Shot Learning
  - Prompt Learning
  - Applications
- Recursive Neural Networks

| Unit Number: 3 | Title: Advanced Text and Speech Processing | No. of hours: 10 |
|---|---|---|

**Content Summary:**

- Advanced Text Classification Techniques:
  - Naïve Bayes
  - Logistic Regression
  - Support Vector Machines (SVMs)
- Applications in:
  - Sentiment Analysis
  - Opinion Mining
  - Text Summarization
- N-gram Features and Enhancements through CNNs
- Multilingual NLP Challenges
- Advanced Entity Linking
- Introduction to Speech Processing:
  - Speech Recognition
  - Speech Synthesis
  - Speech-to-Text and Text-to-Speech Technologies
  - Speaker Recognition and Verification

| Unit Number: 4 | Title: Ethics, Fairness, and Practical Applications in NLP | No. of hours: 8 |
|---|---|---|

**Content Summary:**

- Fairness and Ethics in NLP
- Development of Real-World NLP Solutions:
  - Application-Oriented Projects

    - Spelling Correction
    - Robust Language Modeling
    - Ethical Speech Recognition Design
- Multilingual Speech Processing
- Applications of Large Language Models:
    - Creative Text Generation
    - Summarization
    - Code Generation
- Practical Applications of Speech Recognition:
    - Assistive Technologies
    - Automated Customer Service
    - Real-Time Transcription Services

**Learning Experiences:**

Inside Classroom Learning Experience:

1. **Lecture PPTs:** In-class PowerPoint presentations provide clear explanations of core Natural Language Processing (NLP) concepts, helping students grasp theoretical foundations.
2. **Continuous Assessment**: Regular in-class assessments such as quizzes and exercises will track student progress, providing immediate feedback and identifying areas where students need improvement.
3. **ICT Tools:** Interactive boards and in-class usage of Moodle LMS will support seamless access to learning materials, fostering an interactive learning environment.

**Outside Classroom Learning Experience:**

- **Problem-Based Assignments:** Students will complete problem-based assignments outside of class, which enhance critical thinking and the real-world application of NLP concepts.
- **Project-Based Lab Assignments:** Hands-on lab assignments will be completed outside class, offering practical experience with NLP tools, allowing students to apply theoretical concepts to real-world scenarios.
- **Question Banks:** Students will have access to comprehensive question banks outside of class, ensuring thorough coverage of the syllabus and preparing them for exams.
- **Model Question Papers:** Outside the classroom, students will work on model question papers to familiarize themselves with exam formats and question types, improving their readiness for assessments.

- **ICT Tools:** Students will use Moodle LMS outside the classroom to access learning materials, submit assignments, and review additional resources, ensuring continuous and organized learning.

**Text Books**:

1. Daniel Jurafsky and James H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*, Pearson Education India.
2. Steven Bird, Ewan Klein, and Edward Loper, *Natural Language Processing with Python*, O'Reilly Media.

**References**

1. "Speech and Language Processing" by Dan Jurafsky and James H. Martin
2. "Natural Language Processing in Action: Understanding, Analyzing, and Generating Text with Python" by Hobson Lane, Cole Howard, and Hannes Hapke
3. "Neural Network Methods for Natural Language Processing" by Yoav Goldberg
4. "Introduction to Information Retrieval" by Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze
5. "Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems" by Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, and Harshit Surana

**Additional Readings:**

**Online Learning Resources:**

1. **Stanford University - Natural Language Processing with Deep Learning (CS224N)**
   - Offered by Stanford University, this course covers the foundational concepts of NLP along with the application of deep learning models in NLP tasks. The lectures are available for free.
   - Link: Stanford CS224N
2. **NLTK Book - Natural Language Processing with Python**
   - An excellent resource for beginners, this book provides a practical introduction to NLP with Python, using the Natural Language Toolkit (NLTK). It's available for free online.
   - Link: NLTK Book
3. **fast.ai - Practical Deep Learning for Coders**

- Though not exclusively focused on NLP, this course includes several lessons on applying deep learning to natural language processing. It is practical, with a focus on coding and real-world applications.
- Link: fast.ai Course

**Open-Source Society University (OSSU)**

I) **OSSU Computer Science**

a. OSSU provides an open-source curriculum for learning computer science. While it covers a broad range of topics in computer science, it includes resources for learning about artificial intelligence and machine learning, which are relevant to students interested in NLP.

b. Link: OSSU Computer Science

# NATURAL LANGUAGE PROCESSING LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **NATURAL LANGUAGE PROCESSING LAB** | **ENSP352** | 0-0-2 | 1 |
| **Type of Course:** | DSE-4 | | |

**Pre-requisite(s), if any:** Strong programming skills, particularly in Python.

**Defined Course Outcomes**

| COs | |
|---|---|
| CO 1 | **Demonstrating** proficiency in implementing and utilizing basic and advanced natural language processing (NLP) techniques for text analysis and processing. |
| CO 2 | **Developing** fine-tune, and evaluate machine learning and deep learning models for various NLP tasks, including text classification, named entity recognition, and sentiment analysis. |
| CO 3 | **Implementing** and assess advanced text and speech processing models, focusing on performance improvement and practical applications in real-world scenarios. |
| CO 4 | **Analyzing** and address ethical considerations, fairness, and privacy implications in NLP applications, ensuring responsible and unbiased technology deployment. |

## List of Experiments

| Project Title | Project Statement | CO Mapped |
|---|---|---|
| Text Processing and Analysis Tool | Develop a Python-based tool for basic text processing and analysis. Implement functionalities such as tokenization, stemming, lemmatization, and Parts of Speech (PoS) tagging. Include language modeling using N-grams and smoothing techniques. | CO1 |
| Word Representation and Topic Modeling | Create a Python application to perform word representation using Word2Vec and GloVe. Implement a topic modeling system using Latent Dirichlet Allocation (LDA) to analyze and categorize a given corpus of text data. | CO1 |
| Sentiment Analysis using Recurrent Neural Networks (RNNs) | Build a sentiment analysis application using Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. Train the model on a labeled dataset and evaluate its performance in classifying sentiments. | CO2 |
| Language Model with Gated Recurrent Units (GRUs) | Develop a language model using Gated Recurrent Units (GRUs). Compare its performance with LSTM-based models on the same dataset. Implement applications for text generation based on the trained language model. | CO2 |
| Advanced Text Classification and Sentiment Analysis | Implement text classification techniques using Naïve Bayes, Logistic Regression, and SVMs. Apply these techniques to sentiment analysis, opinion mining, and text summarization tasks. Enhance feature extraction using N-grams and CNNs. | CO3 |
| Speech Recognition and Synthesis System | Develop a basic speech recognition and synthesis system. Implement speech-to-text and text-to-speech functionalities using available libraries. Extend the system to include speaker recognition and verification. | CO3 |

| Project Title | Project Statement | CO Mapped |
|---|---|---|
| Ethical NLP System for Fair Language Modeling | Design an NLP system focusing on ethical considerations and fairness. Implement language modeling for robust applications like spelling correction. Ensure the system avoids biases and treats all user inputs fairly. | CO4 |
| Real-World NLP Application: Automated Customer Service | Create an automated customer service application using advanced language models. Implement functionalities such as real-time transcription services and multilingual speech processing. Ensure ethical and fair use of NLP technologies. | CO4 |

# IMAGE PROCESSING & COMPUTER VISION

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| **Image Processing & Computer Vision** | **ENSP304** | 4-0-0 | 4 | |
| **Type of Course:** | DSE-3 | | | |

**Pre-requisite(s), if any: (1) Linear Algebra and (2) programming in python**

**Course Perspective.** This course introduces students to the essential concepts and techniques of image processing and computer vision. It bridges theoretical knowledge with practical applications, enabling students to understand and implement various algorithms for image enhancement, restoration, segmentation, and object recognition. The course is designed to equip students with the skills required to tackle real-world challenges in fields such as robotics, medical imaging, surveillance, and multimedia applications.

**Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Remembering** key concepts, definitions, and algorithms in image processing and computer vision. |
| **CO 2** | **Understanding** the principles and applications of essential techniques like edge detection and feature extraction. |
| **CO 3** | **Applying** Use image processing methods to enhance digital images and analyze the effects. |
| **CO 4** | **Evaluating** the effectiveness of various computer vision models in different contexts |
| **CO 5** | **Designing** and implement projects that integrate multiple image processing algorithms to solve complex problems. |

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number: 1** | **Title: Title: Introduction to Basic Concepts of Image Formation** | **No. of hours: 10** |

**Content Summary:**

**Fundamentals and Applications of Image Processing:**

- Overview of Image Processing and its applications.
- Components of Image Processing Systems.

**Image Sensing and Acquisition:**

- Image sensing techniques.
- Image acquisition methods.

**Sampling and Quantization:**

- Concept of sampling and quantization.
- Neighbors of pixel adjacency and connectivity.
- Regions and boundaries.
- Distance measures.

**Image Enhancement:**

- Frequency and Spatial Domain techniques.
- Contrast Stretching.
- Histogram Equalization.
- Low pass and High pass filtering.

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: Image Restoration and coloring** | **No. of hours: 12** |

**Content Summary:**

**Image Restoration:**

- Model of the Image Degradation/Restoration Process.
- Noise Models.
- Restoration in the presence of noise using spatial filtering.
- Periodic Noise Reduction using frequency domain filtering.
- Linear Position Invariant Degradations.
- Estimation of Degradation Function.
- Inverse Filtering.
- Wiener Filtering.
- Constrained Least Square Filtering.
- Geometric Mean Filter.
- Geometric Transformations.

**Color Image Processing:**

- Color models and transformations.
- Image Segmentation using color.
- Texture Descriptors.
- Color Features.
- Edge/Boundary detection.
- Object Boundary and Shape Representations.
- Interest or Corner Point Detectors.
- Speeded-Up Robust Features (SURF).
- Saliency detection.

**Unit Number: 3** **Title: Image Compression and Segmentation** **No. of hours: 10**

**Content Summary:**

**Image Compression:**

- Data Redundancies.
- Image Compression models.
- Elements of Information Theory.
- Lossless and Lossy Compression.
- Huffman Coding.
- Shanon-Fano Coding.
- Arithmetic Coding.
- Golomb Coding.
- LZW Coding.
- Run Length Coding.
- Lossless Predictive Coding.
- Bit Plane Coding.
- Image compression standards.

**Image Segmentation and Morphological Image Processing:**

- Discontinuity-based segmentation.
- Similarity-based segmentation.
- Edge linking and boundary detection.
- Thresholding.
- Region-based Segmentation.
- Introduction to Morphology.

- Dilation and Erosion.
- Basic Morphological Algorithms.

| Unit Number: 4 | Title: Object Representation and Computer Vision Techniques | No. of hours: 8 |
|---|---|---|

**Content Summary:**

**Representation and Description:**

- Introduction to Morphology.
- Basic Morphological Algorithms.
- Representation Techniques.
- Boundary Descriptors.
- Regional Descriptors.
- Chain Code.
- Structural Methods.

**Computer Vision Techniques:**

- Review of Computer Vision Applications.
- Artificial Neural Networks for Pattern Classification.
- Convolutional Neural Networks (CNNs).
- Machine Learning Algorithms and their Applications

**Learning Experiences:**

Inside Classroom Learning Experience:

1. **Lecture PPTs:** Classroom lectures will use structured and clear PowerPoint presentations to introduce and reinforce key concepts, enhancing understanding.
2. **Project-Based Lab Assignments:** Hands-on lab assignments will be conducted in the classroom to provide practical experience, making learning more engaging and helping students apply theoretical knowledge.
3. **Continuous Assessment:** Regular quizzes and assessments conducted during class will track student progress, providing immediate feedback to guide improvements.
4. **Interactive Boards & Moodle LMS:** Interactive teaching boards and Moodle LMS will be used in class to ensure seamless access to resources, fostering a more interactive and engaging learning environment.

**Outside Classroom Learning Experience:**

1. **Problem-Based Assignments:** Students will work on problem-based assignments outside of class, encouraging deeper thinking and the application of theory to real-world scenarios.
2. **Project-Based Lab Assignments**: Lab assignments can be continued at home, allowing students to practice and further develop their skills with practical tools and techniques outside the classroom.
3. **Question Banks:** Comprehensive question banks will be available for students to review and self-assess their understanding of the syllabus, promoting thorough exam preparation.
4. **Model Question Papers:** Students will use model question papers at home to simulate exam conditions, helping them practice time management and familiarize themselves with the exam format.
5. **Continuous Assessment and Feedback:** Students will receive feedback on their progress through online assessments and homework submissions, with opportunities for additional guidance from the instructor.
6. **Video Lectures:** Video lectures will be available for students to review complex topics at their own pace, providing an additional resource for detailed learning and revision outside the classroom.
7. **Moodle LMS:** Moodle LMS will provide students with access to lecture materials, assignments, and additional resources outside the classroom, supporting continuous learning.
8. **Support and Feedback:** Students can seek support from the instructor outside of class, whether through office hours, email, or discussion forums, ensuring they receive extra guidance as needed.

**Textbooks**

1. Gonzalez Rafael C. and Woods Richard E., Digital Image Processing, New Delhi: Prentice– Hall of India.
2. Computer Vision: Algorithms and Applications by Richard Szeliski

**Online Learning References:**

I) **Fast.ai: Practical Deep Learning for Coders**
   a. Practical course on deep learning, including applications in image processing and computer vision.
   b. Link: Fast.ai - Practical Deep Learning for Coders

II) **Deep Learning for Computer Vision with Python by Adrian Rosebrock**
   a. A comprehensive book on deep learning techniques for computer vision applications.
   b. Link: PyImageSearch - Deep Learning for Computer Vision with Python

III) **GitHub: OpenCV Projects and Tutorials**
   a. Repository of projects and tutorials on OpenCV, a popular library for computer vision.
   b. Link: GitHub - OpenCV

IV) **Towards Data Science: Image Processing Tutorials**

a. A collection of tutorials on various image processing techniques and applications.

b. Link: Towards Data Science - Image Processing

V) **IEEE Xplore Digital Library: Image Processing and Computer Vision Papers**

a. Access to research papers and articles on the latest developments in image processing and computer vision.

b. Link: IEEE Xplore Digital Library

# IMAGE PROCESSING & COMPUTER VISION LAB

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Image Processing & Computer Vision Lab** | **Course Code** | **L-T-P** | **Credits** | |
| | **ENSP354** | 0-0-2 | 1 | |
| **Type of Course:** | DSE-4 | | | |

**Pre-requisite(s), if any: (1) Linear Algebra and (2) programming in python**

**Defined Course Outcomes**

| **COs** | **Statements** |
|---|---|
| CO 1 | **Applying** image processing techniques using Python libraries. |
| CO 2 | **Analyzing** and evaluate the effectiveness of different image enhancement algorithms. |
| CO 3 | **Implementing** image restoration algorithms and evaluate their performance in the presence of noise. |
| CO 4 | **Developing** image compression algorithms and analyze their impact on image quality. |
| CO 5 | **Formulating** computer vision techniques such as object detection and tracking, gesture recognition, and facial expression recognition using Python. |

## Lab Experiments

| **S.N** | **Experiment** | **CO** |
|---|---|---|
| **1** | Implement a program to acquire and display an image. Demonstrate the process of image sensing and acquisition, and explain the components involved in an image processing system | **CO1** |
| **2** | Develop a program to perform sampling and quantization on a given image. Visualize the effects of different sampling rates and quantization levels on image quality. | **CO1** |
| **3** | Implement image enhancement techniques in the spatial domain, including contrast stretching and histogram equalization. Compare the results before and after enhancement. | **CO1** |
| **4** | Apply frequency domain filtering to an image. Implement both low pass and high pass filters, and demonstrate their effects on the image in terms of noise reduction and edge enhancement. | **CO1** |
| **5** | Implement a noise model to simulate different types of noise (Gaussian, salt-and-pepper) in an image. Apply spatial filtering techniques to restore the image from the noisy version. | **CO2** |

| | | |
|---|---|---|
| **6** | Develop a program to perform periodic noise reduction using frequency domain filtering. Implement and compare inverse filtering and Wiener filtering for image restoration | **CO2** |
| **7** | Implement geometric transformations on an image, such as rotation, scaling, and translation. Demonstrate how these transformations affect the image quality and geometry. | **CO2** |
| **8** | Perform color image processing by converting an image from RGB to another color model (e.g., HSV, YCbCr). Implement and visualize color-based image segmentation techniques. | **CO2** |
| **9** | Implement Huffman Coding and Run Length Coding for image compression. Compare the compression ratios and efficiency of these techniques on different types of images. | **CO3** |
| **10** | Develop a program to perform image segmentation using edge detection techniques. Implement edge linking and boundary detection algorithms to segment objects within an image. | **CO3** |
| **11** | Apply morphological operations, such as dilation and erosion, on a binary image. Implement basic morphological algorithms to enhance the structure and features of the objects in the image. | **CO3** |
| **12** | Implement threshold-based and region-based segmentation techniques. Compare their effectiveness in segmenting different types of images and objects. | **CO3** |
| **13** | Implement boundary and regional descriptors to represent the shape and features of objects in an image. Use chain codes and structural methods to describe object | **CO4** |
| **14** | Develop a basic convolutional neural network (CNN) for image classification. Train the CNN on a dataset and evaluate its performance in recognizing different classes of images. | CO4 |
| **15** | Implement a motion estimation algorithm to track moving objects in a video sequence. Demonstrate object tracking by visualizing the trajectories of the tracked objects. | CO4 |
| **16** | Create a system for face and facial expression recognition using machine learning algorithms. Implement feature extraction techniques and train a classifier to recognize different expressions and identities. | CO4 |

# INTRODUCTION TO GENERATIVE AI

|  |  |  |  |  |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| **Introduction to Generative AI** | **ENSP306** | 4-0-0 | 4 | 40 |
| **Type of Course:** | DSE-3 | | | |
| **Pre-requisite(s), if any:** | | | | |

**Course Perspective.** This course provides an in-depth introduction to the principles, techniques, and applications of Generative Artificial Intelligence (AI). Generative AI is a rapidly evolving field that has the potential to transform numerous industries by enabling machines to create content, predict outcomes, and enhance decision-making processes. This course will cover foundational concepts, the latest advancements in generative models, and practical applications, ensuring students gain a comprehensive understanding of the subject.

**Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** generative AI principles, recent advancements, and applications. |
| **CO 2** | **Applying** probability theory and statistics in generative AI tasks. |
| **CO 3** | **Designing** deep learning models for generative tasks.. |
| **CO 4** | **Evaluating** generative models for specific applications. |
| **CO 5** | **Assessing** ethical implications and propose solutions for generative AI. |

**Course Outline:**

| **Unit Number: 1** | **Title: Foundations of Generative AI** | **No. of hours: 10** |
|---|---|---|

**Content:**

- **Introduction to Generative AI:** Definition, working principles, and recent advancements.
- **Generative Modeling:**
    - Overview of Generative vs. Discriminative Models.
    - Introduction to Probabilistic Generative Models.
    - Naive Bayes as a basic generative model.
    - Challenges in Generative Modeling.
    - Introduction to Representation Learning.

| | | |
|---|---|---|
| **Unit Number: 2** | **Title: Deep Learning** | **No. of hours: 10** |

**Content:**

- Overview of Structured and Unstructured Data.
- Introduction to Deep Neural Networks using Keras and TensorFlow.
- Building and Training Deep Neural Networks:
    - Loading, building, compiling, training, and evaluating models.
    - Techniques to improve models: Convolutional layers, Batch normalization, and Dropout layers.
- Introduction to Autoencoders:
    - Building and understanding Variational Autoencoders (VAEs).
    - Using VAEs for tasks like face generation.

| | | |
|---|---|---|
| **Unit Number: 3** | **Generative Adversarial Networks** | **No. of hours: 10** |

**Content:**

- **Introduction to GANs:**
    - Roles of the Discriminator and Generator.
    - Training processes and challenges in GANs.
- **Advanced GAN Techniques:**
    - Wasserstein GAN (WGAN) and WGAN-GP.
- **Evaluation of GANs:**
    - Qualitative and Quantitative methods.
- **GAN Architectures and Applications.**.

| | | |
|---|---|---|
| **Unit Number: 4** | **Applications and Future Directions** | **No. of hours: 10** |

**Content:**

- **Real-World Applications of Generative AI:**
    - Image synthesis, data augmentation, healthcare, gaming, and art.

- ▪ **Ethical Considerations:**
  - Addressing bias, fairness, deepfakes, and responsible AI practices.
- ▪ **Emerging Trends:**
  - Overview of reinforcement learning, meta-learning, and tools like OpenAI's DALL-E.

**Learning Experiences:**

**Inside Classroom Learning Experience:**

1. **Lecture PPTs:** Provide clear explanations and visual aids to reinforce key concepts in Generative AI, ensuring students understand fundamental principles during classroom sessions.
2. **Problem-based theory assignments:** Theory assignments will encourage application of theoretical knowledge to real-world generative modeling challenges, fostering critical thinking and deeper learning.
3. **Project-based lab assignments:** Class lab sessions will offer hands-on experience, allowing students to build and evaluate AI models in an interactive and practical classroom environment.
4. **Continuous assessment:** Regular quizzes and assessments will be conducted in class to track student progress. Immediate feedback will be provided feedback during classroom sessions, ensuring students receive guidance on areas for improvement.

**Outside Classroom Learning Experience:**

1. **Question Banks:** Comprehensive question banks will be available for students to review and self-assess their understanding of all Generative AI topics covered in the course.
2. **Model Question Papers:** Comprehensive question banks will prepare students for exams by familiarizing them with the exam format and types of questions, enhancing their readiness for assessments.
3. **Continuous Assessment Feedback:** Students will receive feedback on their progress through online assessments and homework submissions, with opportunities to improve their understanding through homework, quizzes, and project evaluations.
4. **Support & Feedback:** Students can seek support with opportunities for one-on-one guidance, peer collaboration, and additional instructor support to enhance learning outcomes outside regular class hours.

**Textbooks**

1. Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play by David Foster

2. Hands-On Generative Adversarial Networks with Keras: Create Powerful Neural Networks for Real-World Applications by Rafael Valle

**Reference Books**

1. Generative Deep Learning, by David Foster, 2nd Edition, O'Reilly Media, Inc.
2. Deep Learning by Ian Goodfellow, Yoshua Bengio and Aaron Courville, The MIT Press
3. Pattern recognition and machine learning by Christopher M. Bishop

# INTRODUCTION TO GENERATIVE AI LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **Introduction to Generative AI Lab** | **ENSP356** | 0-0-2 | 1 |
| **Type of Course:** | DSE-4 | | |
| **Pre-requisite(s), if any: NA** | | | |

**Defined Course Outcomes**

| COs | Statements |
|---|---|
| CO 1 | **Designing,** developing, and evaluating deep neural networks for complex image classification tasks using advanced model improvement techniques. |
| CO 2 | **Implementing** and analyze variational autoencoders (VAEs) for anomaly detection in network traffic data, focusing on security threat identification. |
| CO 3 | **Developing** and enhance generative adversarial networks (GANs) for realistic image synthesis, emphasizing model architecture and performance improvement. |
| CO 4 | **Utilizing** generative AI techniques for data augmentation in healthcare and other creative fields, assessing the impact on model performance and addressing ethical considerations. |

## Lab Experiments

| **S.N** | **Experiments** | **COs** |
|---|---|---|
| 1 | Implement a basic probabilistic generative model using the Naive Bayes classifier. Train the model on a simple dataset and evaluate its performance. | CO1 |

| | | |
|---|---|---|
| | Discuss the challenges faced during the implementation and how they were addressed. | |
| 2 | Develop a representation learning model to learn and visualize the latent features of a given dataset. Implement the model using a suitable machine learning framework and analyze the results. | CO1 |
| 3 | Create a generative modeling framework to generate synthetic data from a given dataset. Compare the performance of generative versus discriminative modeling approaches on the same dataset. | CO1 |
| 4 | Implement a simple generative model to generate new data points from a probabilistic distribution. Discuss the working principles of generative AI and recent advancements in the field. | CO1 |
| 5 | Build and train a deep neural network using Keras and TensorFlow to classify images from the MNIST dataset. Evaluate the model's performance and improve it using techniques like convolutional layers, batch normalization, and dropout layers. | CO2 |
| 6 | Implement an autoencoder and a variational autoencoder (VAE) to compress and reconstruct images from a given dataset. Analyze the differences between the autoencoder and VAE in terms of performance and latent space representation. | CO2 |
| 7 | Design and implement a convolutional neural network (CNN) to classify images from the CIFAR-10 dataset. Train the model, evaluate its performance, and use techniques like data augmentation to improve its accuracy. | CO3 |
| 8 | Implement a basic GAN to generate synthetic images. Train the GAN on a simple image dataset, and evaluate the quality of the generated images using both qualitative and quantitative methods. | CO3 |
| 9 | Develop a Wasserstein GAN (WGAN) and train it on a dataset of images. Compare the performance of the WGAN with a standard GAN in terms of stability and quality of generated images. | CO3 |
| 10 | Implement and compare different GAN architectures and loss functions. Train each GAN on the same dataset and evaluate their performance using qualitative and quantitative methods. | CO3 |

| | | |
|---|---|---|
| 11 | Implement a generative AI model for image synthesis. Train the model on a dataset of images and evaluate the quality of the synthesized images. Discuss the potential applications of image synthesis in various fields. | CO4 |
| 12 | Develop a generative AI model for data augmentation and data generation. Train the model on a dataset with limited data and analyze how data augmentation improves the performance of a classifier trained on the augmented dataset. | CO4 |
| 13 | Create a generative AI application for healthcare, such as generating synthetic medical images for training purposes. Discuss the ethical considerations and challenges associated with using generative AI in healthcare. | CO4 |

# TRANSFER LEARNING

**Program Name:** **B. Tech (Computer Science and Engineering)**

| **Course Name:** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
|---|---|---|---|---|
| **Transfer Learning** | **ENSP308** | 4-0-0 | 4 | 40 |

**Type of Course:** DSE-3

**Pre-requisite(s), if any:**

**Course Perspective.** This course introduces students to the fundamental concepts and advanced techniques of transfer learning, an essential area in machine learning and deep learning. Transfer learning focuses on leveraging knowledge gained from one domain to improve learning in another domain. This course covers theoretical foundations, practical implementations, and applications across various domains, providing students with the skills necessary to apply transfer learning to real-world problems.

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the theoretical foundations, motivations, and applications of transfer learning. |
| **CO 2** | **Implementing** transfer learning algorithms proficiently using Python and deep learning libraries. |
| **CO 3** | **Applying** fine-tuning, feature extraction, and model adaptation techniques effectively across different domains and tasks. |
| **CO 4** | **Evaluating** transfer learning models using standard metrics and methodologies. |
| **CO 5** | **Analyzing** real-world applications and ethical implications of transfer learning for inclusive and meaningful solutions. |

**Course Outline:**

| Unit Number: 1 | Title: Foundations of Transfer Learning | No. of hours: 10 |
|---|---|---|

**Content Summary:**

**Introduction to Transfer Learning:**

- Overview of AI, Machine Learning, and Transfer Learning
- Relationship to Existing Machine Learning Paradigms
- Fundamental Research Issues and Applications

**Instance-Based Transfer Learning:**

- Instance-Based Noninductive Transfer Learning
- Instance-Based Inductive Transfer Learning

**Feature-Based Transfer Learning:**

- Minimizing Domain Discrepancy
- Learning Universal Features
- Feature Augmentation

**Model-Based Transfer Learning:**

- Transfer through Shared Model Components
- Transfer through Regularization

| Unit Number: 2 | Title: Advanced Transfer Learning Techniques | No. of hours: 10 |
|---|---|---|

**Content Summary:**

**Relation-Based Transfer Learning:**

- Markov Logic Networks (MLNs)
- Relation-Based Transfer Learning with MLNs

**Heterogeneous Transfer Learning:**

- Problem Definition and Methodologies
- Applications of Heterogeneous Transfer Learning

**Adversarial Transfer Learning:**

- Generative Adversarial Networks (GANs)
- Transfer Learning with Adversarial Models

**Transfer Learning in Reinforcement Learning:**

- Inter-task and Inter-domain Transfer Learning

| Unit Number: 3 | Title: Multi-task and Theoretical Aspects of Transfer Learning | No. of hours: 12 |
|---|---|---|

**Content:**

**Multi-task Learning:**

- Supervised, Unsupervised, and Semi-Supervised Learning
- Active, Reinforcement, and Online Learning
- Multi-view and Distributed Multi-task Learning

**Transfer Learning Theory:**

- Generalization Bounds for Multi-task Learning
- Generalization Bounds for Supervised and Unsupervised Transfer Learning

**Transitive Transfer Learning (TTL):**

- TTL over Mixed Graphs
- TTL with Hidden Feature Representations and Deep Neural Networks

**AutoTL: Learning to Transfer Automatically:**

- The L2T Framework
- Parameterizing and Inferring What to Transfer
- Connections to Other Learning Paradigms

| **Unit Number: 4** | **Title: Applications of Transfer Learning** | **No. of hours: 8** |
|---|---|---|

**Content Summary:**

**Specialized Learning Techniques:**

- Few-Shot, Zero-Shot, and One-Shot Learning
- Bayesian Program Learning and Poor Resource Learning

Transfer Learning Applications:

- Computer Vision and Medical Image Analysis
- Natural Language Processing and Sentiment Analysis
- Dialogue Systems and Spoken Language Understanding
- Natural Language Generation

**Learning Experiences:**

**Inside Classroom Learning Experience:**

1. **Lecture PPTs:** These lectures will provide clear explanations of key concepts and advanced techniques, helping students build a strong theoretical understanding.
2. **Problem-based Theory Assignments:** In-class assignments will focus on applying transfer learning concepts to solve real-world problems, promoting critical thinking and concept application.

3. **Project-based Lab Assignments:** Students will implement transfer learning models during lab sessions, making learning practical, relevant, and interactive.
4. **Continuous Assessment:** Regular classroom quizzes and assessments will provide ongoing feedback, enabling students to identify their strengths and areas for improvement.

**Outside Classroom Learning Experience:**

1. **Question Banks:** These resources will reinforce understanding of both foundational and advanced topics in transfer learning, ensuring comprehensive preparation for exams.
2. **Model Question Papers:** By working on model papers, students will familiarize themselves with typical question formats and challenges, improving exam readiness.
3. **Continuous Assessment Feedback:** Students will receive regular feedback on homework and projects, helping them refine their skills and improve performance through continuous assessment.
4. **Support & Feedback:** Students will have access to one-on-one guidance, peer reviews, and collaboration opportunities to deepen their understanding and enhance their learning outside of class.

**Textbook**

1. “Transfer Learning” by Qiang Yang, Yu Zhang, Wenyuan Dai, Sinno Jialin Pan

**Reference Books:**

1. "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
2. "Transfer Learning in Action" by Yuxi (Hayden) Liu
3. "Transfer Learning for Natural Language Processing" by Paul Azunre
4. "Deep Learning for Computer Vision" by Rajalingappaa Shanmugamani
5. "Hands-On Transfer Learning with Python" by Dipanjan Sarkar and Raghav Bali

**Additional Readings:**

I) **Open Source Projects:**
   a. TensorFlow Hub: GitHub Repository
   b. PyTorch Hub: GitHub Repository

II) **Community Forums and Discussions:**
   a. Reddit: r/MachineLearning
   b. Stack Overflow: Transfer Learning Tag

III) **OSSU Link:**
   Open Source Society University - Data Science Curriculum

# TRANSFER LEARNING LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** | **Course Code** | **L-T-P** | **Credits** |
| **Transfer Learning Lab** | **ENSP358** | 0-0-2 | 1 |
| **Type of Course:** | DSE-4 | | |
| **Pre-requisite(s), if any: NA** | | | |

**Defined Course Outcomes**

| **COs** | **Statement** |
|---|---|
| CO 1 | **Applying** and evaluate basic transfer learning models and techniques to solve image and text classification tasks using pre-trained models. |
| CO 2 | **Implementing** and analyze advanced transfer learning techniques, including relation-based, adversarial, and heterogeneous transfer learning, to improve model performance across different domains. |
| CO 3 | **Developing** and assess multi-task learning models and theoretical aspects of transfer learning, focusing on improving model generalization and performance in multi-domain tasks. |
| CO 4 | **Utilizing** transfer learning for practical applications, including few-shot learning, zero-shot learning, sentiment analysis, and privacy-preserving techniques, to address real-world problems effectively. |

## Lab Experiments

| S.N | Experiment | CO |
|---|---|---|
| 1 | Implement an instance-based transfer learning model using a simple dataset to demonstrate the concepts of noninductive and inductive transfer learning. | CO1 |
| 2 | Develop a feature-based transfer learning model to minimize domain discrepancy and learn universal features. Use a dataset with different domains to showcase the effectiveness of feature augmentation. | CO1 |
| 3 | Create a model-based transfer learning application by sharing model components and applying regularization techniques. Evaluate the performance of the model on a target task with limited data. | CO1 |

|  |  |  |
|---|---|---|
| 4 | Compare the performance of a naive machine learning model with a transfer learning model on the same dataset. Discuss the advantages and challenges of transfer learning in this context. | CO1 |
| 5 | Implement a relation-based transfer learning model using Markov Logic Networks (MLNs). Apply the model to a dataset with relational data and evaluate its performance. | CO2 |
| 6 | Develop a heterogeneous transfer learning application that addresses the challenges of transferring knowledge between different feature spaces or distributions. Demonstrate the application on a real-world dataset. | CO2 |
| 7 | Create an adversarial transfer learning model using GANs. Train the model on a source domain and evaluate its ability to generate or classify data in a target domain. | CO2 |
| 8 | Implement transfer learning in a reinforcement learning context by transferring knowledge from one task to another. Compare the performance of the transfer learning model with a model trained from scratch. | CO2 |
| 9 | Develop a multi-task learning model that simultaneously trains on multiple tasks. Evaluate the performance improvements achieved through shared learning compared to individual task training. | CO3 |
| 10 | Create a transitive transfer learning model using mixed graphs and hidden feature representations. Apply the model to a dataset with multiple related tasks and analyze the results. | CO3 |
| 11 | Implement transfer learning models for various applications, such as computer vision, medical image analysis, and natural language processing. Compare the effectiveness of transfer learning across these different domains | CO4 |

# MINOR PROJECT-III

**Program Name:** **B. Tech (Computer Science and Engineering)**

| | | | |
|---|---|---|---|
| **Course Name: Minor Project-III** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSI352** | **---** | 2 |

**Type of Course:** Project-III

**Pre-requisite(s), if any: NA**

**Duration:**

The minor project will last for three months.

**Project Requirements:**

1. **Problem Identification and Analysis:**
   - Identify a relevant problem in society or industry.
   - Conduct a thorough analysis of the problem, considering various perspectives and implications.
2. **Implementation:**
   - Develop and implement a solution to address the identified problem.
3. **Data Visualization:**
   - Utilize appropriate data visualization techniques to represent the problem, solution, and outcomes effectively.
4. **Presentation of Solutions:**
   - Prepare a comprehensive presentation of the implemented solution, including its development process, outcomes, and impact.
5. **Case Studies:**
   - Conduct case studies related to the problem and solution, analyzing existing examples and drawing relevant insights.

**Guidelines:**

1. **Project Selection:**
   - Choose a societal or industrial problem relevant to the field of computer science and engineering.
   - Ensure the problem is specific and well-defined.

2. **Literature Review:**
    - Conduct a thorough review of existing literature and solutions related to the problem.
    - Identify gaps in existing solutions and potential areas for further investigation.
3. **Implementation:**
    - Develop a detailed plan for implementing the solution.
    - Execute the implementation using appropriate tools, technologies, and methodologies.
4. **Data Visualization:**
    - Collect relevant data and use visualization techniques to represent the problem, solution, and outcomes.
    - Ensure the visualizations are clear, accurate, and effectively communicate the information.
5. **Documentation:**
    - Document the entire process, including problem identification, literature review, implementation, data visualization, and case studies.
    - Use appropriate formats and standards for documentation.
6. **Presentation:**
    - Prepare a presentation summarizing the problem, existing solutions, implementation process, data visualization, and case studies.
    - Ensure the presentation is clear, concise, and well-structured.

**Evaluation Criteria for Minor Project (Out of 100 Marks):**

1. **Problem Identification and Analysis (15 Marks):**
    - Comprehensive identification and analysis of the problem: 15 marks
    - Good identification and analysis of the problem: 12 marks
    - Basic identification and analysis of the problem: 9 marks
    - Poor identification and analysis of the problem: 5 marks
    - No identification and analysis of the problem: 0 marks
2. **Implementation (30 Marks):**
    - Successful and thorough implementation: 30 marks
    - Good implementation: 25 marks
    - Moderate implementation: 20 marks
    - Basic implementation: 15 marks
    - Poor implementation: 10 marks
    - No implementation: 0 marks
3. **Data Visualization (20 Marks):**
    - Effective and clear data visualization: 20 marks

- Good data visualization: 15 marks
- Moderate data visualization: 10 marks
- Basic data visualization: 5 marks
- Poor data visualization: 0 marks

4. **Presentation of Solutions (15 Marks):**
    - Clear, concise, and engaging presentation: 15 marks
    - Clear but less engaging presentation: 12 marks
    - Somewhat clear and engaging presentation: 9 marks
    - Unclear and disengaging presentation: 5 marks
    - No presentation: 0 marks
5. **Case Studies (20 Marks):**
    - Comprehensive and insightful case studies: 20 marks
    - Good case studies: 15 marks
    - Moderate case studies: 10 marks
    - Basic case studies: 5 marks
    - Poor case studies: 0 marks

**Total: 100 Marks**

**Course Outcomes:**

By the end of this course, students will be able to:

1. **Identify and Analyze Problems:**
    - Identify relevant societal or industrial problems and conduct a thorough analysis of these problems.
2. **Implement Solutions:**
    - Develop and implement effective solutions to address identified problems using appropriate tools and technologies.
3. **Visualize Data:**
    - Utilize data visualization techniques to represent problems, solutions, and outcomes clearly and effectively.
4. **Present Solutions:**
    - Prepare and deliver comprehensive presentations summarizing the implementation process, outcomes, and impact of their solutions.
5. **Conduct Case Studies:**
    - Conduct case studies related to the problem and solution, analyzing existing examples and drawing relevant insights.

6. **Literature Review:**
    - Conduct comprehensive literature reviews to identify gaps in existing solutions and potential areas for further investigation.
7. **Documentation:**
    - Document the entire process, including problem identification, literature review, implementation, data visualization, and case studies, using appropriate formats and standards.
8. **Professional Development:**
    - Develop skills in research, analysis, implementation, data visualization, documentation, and presentation, contributing to overall professional growth.

## COMPETITIVE CODING -IV

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name: COMPETITIVE CODING -IV** | **Course Code** | **L-T-P** | **Credits** |
| | | 3-0-0 | 0 |
| **Type of Course:** | AUDIT-4 | | |
| **Contact Hours** | 30 | | |

**Course Outcomes**

| | |
|---|---|
| **CO1** | **Understanding** system design principles and identify functional and non-functional requirements for projects. |
| **CO2** | **Applying** scaling and load balancing techniques and evaluate caching strategies for system optimization. |
| **CO3** | **Designing** efficient database schemas and implement ACID transactions in SQL and NoSQL. |
| **CO4** | **Solving** algorithmic problems using advanced techniques and apply string matching algorithms in coding challenges. |

| Unit Number: 1 | Title: Foundations and Advanced Concepts in System Design | No. of hours: 8 |
|---|---|---|
| **Content:**<br>**Introduction to System Design**<br>• **Principles of System Design**: Basics of system design: modularity, scalability, and maintainability, **Hands-on**: Design a simple e-commerce or social media system blueprint.<br>• **Functional vs. Non-Functional Requirements**: Understand key differences, **Hands-on**: Identify functional and non-functional requirements for a simple project.<br>**Scalability and Load Balancing**<br>• **Scaling Techniques**: **Vertical Scaling**: Adding resources to a single server, **Horizontal Scaling**: Distributing load across multiple servers, **Hands-on**: Set up vertical and horizontal scaling scenarios using cloud tools.<br>• **Load Balancing**: Round-robin, least connections, and IP hashing strategies,<br>**Caching Strategies:**<br>• **Client-Side vs. Server-Side Caching**: Comparison of caching techniques.<br>• **Eviction Policies**: LRU and LFU policies for cache eviction. | | |
| **Unit Number: 2** | **Title: Advanced Database concepts** | **No. of hours: 8** |
| **Content:**<br>**Database Indexing:** different types of indexes (e.g., B-tree, hash, bitmap), how indexes improve query performance, create indexes and their impact on write operations.<br>**Database Transactions:** ACID properties (Atomicity, Consistency, Isolation, Durability), implementing transactions in both SQL and NoSQL databases, scenarios like rollbacks and savepoints**.**<br>**Database Sharding:** partitioning techniques, sharding, sharding strategies for scalability**.**<br>**Data Modeling:** entity-relationship diagrams (ERDs), Normalize data models based on business requirements, Design efficient schemas for different use cases. | | |
| **Unit Number: 3** | **Title: Advanced Concepts** | **No. of hours: 8** |
| **Content:**<br>**Bit Manipulation**: XOR operations, Bitwise AND, OR, NOT, Counting set bits, Power of two, Bit masking | | |

| | | |
|---|---|---|
| **Divide and Conquer:** Matrix exponentiation, Strassen’s algorithm for matrix multiplication, Closest pair of points<br>**Two Pointers:** Fast and slow pointer, Merging sorted arrays, Triplets, pairs with given sum<br>**Sliding Window :** Maximum in a sliding window, Smallest subarray with sum greater than a given value<br>**Union-Find (Disjoint Set Union - DSU) :** Union by rank, Path compression<br>**String Matching:** KMP algorithm, Rabin-Karp, Z-algorithm | | |
| **Unit Number: 4** | **Title: Miscellaneous** | **No. of hours: 6** |
| **Content:**<br>**Hashing: Hash tables, Hash maps and sets, Collision handling, Anagram checks**<br>**Simulation and Design Problems:** LRU cache design, Parking lot simulation, Elevator design, Rate limiter<br>**Concurrency:** Multithreading problems, Deadlock detection<br>**Graphical Algorithms:** Flood fill, Convex hull, Image rendering algorithms | | |

## Lab Experiments

| S. No. | Problem Statement | Mapped CO |
|---|---|---|
| 1 | Design a simple e-commerce system with modularity, scalability, and maintainability. | CO1 |
| 2 | Identify functional and non-functional requirements for a social media platform. | CO1 |
| 3 | Implement vertical scaling on a single server and measure performance. | CO2 |
| 4 | Set up horizontal scaling across multiple servers using a cloud platform. | CO2 |
| 5 | Implement a round-robin load balancer for distributing requests across multiple servers. | CO2 |
| 6 | Compare client-side and server-side caching strategies for a web application. | CO2 |
| 7 | Implement Least Recently Used (LRU) cache eviction policy. | CO2 |
| 8 | Create a B-tree index for a database to optimize search queries. | CO3 |
| 9 | Implement ACID-compliant transactions in an SQL database. | CO3 |

| S. No. | Problem Statement | Mapped CO |
|---|---|---|
| 10 | Implement database sharding for scalability in a NoSQL database. | CO3 |
| 11 | Normalize a database schema to 3NF based on given business requirements. | CO3 |
| 12 | Use bitwise operations to check if a number is a power of two. | CO4 |
| 13 | Implement matrix multiplication using Strassen’s algorithm. | CO4 |
| 14 | Find the closest pair of points in a set using divide and conquer. | CO4 |
| 15 | Merge two sorted arrays using the two-pointer technique. | CO4 |
| 16 | Find the maximum element in a sliding window of size k. | CO4 |
| 17 | Solve the union-find problem using path compression and union by rank. | CO4 |
| 18 | Implement the KMP string matching algorithm to find a pattern in a text. | CO4 |
| 19 | Implement Rabin-Karp algorithm for string matching in a large document. | CO4 |
| 20 | Design a hash table with collision handling using chaining. | CO4 |
| 21 | Implement an anagram checker using hash maps. | CO4 |
| 22 | Simulate an LRU cache system design. | CO2, CO4 |
| 23 | Design a rate limiter using a sliding window algorithm. | CO4 |
| 24 | Implement deadlock detection using multithreading. | CO4 |
| 25 | Solve the flood fill problem using depth-first search (DFS). | CO4 |
| 26 | Implement the convex hull algorithm for a set of 2D points. | CO4 |
| 27 | Design a simple elevator simulation system with multithreading. | CO4 |
| 28 | Implement a parking lot simulation with object-oriented design principles. | CO2, CO4 |
| 29 | Design a system to detect and recover from transaction rollbacks in a database. | CO3 |
| 30 | Optimize an e-commerce website with horizontal scaling and caching strategies. | CO2 |

**Learning Experiences**

**Classroom Learning Experience**

1. Engagement through Lecture PPTs: Utilize well-structured presentations to convey key concepts of system design, database indexing, and algorithmic techniques.

2. Problem-Based Theory Assignments: Assign real-world challenges like load balancing and caching strategies to enhance problem-solving skills.
3. Project-Based Lab Work: Facilitate hands-on lab assignments where students design and implement system blueprints and databases collaboratively.
4. Comprehensive Question Bank: Provide a diverse question bank covering the syllabus to allow systematic practice for exams and coding interviews.
5. Model Question Papers and Assessments: Conduct continuous assessments through quizzes and coding challenges to test understanding of complex concepts.
6. Support & Feedback System: Offer timely feedback on assignments and projects, with access to instructors for additional support and clarification.

**Outside Classroom Learning Experience**

1. Use of ICT Tools & Interactive Boards: Host course materials on Moodle LMS for anytime access, utilizing interactive boards for live demonstrations.
2. Video Lectures for Critical Topics: Provide pre-recorded lectures on advanced topics like ACID transactions and multithreading for flexible learning and review.

**Text Books:**

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press. ISBN: 978-0262033848.
- McDowell, G. L. (2015). *Cracking the Coding Interview: 189 Programming Questions and Solutions* (6th ed.). CareerCup. ISBN: 978-0984782857.
- Skiena, S. S. (2008). *The Algorithm Design Manual* (2nd ed.). Springer. ISBN: 978-1848000698.

**Online References**

- LeetCode (www.leetcode.com)
- HackerRank (www.h System Design Primer (https://github.com/donnemartin/system-design-primer)ackerrank.com)
- Coursera - "Data Structures and Algorithms Specialization" by University of California, San Diego & National Research University Higher School of Economics

## Semester: VII

## (Discipline Specific Elective-III)

## COMPUTATIONAL SERVICES IN THE CLOUD

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name: Computational Services in The Cloud** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENSP401** | 4-0-0 | 4 | 40 |
| **Type of Course:** | DSE-5 | | | |
| **Pre-requisite(s), if any: Basics of Cloud Computing** | | | | |

**Course Perspective.** This course introduces students to the fundamental concepts and applications of cloud computing, exploring the paradigm shift towards cloud-based IT resources and services. It covers various cloud service models (IaaS, PaaS, SaaS), deployment models (public, private, hybrid, community), and key characteristics and challenges of cloud computing. Students will learn about virtualization, microservices, cloud storage, serverless computing, and resource management fundamentals. Additionally, the course includes case studies on cloud market analysis, security, compliance, big data handling, and a comparative study of public clouds. By the end of the course, students will be equipped to understand, implement, and analyze cloud computing technologies and solutions. The course is divided into 4 modules:

a) Foundations of Cloud Computing
b) Advanced Cloud Computing and Virtualization
c) Cloud Security, Privacy, and Compliance
d) Applications of Cloud Computing and Future Trends

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|

|  |  |
|---|---|
| **CO 1** | **Explaining** the core concepts of the cloud computing paradigm: how and why this paradigm shift came about, the characteristics, advantages and challenges brought about by the various models and services in cloud computing. |
| **CO 2** | **Applying** the fundamental concepts in datacenters to understand the tradeoffs in power, efficiency and cost. |
| **CO 3** | **Identifying** resource management fundamentals, i.e. resource abstraction, sharing and sandboxing and outline their role in managing infrastructure in cloud computing. |
| **CO 4** | **Analyzing** various cloud programming models and apply them to solve problems on the cloud. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| **Unit Number: 1** | **Title: Foundations of Cloud Computing** | **No. of hours: 10** |
|---|---|---|
| **Content:**<br>**Introduction to Cloud Computing:**<br>• Definitions and basic concepts<br>• Cloud delivery models (IaaS, PaaS, SaaS)<br>• Cloud deployment models (Public, Private, Hybrid)<br>• Benefits and challenges of cloud computing<br>**Cloud Infrastructure and Architecture:**<br>• Cloud computing services and inter-cloud interoperability<br>• Virtualization and its importance<br>**Security and Ethical Issues:**<br>• Security and privacy concerns<br>• Ethical issues in cloud computing | | |
| **Unit Number: 2** | **Title: Advanced Cloud Computing and Virtualization** | **No. of hours: 12** |

| | | |
|---|---|---|
| **Content:**<br>**Virtualization Technologies:**<br>• Virtual machine monitors<br>• Full virtualization and paravirtualization<br>• Virtualization technology (hardware-based and OS-based)<br>**Resource Management and Scheduling:**<br>• Cloud resource management<br>• Scheduling algorithms and dynamic application scaling<br>• Optimization of network virtualization<br>**Virtualization Security:**<br>• Virtualization security risks | | |
| **Unit Number: 3** | **Title: Cloud Security, Privacy, and Compliance** | **No. of hours: 10** |
| **Content:**<br>**Cloud Security Basics:**<br>• Cloud security risks and challenges<br>• Security mechanisms (encryption, hashing, digital signatures)<br>• Identity and access management<br>**Advanced Security Measures:**<br>• Trusted virtual machine monitors<br>• Cloud security policies and controls<br>• Cloud security threats (traffic eavesdropping, denial of service)<br>**Compliance and Legal Issues:**<br>• Multi-regional compliance<br>• Privacy impact assessment<br>• Case studies on cloud security | | |
| **Unit Number: 4** | **Title: Applications of Cloud Computing and Future Trends** | **No. of hours: 8** |
| **Content:**<br>**Cloud Applications:**<br>• Scientific research and high-performance computing | | |

- Social computing and digital content
- Big data and cloud-based AI/ML applications

**Emerging Trends:**

- Edge computing and fog computing
- Future challenges and opportunities
- Energy use and ecological impact of data centers

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures and Video Sessions**: Explore cloud computing concepts through engaging presentations.
2. **Problem-Based Theory Assignments**: Analyze real-world scenarios to enhance problem-solving skills.
3. **Project-Based Lab Assignments**: Gain practical experience by implementing cloud solutions in labs.
4. **Collaborative Group Work**: Collaborate on case studies related to cloud services.
5. **Continuous Assessment and Feedback**: Receive ongoing assessments and instructor feedback.

**Outside Classroom Learning Experience**

1. **Use of ICT Tools and Moodle LMS**: Access course materials anytime via Moodle.
2. **Engagement with a Question Bank and Model Papers**: Prepare for exams with a question bank and model papers.
3. **Application of Theoretical Knowledge to Practical Scenarios**: Apply cloud concepts to real-world situations.

**Textbooks:**

1. Cloud Computing: Concepts, Technology & Architecture by Thomas
2. “Cloud Computing: Theory and Practice” by Dan C. Marinescu
3. “Cloud Computing: Concepts, Technology & Architecture” by Thomas Erl, Zaigham Mahmood, and Ricardo Puttini

**References**

1. Lizhe Wang, Rajiv Ranjan, Jinjun Chen and Boualem Benatallah, Cloud Computing (1 ed.), CRC Press, 2017. ISBN 978-1351833097.
2. Judith S. Hurwitz and Daniel Kirsch, Cloud Computing For Dummies (2 ed.), Hoboken: John Wiley & Sons, 2020. ISBN 978-1119546658.

# COMPUTATIONAL SERVICES IN THE CLOUD LAB

| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
|---|---|---|---|
| **Course Name: Computational Services in The Cloud Lab** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSP451** | 0-0-2 | 1 |
| **Type of Course:** | DSE-7 | | |
| **Pre-requisite(s), if any: Basics of Cloud Computing** | | | |

**Defined Course Outcomes**

| **COs** | **Statement** |
|---|---|
| CO 1 | **Implementing** advanced resource management and scheduling systems in cloud environments to optimize the efficiency and performance of virtualized resources. |
| CO 2 | **Developing** comprehensive security and compliance frameworks for cloud infrastructures, addressing various security threats and ensuring regulatory compliance. |
| CO 3 | **Enhancing** data privacy and compliance strategies for multi-regional cloud deployments, ensuring adherence to global and regional data protection regulations. |
| CO 4 | **Leveraging** cloud computing resources for high-performance scientific research, enabling scalable and efficient data processing, storage, and analysis. |

**List of Programs**

| **S.N** | **Project Detail** | **COs** |
|---|---|---|
| 1 | Set up a virtual machine (VM) on a cloud platform (e.g., AWS, Azure, Google Cloud). Explore different VM configurations and understand the basics of IaaS. | CO1 |
| 2 | Deploy a simple web application using a PaaS provider (e.g., Heroku, Google App Engine). Demonstrate the deployment process and manage application scaling | CO1 |
| 3 | Implement a cloud storage solution using a SaaS provider (e.g., Dropbox, Google Drive). Upload, share, and manage files to understand cloud storage benefits and challenges. | CO1 |

| | | |
|---|---|---|
| 4 | Investigate the security and privacy settings of a cloud service provider. Configure security groups and access controls to secure your cloud resources. | CO1 |
| 5 | Install and configure a virtual machine monitor (VMM) like VMware or VirtualBox. Compare full virtualization and paravirtualization techniques. | CO2 |
| 6 | Install and configure a virtual machine monitor (VMM) like VMware or VirtualBox. Compare full virtualization and paravirtualization techniques. | CO2 |
| 7 | Optimize network virtualization by setting up and managing virtual networks in a cloud environment. Analyze the performance benefits of network virtualization. | CO2 |
| 8 | Implement encryption and hashing mechanisms to secure data stored in the cloud. Demonstrate how these mechanisms protect data integrity and confidentiality. | CO3 |
| 9 | Set up identity and access management (IAM) in a cloud environment. Configure single sign-on (SSO) for multiple cloud services to streamline user access. | CO3 |
| 10 | Develop a cloud-based AI application using a cloud provider's machine learning services (e.g., AWS SageMaker, Google AI Platform). Train and deploy a machine learning model in the cloud. | CO4 |
| 11 | Implement a cloud-based big data solution using Hadoop or Spark on a cloud platform. Process and analyze a large dataset to understand the benefits of cloud-based big data processing. | CO4 |
| 12 | Explore edge computing by deploying a cloud application that interacts with IoT devices. Demonstrate how edge computing can reduce latency and improve performance. | CO4 |

## MICROSOFT AZURE CLOUD FUNDAMENTALS

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name: Microsoft Azure Cloud Fundamentals** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENSP403** | 4-0-0 | 4 | 40 |

| **Type of Course:** | DSE-5 |
|---|---|
| **Pre-requisite(s), if any: Basics of Cloud Computing** | |

**Course Perspective.** This course introduces students to the fundamental concepts of cloud computing with a focus on Microsoft Azure. It aims to bridge the gap between theoretical cloud principles and practical Azure applications, emphasizing the relevance of cloud services in modern engineering and technology. Students will explore core topics such as cloud computing models, Azure architecture, compute and networking services, storage services, and cost management. The course is divided into four modules:

1. Introduction to Cloud Computing and Azure Fundamentals
2. Introduction to Microsoft Azure
3. Azure Storage Services and Identity Management
4. Azure Cost Management, Governance, and Monitoring

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Identifying** the core concepts of cloud computing and Microsoft Azure, including deployment models and service models. |
| **CO 2** | **Understanding** the benefits of cloud services, such as high availability, scalability, and security. |
| **CO 3** | **Understanding** Azure architecture components and compute/networking services, analyzing their functionality and use cases. |
| **CO 4** | **Determining** the appropriate Azure storage services for different performance requirements and analyze identity management and security features for access control. |
| **CO 5** | **Analyzing** cost management strategies in Azure, analyze governance and compliance tools, and determine effective methods for managing and deploying Azure resources. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| **Unit Number: 1** | **Title: Introduction to Cloud Computing and Azure Fundamentals** | **No. of hours: 10** |
|---|---|---|

| | | |
|---|---|---|
| **Content Summary:**<br>**Cloud Computing Basics:** What is cloud computing, Delivery models, deployment models, defining attributes, resources, and organization of the infrastructure. Network-Centric Computing and Network-Centric Content. Cloud computing delivery models and services, Applications of cloud, Ethical Issues in Cloud Computing, Major Challenges Faced by Cloud Computing | | |
| **Unit Number: 2** | **Title Introduction to Microsoft Azure** | **No. of hours: 12** |
| **Content Summary:**<br>**Azure Architecture Components**: Azure regions, availability zones, datacentres, Azure resources, resource groups, subscriptions, Management groups hierarchy,<br>**Azure Compute and Networking Services:** Compute types comparison: Container instances, VMs, Functions, Virtual machine options: VMs, VM Scale Sets, availability sets, Azure Virtual Desktop, Application hosting options, Virtual networking: Azure Virtual Networks, subnets, peering, DNS, VPN Gateway, ExpressRoute, Public and private endpoints | | |
| **Unit Number: 3** | **Title: Azure Storage Services and Identity Management** | **No. of hours: 10** |
| **Content Summary:**<br>**Azure Storage Services**: Create and manage virtual machines using Azure. Different VM sizes and types based on performance requirements. VM scaling and load balancing for optimizing application performance. Azure storage services: Blob Storage, Table Storage, File Storage, and Disk Storage. | | |
| **Unit Number: 4** | **Title: Azure Cost Management, Governance, and Monitoring** | **No. of hours: 8** |
| **Content Summary:**<br>**Cost Management in Azure:** Factors affecting costs, Pricing and TCO calculators, Azure Cost Management and Billing tool, Tagging usage.<br>**Governance and Compliance:** Azure Blueprints, Azure Policy, Resource locks, Service Trust Portal<br>**Monitoring Tools in Azure:** Azure Advisor, Azure Service Health, Azure Monitor: Log Analytics, alerts, Application Insights | | |

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures and Video Sessions**: Learn Microsoft Azure concepts through engaging presentations.

2. **Problem-Based Theory Assignments**: Analyze real-world Azure scenarios to develop practical skills.
3. **Project-Based Lab Assignments**: Implement Azure solutions in hands-on labs for experiential learning.
4. **Collaborative Group Work**: Work together on case studies focused on Azure applications.
5. **Continuous Assessment and Feedback**: Receive regular assessments and instructor feedback to monitor progress.

**Outside Classroom Learning Experience**

1. **Use of ICT Tools and Moodle LMS**: Access course materials anytime via Moodle for flexible learning.
2. **Engagement with a Question Bank and Model Papers**: Prepare for exams using a question bank and model papers.
3. **Application of Theoretical Knowledge to Practical Scenarios**: Apply Azure concepts to real-world scenarios for better understanding.

**Text Book**:

a) "Cloud Computing: Theory and Practice" by Dan C. Marinescu

b) "Exam Ref AZ-900 Microsoft Azure Fundamentals" by Jim Cheshire.

**References**

a) "Exam Ref AZ-900 Microsoft Azure Fundamentals" by Jim Cheshire.
b) "Microsoft Azure Essentials: Fundamentals of Azure" by Michael Collier and Robin Shahan.
c) "Azure for Architects: Implementing cloud design, DevOps, IoT, and serverless solutions on your public cloud" by Ritesh Modi.
d) "Azure Security Center: Protecting your cloud workloads" by Yuri Diogenes, Tom Shinder, and Debra Shinder.
e) "Azure Cost Management and Billing" by Sjoukje Zaal

**Additional Readings:**

**Online Learning Resources:**

I) **Microsoft Learn:**

Microsoft's official learning platform offers a wide range of Azure courses, modules, and learning paths for beginners to advanced users. Explore topics such as cloud computing basics, Azure architecture components, compute and networking services, storage services, identity management, cost management, governance, and monitoring. Microsoft Learn

**R 1. Coursera:**

Coursera provides Azure courses offered by top universities and organizations. Topics include cloud computing basics, Azure architecture, services, security, governance, and more. Coursera Azure Courses

2. **Open-Source Society University (OSSU):**

OSSU offers a structured, open-source curriculum for self-learning various topics, including cloud computing and Azure fundamentals. You can follow their curriculum to gain a comprehensive understanding of Azure concepts and services. OSSU Cloud Computing Curriculum

# MICROSOFT AZURE CLOUD FUNDAMENTALS LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name: Microsoft Azure Cloud Fundamentals Lab** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSP453** | 0-0-2 | 1 |
| **Type of Course:** | DSE-7 | | |
| **Pre-requisite(s), if any: Basics of Cloud Computing** | | | |

**Defined Course Outcomes**

| **COs** | **Statement** |
|---|---|
| CO 1 | **Deploying** and manage scalable web applications using Azure architecture components, ensuring high availability, fault tolerance, and optimal performance. |
| CO 2 | **Developing** and optimize Azure storage solutions for data-intensive applications, focusing on efficient data storage, retrieval, and performance. |
| CO 3 | **Establishing** secure and compliant environments in Azure, ensuring governance, cost management, and continuous monitoring for mission-critical applications. |
| CO 4 | **Migrating** on-premise applications to Azure, ensuring minimal downtime and optimized performance through effective planning, resource management, and monitoring. |

**Lab Experiments**

| **S.N** | **Project Detail** | **COs** |
|---|---|---|
| 1 | Set up a cloud environment using a chosen cloud provider (e.g., AWS, Azure, Google Cloud). Explore and document the different delivery models (IaaS, PaaS, SaaS) and deployment models (Public, Private, Hybrid). | CO1 |
| 2 | Implement a network-centric application using cloud services. Demonstrate how network-centric content can be delivered and managed in a cloud environment. | CO1 |
| 3 | Explore a cloud-based application (e.g., Google Docs, Office 365). Analyze its benefits and the ethical issues it presents in terms of data privacy and security. | CO1 |

|  |  |  |
|---|---|---|
| 4 | Explore Azure regions, availability zones, and datacenters. Create and manage Azure resources and resource groups, and understand the subscription and management groups hierarchy. | CO2 |
| 5 | Compare different Azure compute types (Container instances, VMs, Functions). Create and manage VMs, VM Scale Sets, and availability sets. Explore Azure Virtual Desktop and application hosting options. | CO2 |
| 6 | Set up a virtual network in Azure. Configure subnets, peering, DNS, VPN Gateway, and ExpressRoute. Explore the use of public and private endpoints in Azure networking. | CO2 |
| 7 | Host a web application on Azure using different hosting options. Compare the performance and cost implications of using VMs, Azure App Service, and Azure Functions. | CO2 |
| 8 | Create and manage virtual machines in Azure. Explore different VM sizes and types based on performance requirements. Implement VM scaling and load balancing to optimize application performance. | CO3 |
| 9 | Set up Azure Blob Storage and upload/download data. Explore Table Storage and File Storage services. Implement Disk Storage and understand its use cases. | CO3 |
| 10 | Use the Azure Pricing Calculator and TCO Calculator to estimate the costs of running a sample application on Azure. Explore the Azure Cost Management and Billing tool to monitor and control costs. | CO4 |
| 11 | Set up monitoring for an Azure application using Azure Monitor. Configure Log Analytics, set up alerts, and use Application Insights to monitor application performance and health | CO4 |
| 12 | Use Azure Advisor and Azure Service Health to optimize and maintain the health of Azure resources. Implement recommendations provided by Azure Advisor and monitor service issues using Azure Service Health. | CO4 |

# STORAGE AND DATABASES ON CLOUD

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **Storages and Databases on Cloud** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENSP405** | 4-0-0 | 4 | 40 |
| **Type of Course:** | DSE-5 | | | |
| **Pre-requisite(s), if any: Basics of Cloud Databases** | | | | |

**Course Perspective.** The course covers the basics of cloud computing and introduces various cloud storage and database types. It discusses migration techniques, security, and performance considerations for cloud databases. The AWS cloud storage unit focuses on Amazon S3, EC2 Instance Storage, and more. It also help student analyzing case studies of companies like Netflix and Spotify using cloud storage and databases. The course is divided into 4 modules:

a) Introduction to Storage on cloud
b) Data Integration, Migration, Security and performance on cloud NET Framework Fundamentals
c) Cloud-Hosted Data Storage Systems
d) Case Study

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** cloud storage and database fundamentals, including security best practices. |
| **CO 2** | **Applying** indexing, caching, and query optimization for performance in cloud storage and databases. |
| **CO 3** | **Analyzing** requirements to select suitable cloud storage and database solutions. |
| **CO 4** | **Differentiating** between types of cloud storage and database services. |
| **CO 5** | **Articulating** best practices for designing scalable, reliable, and secure cloud storage and databases. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number:1** | **Title: Introduction to Storage on cloud** | **No. of hours: 8** |
| **Content:**<br>Introduction to Cloud Computing, Overview of cloud databases and cloud storages, types of cloud storages (Object, block and file), different types of cloud database management systems, Gartner Magic Quadrant for Cloud Database Management Systems, Advantages of Working with Cloud Databases, Considerations for Cloud Databases, Top Cloud Database, Factors that help in choosing the right cloud database, Challenges involved in using cloud storages and databases. | | |
| **Unit Number:2** | **Title: Data Integration, Migration, Security and performance on cloud NET FrameworkFundamentals** | **No. of hours: 10** |
| **Content:**<br>Techniques, tools, methods, and considerations for migrating from on-premise databases to cloud databases; backup, recovery, and disaster planning, including automated backups, point-in-time recovery, and replication; performance optimization and monitoring, including query optimization, indexing, caching, and monitoring tools; scalability and high availability, including load balancing, replication, sharding, and auto-scaling; cloud data warehousing. | | |
| **Unit Number:3** | **Title: Cloud-Hosted Data Storage Systems** | **No. of hours: 10** |
| **Content:**<br>**Introduction,**<br>Introduction to AWS cloud storage, AWS management console, AWS Storage Services, Uploading files and images, Creating a web server, Overview of Amazon S3, Storage Classes, EC2 Instance Storage, network file system Amazon Elastic Block Store, Amazon Elastic file system, Amazon Cloud Front. Brief introduction to Google Cloud Storage, and Azure Blob Storage. | | |
| **Unit Number:4** | **Title: Case Study** | **No. of hours: 12** |
| **Content:**<br>Case Studies and Real-world Examples of Netflix , Airbnb, Pinterest, spotify, coca-cola etc. Analyzing real- | | |

world use cases of organizations using cloud storage and databases, discussing architecture decisions, challenges, and lessons learned.

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures**: Learn cloud storage and database concepts through engaging presentations.
2. **Problem-Based Assignments**: Analyze real-world scenarios to improve understanding of cloud solutions.
3. **Project Labs**: Implement cloud database solutions in hands-on labs.
4. **Collaborative Work**: Team up on case studies related to cloud storage management.
5. **Continuous Feedback**: Receive regular assessments and instructor feedback.

**Outside Classroom Learning Experience**

1. **Moodle Access**: Access course materials anytime via Moodle.
2. **Question Bank**: Use a question bank and model papers for exam preparation.
3. **Real-World Applications**: Apply cloud concepts to practical scenarios.

**References**

1. Cloud Computing: Concepts, Technology & Architecture by Thomas Erl, Ricardo Puttini, and Zaigham Mahmood
2. Designing Data-Intensive Applications by Martin Kleppmann
3. Cloud Architecture Patterns: Using Microsoft Azure" by Bill Wilder

**Additional Readings:**

**Online Learning Resources for Storage and Databases on Cloud**

I) **Microsoft Learn: Introduction to Azure Storage**
   a. **Description:** Comprehensive learning path covering Azure Storage services, including Blob, File, and Disk Storage.
   b. **Link:** Microsoft Learn - Introduction to Azure Storage

II) **AWS Training and Certification: Storage Learning Path**
   a. **Description:** AWS offers a detailed learning path for storage services, including Amazon S3, EBS, and more.
   b. **Link:** AWS Training - Storage Learning Path

III) **Google Cloud Training: Storage and Databases**
   a. **Description:** Google Cloud offers courses on Cloud Storage, SQL, and NoSQL database services.

b. **Link:** Google Cloud Training - Storage and Databases

## STORAGE AND DATABASES ON CLOUD LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** **Storages and Databases on Cloud Lab** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSP455** | 0-0-2 | 1 |
| **Type of Course:** | DSE-7 | | |
| **Pre-requisite(s), if any: Basics of Cloud Databases** | | | |

**Defined Course Outcomes**

| **COs** | **Statements** |
|---|---|
| **CO 1** | **Implementing** database migration, backup, recovery, and performance optimization strategies for transitioning on-premise databases to AWS cloud. |
| **CO 2** | **Developing** cloud storage solutions for large-scale file management and optimize performance using AWS storage services and content delivery networks. |
| **CO 3** | **Designing** and manage cloud data warehousing solutions, including ETL processes, performance monitoring, and scalability configurations. |
| **CO 4** | **Analyzing** and apply best practices from real-world cloud storage use cases to enhance the scalability, reliability, and performance of cloud-based applications. |

## Lab Experiments

| Project No. | Project Detail | Mapped CO/COs |
|---|---|---|
| 1 | Explore different types of cloud storages (Object, Block, File). Set up and compare examples of each type using a cloud provider (e.g., AWS S3 for object storage, EBS for block storage, EFS for file storage). | CO1 |
| 2 | Research and analyze the Gartner Magic Quadrant for Cloud Database Management Systems. Create a report summarizing the top cloud database providers and their key features. | CO1 |
| 3 | Implement a migration process from an on-premise database to a cloud database using a migration tool (e.g., AWS Database Migration Service, Google Cloud Database Migration Service). Document the steps and considerations involved. | CO2 |
| 4 | Develop a cloud storage solution for a media sharing platform using AWS storage services to handle large-scale file uploads and downloads. | CO2 |
| 5 | Configure Amazon CloudFront for content delivery. Upload and distribute content using CloudFront and analyze the performance benefits. Briefly explore and set up storage using Google Cloud Storage and Azure Blob Storage. | CO3 |
| 6 | Create a cloud data warehouse for an e-commerce company to store and analyze sales data using AWS Redshift. | CO3 |
| 7 | Develop a cloud storage and content delivery network (CDN) solution for a video streaming service to ensure | CO4 |
| 8 | Conduct a comprehensive analysis of how major companies like Netflix, Airbnb, and Spotify use cloud storage and databases to enhance their operations. | CO4 |

## APPLICATION DEVELOPMENT AND DEVOPS ON CLOUD

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name: APPLICATION DEVELOPMENT AND DEVOPS ON CLOUD** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENSP407** | 4-0-0 | 4 | 40 |
| **Type of Course:** | DSE-5 | | | |
| **Pre-requisite(s), if any: Basics of DEVOPS Technology** | | | | |

**Course Perspective.** The syllabus aims to equip students with practical skills and theoretical knowledge to design, develop, and deploy applications in cloud environments while implementing DevOps practices to enhance software development, delivery, and operations on the cloud. It prepares them for a career in the dynamic and rapidly growing field of cloud computing and DevOps, where demand for skilled professionals is high due to the increasing adoption of cloud technologies in various industries. The course is divided into 4 modules:

a) Introduction to Cloud Computing
b) Cloud-Based Application Development
c) DevOps Practices in Cloud
d) Cloud-Based DevOps Tools and Best Practices

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the fundamental concepts of cloud computing and the various service and deployment models. |
| **CO 2** | **Developing** cloud-native applications using containerization and microservices architecture. |
| **CO 3** | **Implementing** DevOps practices in cloud environments, including CI/CD pipelines and Infrastructure as Code. |
| **CO 4** | **Utilizing** cloud-based DevOps tools for version control, collaboration, testing, and performance optimization. |

| | |
|---|---|
| **CO 5** | **Analyzing** best practices for application security, cost management, and high availability in the cloud. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| **Unit Number: 1** | **Title: The problem of Delivering Software** | **No. of hours: 10** |
|---|---|---|
| **Content Summary:** Introduction to DevOps: Principles, Practices, Common Release antipatterns, benefits. Configuration Management: using version control, managing dependencies, managing software configuration, managing tools | | |
| **Unit Number: 2** | **Title: Continuous Integration and Testing Strategy** | **No. of hours: 10** |
| **Content Summary:** Introduction to contigous integration. Implementing contiguous integration, Essential practices, distributed version control system. Testing Strategy: Introduction of testing, Types of tests, real-life situation and strategies, and managing defect backlogs | | |
| **Unit Number: 3** | **Title: The deployment Pipeline** | **No. of hours: 10** |
| **Content Summary:** Anatomy of the Deployment Pipeline: Introduction of deployment pipeline, deployment pipeline practices, the automated acceptance test gate, test strategy, prepare to release, implement a deployment pipeline. Build and deployment scripting, the commit stage: principles and practices, Automated Acceptance testing, Testing Non functional Requirements, deploying and releasing application. | | |
| **Unit Number: 4** | **Title: The delivering Ecosystem** | **No. of hours: 10** |
| **Content Summary:** Managing infrastructure and Environments: understanding the needs of the operation team . Managing server provisioning and configuration, managing the configuration of middleware, managing infrastructure services, virtualization, cloud architecture, monitoring infrastructure and application, managing data: Database scripting, data management and deployment pipeline. | | |

|  |
|---|
| Managing components and dependencies: Introduction, keeping your application releasable, dependencies, components, managing dependency graph.<br>Managing Continuous delivery: introduction, maturity model, project lifecycle, risk management process. |

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures**: Learn application development and DevOps concepts through engaging presentations.
2. **Problem-Based Assignments**: Analyze real-world scenarios to enhance development and deployment skills.
3. **Project Labs**: Implement cloud-based applications and DevOps practices in hands-on labs.
4. **Collaborative Work**: Work in teams on case studies related to application deployment.
5. **Continuous Feedback**: Receive regular assessments and instructor feedback to monitor progress.

**Outside Classroom Learning Experience**

1. **Moodle Access**: Access course materials anytime via Moodle for flexible learning.
2. **Question Bank**: Utilize a question bank and model papers for effective exam preparation.
3. **Real-World Applications**: Apply development and DevOps concepts to practical scenarios.

**Textbooks**:

- Jez Humble and David Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Pearson Education, Inc., 2011.

**References**

- Jez Humble and David Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Pearson Education, Inc., 2011.
- Thomas Erl, Ricardo Puttini, and Zaigham Mahmood, Cloud Computing: Concepts, Technology & Architecture, Prentice Hall, 2013.
- Arun Eapen, Docker on Amazon Web Services: Build, deploy, and manage your container applications at scale on AWS, Packt Publishing, 2017.
- Sam Newman, Building Microservices: Designing Fine-Grained Systems, O'Reilly Media, Inc., 2015.
- Mark Richards and Neal Ford, Fundamentals of Software Architecture: An Engineering Approach, O'Reilly Media, Inc., 2020.

**Additional Readings:**

**Online Learning Resources for Application Development and DevOps on Cloud**

I) **Microsoft Learn: Azure DevOps and Development**

   a. **Description:** Comprehensive learning paths and modules on Azure DevOps, including CI/CD, IaC, and cloud-based application development.

   b. **Link:** Microsoft Learn - Azure DevOps and Development

II) **AWS Training and Certification: DevOps on AWS**

   a. **Description:** Detailed courses and certifications for learning DevOps practices and application development on AWS, covering tools like AWS CodePipeline, CodeBuild, and more.

   b. **Link:** AWS Training - DevOps on AWS

III) **Google Cloud Training: Application Development**

   a. **Description:** Google Cloud provides courses on developing applications using Google Cloud services, including Kubernetes, App Engine, and Cloud Functions.

   b. **Link:** Google Cloud Training - Application Development

# APPLICATION DEVELOPMENT AND DEVOPS ON CLOUD LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** **APPLICATION DEVELOPMENT AND DEVOPS ON CLOUD LAB** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSP457** | 0-0-2 | 1 |
| **Type of Course:** | DSE-7 | | |
| **Pre-requisite(s), if any: Basics of DEVOPS Technology** | | | |

**Defined Course Outcomes**

| COs | Course Outcomes (COs) |
|---|---|
| **CO 1** | **Implementing** continuous integration (CI) pipelines to automate the build, test, and integration processes, ensuring smooth and efficient integration of new code changes. |
| **CO 2** | **Developing** and implement automated deployment pipelines for microservices and mobile applications, ensuring reliable and efficient deployment processes. |
| **CO 3** | **Integrating** comprehensive testing strategies, including acceptance and non-functional requirements testing, into CI/CD pipelines to ensure high code quality and performance standards. |
| **CO 4** | **Managing** and monitor cloud-based application infrastructure using automation tools, ensuring efficient provisioning, configuration, and continuous monitoring. |

## Lab Experiments

| S.N | Experiment | COs |
|---|---|---|
| 1 | Set up a version control system (e.g., Git) for a sample software project. Demonstrate how to manage code versions, branches, and merges. | CO1 |
| 2 | Implement configuration management using a tool such as Ansible or Chef. Create scripts to manage software configurations and dependencies for a sample application. | CO1 |
| 3 | Explore common release antipatterns in software delivery. Analyze a real-world case study and propose solutions to mitigate these antipatterns using DevOps principles. | CO1 |

|  |  |  |
|---|---|---|
| 4 | Implement continuous integration for a sample project using a CI tool (e.g., Jenkins, Travis CI). Configure the tool to automatically build and test the project whenever code changes are committed. | CO2 |
| 5 | Set up a distributed version control system (e.g., Git) for a collaborative project. Demonstrate branching, merging, and managing code changes in a distributed environment. | CO2 |
| 6 | Implement a deployment pipeline for a sample application. Automate the build, test, and deployment stages using a CI/CD tool like Jenkins or GitLab CI. | CO3 |
| 7 | Write and execute build and deployment scripts for a sample project. Use scripting languages like Bash or PowerShell to automate the process. | CO3 |
| 8 | Set up and configure infrastructure for a sample application using Infrastructure as Code (IaC) tools like Terraform or CloudFormation. Demonstrate server provisioning, middleware configuration, and monitoring. | CO4 |
| 9 | Implement continuous delivery for a sample project. Develop a maturity model, define the project lifecycle, and establish a risk management process to ensure smooth delivery and deployment. | CO4 |

# DISCIPLINE SPECIFIC ELECTIVE - IV (FULL STACK DEVELOPMENT)

## MOBILE APPLICATION DEVELOPMENT USING IOS

| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
|---|---|---|---|---|
| **Course Name: Mobile Application Development using iOS** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENSP409** | 4-0-0 | 4 | 40 |
| **Type of Course:** | DSE-6 | | | |
| **Pre-requisite(s), if any: Basics of Android** | | | | |

**Course Perspective.** The objective of the course is to provide skills to develop applications for OS X and iOS. It includes an introduction to the development framework Xcode. Objective-C is used as a programming language to develop applications. Objective-C is the superset of the C programming language and provides object-oriented capabilities and a dynamic runtime. Objective-C inherits the syntax, primitive types, and flow control statements of C and adds syntax for defining classes and methods. The course is divided into 4 modules:

1. Introduction to IDE and SDK of iOS App Development
2. Swift Programming
3. Encapsulating Data
4. Developing iOS Applications

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the fundamental concepts of variables, constants, and basic data types in SWIFT. |
| **CO 2** | **Analyzing** the use of control flow statements such as for, if, and switch in various programming scenarios. |
| **CO 3** | **Applying** object-oriented concepts in SWIFT, including the use of classes, structures, and |

| | |
|---|---|
| | protocols. |
| **CO 4** | **Creating** functions, closures, and extensions to enhance code modularity and reuse. |
| **CO5** | **Evaluating** error handling techniques and type checking mechanisms to develop robust SWIFT applications |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| **Unit Number: 1** | **Title: Introduction to SWIFT Language** | **No. of hours: 10** |
|---|---|---|
| **Content Summary:** Variables & Constants, Introduction to functions (methods), Arrays, Dictionaries, Data, Date and other basic data types, Enums, structures, closuresFor, If, switch statement, Object oriented concepts with SWIFT, Type check, AnyObject, Any Protocols, Extensions, Error handling, Working with classes | | |
| **Unit Number: 2** | **Title: Working with Xcode** | **No. of hours: 8** |
| **Content Summary:** Introduction to XCODE, COCOA touch framework, iOS application architecture, Application lifecycle | | |
| **Unit Number: 3** | **Title: Introduction to view controllers and Views** | **No. of hours: 12** |
| **Content Summary:** View Controllers, view, view lifecycle, Basic Controls – Label, Buttons, Text field, image View, Table view with default cells and customized cells, Collection view with default cells and customized cells, Picker view, Date picker, scroll view, navigation and Tab bar controller, Understanding Interface builder, XIB files, Creating outlets and Actions, Handling touch and gesture events, Segment and Page control, switch view, UIAlertView | | |
| **Unit Number: 4** | **Title: Integrating with Database** | **No. of hours: 10** |
| **Content Summary:** Introduction to data storage methods in iOS, Using Core Data, SQLite database, User Defaults, Property List | | |

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures**: Explore iOS app development concepts through engaging presentations.
2. **Problem-Based Assignments**: Analyze real-world scenarios to improve mobile development skills.
3. **Project Labs**: Build and test iOS applications in hands-on lab sessions.
4. **Collaborative Work**: Work in teams on case studies related to mobile app development.
5. **Continuous Feedback**: Receive regular assessments and instructor feedback to track progress.

**Outside Classroom Learning Experience**

1. **Moodle Access**: Access course materials anytime via Moodle for convenient learning.
2. **Question Bank**: Use a question bank and model papers for exam preparation.
3. **Real-World Applications**: Apply iOS development concepts to practical mobile app scenarios.

**References**

1. iOS 14 Programming for Beginners: Kickstart your iOS app development journey with the Swift programming language and Xcode 12, 6th Edition, Ahmad Sahar and Craig Clayton.
2. Mastering iOS 14 Programming: Build professional-grade iOS applications with Swift 5 and Xcode 12.

**Additional Readings:**

**Online Learning Resources for Mobile Application Development Using iOS**

- **Apple Developer Documentation**
  - **Description:** Comprehensive documentation and tutorials for iOS app development using Swift and Xcode.
  - **Link:** Apple Developer Documentation
- **Ray Wenderlich: iOS and Swift Tutorials**
  - **Description:** A collection of high-quality tutorials and courses on iOS app development, covering Swift, Xcode, and various iOS frameworks.
  - **Link:** Ray Wenderlich iOS Tutorials
- **GitHub: iOS Development Resources**
  - **Description:** A curated list of open-source projects, libraries, and resources for learning and improving iOS development skills.
  - **Link:** GitHub - iOS Development Resources

# MOBILE APPLICATION DEVELOPMENT USING IOS LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** **Mobile Application Development using iOS Lab** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSP459** | 0-0-2 | 1 |
| **Type of Course:** | DSE-8 | | |
| **Pre-requisite(s), if any: Basics of Android** | | | |

**Defined Course Outcomes**

| COs | **Statements** |
|---|---|
| **CO 1** | **Understanding** and apply fundamental concepts of iOS development using Xcode and the Cocoa Touch framework to build robust and user-friendly applications. |
| **CO 2** | **Developing** interactive and dynamic user interfaces in iOS applications using view controllers, views, and gesture recognizers. |
| **CO 3** | **Creating** and manage user interfaces and view controllers in iOS applications using Xcode, demonstrating proficiency in Interface Builder and UIKit components. |
| **CO 4** | **Developing** interactive and dynamic user interfaces in iOS applications using view controllers, views, and gesture recognizers. |

## Lab Experiments

| **S.N** | **Experiment** | **COs** |
|---|---|---|
| 1 | Set up the iOS development environment by installing Xcode. Create a simple "Hello, World!" iOS application to familiarize with the Xcode IDE and Swift programming basics. | CO1 |
| 2 | Develop a basic iOS application that demonstrates the use of Swift syntax, variables, data types, and control flow. Create a simple calculator app to perform basic arithmetic operations. | CO1 |

| | | |
|---|---|---|
| 3 | Use Xcode and Interface Builder to design a user interface for an iOS app. Create a simple user interface with labels, buttons, and text fields, and handle user interactions. | CO1 |
| 4 | Implement a simple iOS app to demonstrate the app lifecycle and navigation between view controllers. Create a multi-screen app that navigates between different views using navigation controllers. | CO1 |
| 5 | Design a responsive user interface using Auto Layout and the constraint system. Create an iOS app with a login screen that adjusts to different screen sizes and orientations. | CO2 |
| 6 | Implement navigation between different views using storyboards and segues. Create a multi-screen app with a main menu and detailed views for each menu item. | CO2 |
| 7 | Implement gesture recognition and touch event handling in an iOS app. Create an app that responds to tap, swipe, and pinch gestures to perform different actions. | CO2 |
| 8 | Implement data persistence using Core Data. Create an iOS app that allows users to add, edit, and delete notes, and save them to a local database. | CO3 |
| 9 | Use User Defaults and the file system to store and retrieve user preferences and data. Create an app that saves user settings and displays them when the app is reopened. | CO3 |
| 10 | Implement offline data storage and synchronization. Create an iOS app that allows users to add data while offline and syncs with a remote server when the device is back online. | CO3 |
| 11 | Implement advanced UI components and animations in an iOS app. Create a visually appealing app with custom views, animations, and transitions between screens. | CO4 |
| 12 | Access and use iOS sensors and hardware features. Create an app that uses the camera to take photos, and the GPS to display the user's current location on a map. | CO4 |
| 13 | Debug and test an iOS app using Xcode's debugging tools. Implement unit tests and UI tests to ensure the app functions correctly under different scenarios. | CO4 |

# DEVOPS & AUTOMATION

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **DevOps & Automation** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENSP411** | 4-0-0 | 4 | 40 |
| **Type of Course:** | DSE-6 | | | |
| **Pre-requisite(s), if any: Basics of DEVEOPS** | | | | |

**Course Perspective.** Throughout the subject, students will engage in hands-on exercises and projects to gain practical experience with various DevOps tools and practices. By the end of the course, students will be well-equipped to embrace the DevOps culture and apply automation techniques to enhance software development, delivery, and operations processes. The course is divided into 4 modules:

a) Introduction to DevOps
b) Version Control and CI/CD
c) Containerization and Orchestration
d) Configuration Management and Monitoring

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** the principles and benefits of DevOps, and its role in enhancing collaboration and efficiency between development and operations teams. |
| **CO 2** | **Acquiring** hands-on experience with popular DevOps tools such as Git, Jenkins, Docker, Kubernetes, and Ansible for implementing continuous integration, continuous delivery, and automated deployment processes. |
| **CO 3** | **Demonstrating** proficiency in containerization and orchestration techniques using Docker and Kubernetes for efficient and scalable application deployment and management. |
| **CO 4** | **Implementing** configuration management and Infrastructure as Code (IaC) using Ansible and Terraform to automate the provisioning and management of infrastructure resources. |
| **CO 5** | **Developing** skills in monitoring, logging, and security practices in the context of DevOps, ensuring application performance, resilience, and adherence to security best practices. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| **Unit Number: 1** | **Title: Introduction to DevOps** | **No. of hours: 12** |
|---|---|---|
| **Content Summary:**<br><br>**DevOps Principles and Culture**: Understand the core principles of DevOps and its cultural impact. Collaboration, automation, continuous integration, continuous delivery, and continuous deployment.<br>**DevOps Toolchain**: Overview of tools and technologies used in DevOps practices. Introduction to popular DevOps tools like Git, Jenkins, Docker, Kubernetes, and Ansible.<br>**Version Control with Git**: Branching, merging, and collaborative development using Git. **Continuous Integration (CI):** Setting up CI pipelines with Jenkins for automated building and testing.<br>**Continuous Delivery and Deployment:** Implementing CD pipelines for deploying. | | |
| **Unit Number: 2** | **Title: Version Control and CI/CD** | **No. of hours: 8** |
| **Content Summary:**<br>**Version Control with Git**: Version control concepts, Git workflows, and collaboration strategies.<br>**Continuous Integration with Jenkins**: Setting up Jenkins pipelines, automated testing, and deployment.<br>**Maven Integration**: Integrate Maven for dependency management and building projects. | | |
| **Unit Number: 3** | **Title: Containerization and Orchestration** | **No. of hours: 8** |
| **Content Summary:**<br>**Introduction to Docker:** Docker concepts, container management, and Docker file creation.<br>**Container Orchestration with Kubernetes:** Kubernetes architecture, deployment, scaling, and networking.<br>**Docker Compose:** Managing multi-container applications with Docker Compose. | | |
| **Unit Number: 4** | **Title: Configuration Management and Monitoring** | **No. of hours: 12** |

**Content Summary:**

**Configuration Management with Ansible**: Ansible playbooks, roles, and infrastructure automation.

**Infrastructure as Code (IaC)**: Terraform for provisioning and managing infrastructure.

**Monitoring and Logging**: Monitoring tools, log management, and application performance monitoring in DevOps.

**Security in DevOps:** Implementing security best practices in CI/CD pipelines and containerized environments.

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures**: Explore DevOps concepts and automation techniques through engaging presentations.
2. **Problem-Based Assignments**: Analyze real-world scenarios to enhance DevOps and automation skills.
3. **Project Labs**: Implement automation tools and practices in hands-on lab sessions.
4. **Collaborative Work**: Work in teams on case studies related to DevOps implementation.
5. **Continuous Feedback**: Receive regular assessments and instructor feedback to monitor progress.

**Outside Classroom Learning Experience**

1. **Moodle Access**: Access course materials anytime via Moodle for flexible learning.
2. **Question Bank**: Utilize a question bank and model papers for effective exam preparation.
3. **Real-World Applications**: Apply DevOps and automation concepts to practical scenarios.

**References**

- Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Authors: Jez Humble and David Farley, Publisher: Pearson Education, Inc., Year: 2011
- The Kubernetes Book, Author: Nigel Poulton, Publisher: Independently published, Year: 2018
- Building Microservices: Designing Fine-Grained Systems, Author: Sam Newman, Publisher: O'Reilly Media, Inc., Year: 2015
- Microservices Patterns: With examples in Java, Author: Eberhard Wolff, Publisher: Manning Publications, Year: 2018
- Terraform: Up & Running: Writing Infrastructure as Code, Author: Yevgeniy Brikman, Publisher: O'Reilly Media, Inc., Year: 2017

**Additional Readings:**

**Online Learning Resources for DevOps & Automation**

I) **Kubernetes Academy by VMware**

a. **Description:** Free courses provided by VMware on Kubernetes, covering everything from basic concepts to advanced orchestration techniques.

b. **Link:** Kubernetes Academy by VMware

II) **HashiCorp Learn: Terraform**

a. **Description:** HashiCorp's official resource for learning Terraform, providing tutorials and hands-on labs for infrastructure as code.

b. **Link:** HashiCorp Learn - Terraform

III) **Docker: Docker for Developers**

a. **Description:** Docker's official training resources for developers, covering containerization, Docker Compose, and more.

b. **Link:** Docker - Docker for Developers

# DEVOPS & AUTOMATION LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** **DevOps & Automation Lab** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSP461** | 0-0-2 | 1 |
| **Type of Course:** | DSE-8 | | |
| **Pre-requisite(s), if any: Basics of DEVEOPS** | | | |

**Defined Course Outcomes**

| COs | Course Outcomes |
|---|---|
| **CO 1** | **Implementing** collaborative development and continuous integration using Git and Jenkins, demonstrating proficiency in version control, automated testing, and deployment processes. |
| **CO 2** | **Developing** and deploy microservices applications using Docker for containerization and Kubernetes for orchestration, managing multi-container applications efficiently. |
| **CO 3** | **Managing** automated infrastructure provisioning and configuration using Ansible and Terraform, demonstrating expertise in infrastructure as code and configuration management. |
| **CO 4** | **Implementing** continuous monitoring, logging, and security best practices in a DevOps environment, ensuring application performance, system health, and data integrity. |

**Lab Experiments**

| S.N | Experiment | Mapped CO(s) |
|---|---|---|
| 1 | Set up a Git repository and practice branching, merging, and collaborative development. Create a small project and manage code versions using Git. | CO1 |
| 2 | Install and configure Jenkins for continuous integration. Create a simple CI pipeline that automatically builds and tests a project whenever code changes are committed to the repository. | CO1 |
| 3 | Implement a continuous delivery pipeline using Jenkins. Deploy a sample application to a staging environment automatically after successful builds and tests. | CO1 |
| 4 | Implement different Git workflows (e.g., GitFlow, Feature Branch Workflow) for a collaborative project. Manage branches, merges, and resolve conflicts. | CO2 |

| | | |
|---|---|---|
| 5 | Set up a Jenkins pipeline for continuous integration. Configure automated testing and deployment for a sample project. Integrate with a version control system like Git. | CO2 |
| 6 | nstall Docker and create Dockerfiles for a sample application. Build, run, and manage containers using Docker commands. | CO3 |
| 7 | Use Docker Compose to manage multi-container applications. Create a Docker Compose file to run a web application with a database and other services. | CO3 |
| 8 | Use Terraform to provision and manage cloud infrastructure. Create Terraform scripts to deploy a web application on a cloud provider (e.g., AWS, Azure). | CO4 |
| 9 | Set up monitoring and logging for a sample application. Use tools like Prometheus, Grafana, and ELK Stack (Elasticsearch, Logstash, Kibana) to monitor and analyze application performance and logs. | CO4 |

# .NET FRAMEWORK

| | | | | |
|---|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
| **Course Name:** **.NET Framework** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENSP413** | 4-0-0 | 4 | 40 |
| **Type of Course:** | DSE-6 | | | |
| **Pre-requisite(s), if any: Basics of Programming** | | | | |

**Course Perspective.** The ".NET Framework" syllabus covers introduction and components of .NET, programming languages, Visual Studio, OOP, exception handling, memory management, Windows Forms/WPF, ASP.NET, web services, .NET Core, Entity Framework, and WCF. Emphasis on practical application and development skills for building robust and secure applications. The course is divided into 4 modules:

a) .NET Framework
b) .NET Framework Fundamentals
c) Building Applications with .NET Framework
d) ASP.NET Framework

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | **Statements** |
|---|---|
| **CO 1** | **Understanding** .NET Framework's architecture, CLR, and CTS for cross-language integration and platform independence. |
| **CO 2** | **Applying** OOP concepts in .NET for designing robust software solutions. |
| **CO 3** | **Utilizing** Visual Studio debugging for diagnosing and fixing errors in .NET applications. |
| **CO 4** | **Demonstrating** proficiency in memory management and garbage collection in .NET. |
| **CO 5** | **Designing** web applications using ASP.NET, incorporating best practices. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| | | |
|---|---|---|
| **Unit Number:1** | **Title: .NETFramework** | **No. of hours: 8** |
| **Content Summary:**<br>NET Framework - Architecture, Common Language Runtime, Common Type System, Namespaces, Assemblies, Memory Management, Process Management, Class Libraries | | |
| **Unit Number:2** | **Title: .NET Framework Fundamentals** | **No. of hours: 8** |
| **Content Summary:**<br>Object-Oriented Programming (OOP) in .NET, Classes, objects, and inheritance,Exception Handling and Debugging, Debugging techniques and tools in Visual Studio, Logging and error reporting in .NET applications, Memory Management and Garbage Collection, Automatic memory management in .NET, Garbage collection, Finalizers and the Dispose pattern | | |
| **Unit Number:3** | **Title: Building Applications with .NET Framework** | **No. of hours: 12** |
| **Content Summary:**<br>.NET - Declaration, Expression, Control Structures, Function, String, Array, Encapsulation, Class, Property, Indexer, Delegate, Inheritance, Interface, Polymorphism, Exception Handling, Modules, Graphics, File handling and Data Access. .NET – Form- Event–Form Controls – Containers – Menus - Data controls - Printing – Reporting – Dialogs – Components - Single and Multiple Document Interfaces. | | |
| **Unit Number:4** | **Title: ASP.NETFramework** | **No. of hours: 12** |
| **Content Summary:**<br>ASP.NET – Web Pages, Web Forms, Web Site Design, Data Controls, Validation Controls, HTML, Navigation Controls, Login Controls, Reports - Master Pages – Web Service Architecture - Basic Web Services – Web Reference – Standards | | |

**Learning Experiences:**

1. **Interactive Lectures**: Explore .NET Framework concepts through engaging presentations.
2. **Problem-Based Assignments**: Analyze real-world scenarios to enhance .NET development skills.
3. **Project Labs**: Build and test applications using the .NET Framework in hands-on labs.
4. **Collaborative Work**: Work in teams on case studies related to .NET application development.
5. **Continuous Feedback**: Receive regular assessments and instructor feedback to track progress.

**Outside Classroom Learning Experience**

1. **Moodle Access**: Access course materials anytime via Moodle for convenient learning.
2. **Question Bank**: Utilize a question bank and model papers for effective exam preparation.
3. **Real-World Applications**: Apply .NET concepts to practical application scenarios.

**Textbooks**

1. Pro C# 8 with .NET Core: Foundational Principles and Practices in Programming by Andrew Troelsen and Philip Japikse, Apress, 9th Edition, 2020
2. Pro ASP.NET Core 3 by Adam Freeman, Apress
3. ASP.NET Core in Action by Andrew Lock

**Additional Readings:**

**Online Learning Resources:**

I) Online Tutorials and Documentation: Direct students to the official Microsoft documentation for .NET Framework, which provides comprehensive guides and resources. Microsoft .NET Documentation

II) Hands-on Coding Exercises: Assign coding exercises from platforms like LeetCode or HackerRank that focus on implementing concepts of .NET Framework. LeetCode HackerRank

III) Project-Based Learning: Encourage students to work on small projects using different aspects of the .NET Framework. Provide examples of projectideas and resources like GitHub repositories for inspiration. GitHub

# .NET FRAMEWORK LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** **.NET Framework Lab** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSP463** | 0-0-2 | 1 |
| **Type of Course:** | DSE-8 | | |
| **Pre-requisite(s), if any: Basics of Programming** | | | |

**Lab Experiments**

**Defined Course Outcomes**

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** and apply object-oriented design principles, exception handling, memory management, and debugging techniques to develop robust .NET applications. |
| **CO 2** | **Developing** graphical user interfaces and handle events in .NET applications to create interactive and user-friendly software solutions. |
| **CO 3** | **Implementing** web development techniques in ASP.NET, including web forms, user authentication, master pages, and web services to build secure and dynamic web applications. |
| **CO 4** | **Analyzing** and utilize data handling, reporting, and visualization techniques to create comprehensive and functional software systems for various domains. |

## Lab Experiments

| S.N | Experiment | Mapped CO/COs |
|---|---|---|
| 1 | Explore the architecture of the .NET Framework. Create a simple console application to understand the basic structure and components of a .NET project. | CO1 |
| 2 | Demonstrate the functionality of the Common Language Runtime (CLR). Create a .NET application that uses various data types and namespaces to show how the CLR manages execution. | CO1 |
| 3 | Implement a .NET application that showcases the Common Type System (CTS). Define and use various data types, and demonstrate type conversion and interoperability. | CO1 |

| | | |
|---|---|---|
| 4 | Create and manage assemblies in a .NET application. Demonstrate how to build, reference, and use assemblies in a multi-project solution. | CO1 |
| 5 | Implement a simple object-oriented application in .NET. Define classes, create objects, and demonstrate inheritance and polymorphism. | CO2 |
| 6 | Implement a .NET application that logs errors and handles exceptions gracefully. Use a logging framework (e.g., NLog, log4net) to record application events and errors. | CO2 |
| 7 | Build a .NET application demonstrating advanced OOP concepts such as encapsulation, properties, indexers, delegates, interfaces, and polymorphism. | CO3 |
| 8 | Create a .NET application that handles graphics and file I/O. Implement functionality to draw shapes, handle images, and perform file read/write operations. | CO3 |
| 9 | Implement a .NET application with a rich user interface. Use forms, event handling, form controls, containers, menus, data controls, printing, and reporting functionalities to create a feature-rich application. | CO3 |
| 10 | Create a basic ASP.NET web application. Design web pages and web forms to understand the structure and components of an ASP.NET project. | CO4 |
| 11 | Develop an ASP.NET application with user authentication. Use login controls to implement user authentication and authorization, and create a simple reporting feature to display user data. | CO4 |
| 12 | Create and consume a basic web service in ASP.NET. Implement a web service that provides data to a client application, and demonstrate how to use web references to integrate the web service with an ASP.NET project. | CO4 |

## NEW AGE PROGRAMMING LANGUAGES

| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | | |
|---|---|---|---|---|
| **Course Name:** **New-Age programming languages** | **Course Code** | **L-T-P** | **Credits** | **Contact Hours** |
| | **ENSP415** | 4-0-0 | 4 | 40 |
| **Type of Course:** | DSE-6 | | | |
| **Pre-requisite(s), if any: Basics of Programming** | | | | |

**Course Perspective.** New-Age programming languages (GO, F#, Clojure, Kotlin) provides an introduction to the concepts and applications of modern programming languages. It explores the features and benefits of GO, F#, Clojure, and Kotlin, and develop practical skills in programming using these languages. The course will cover language syntax, data types, control structures, functional programming concepts, concurrency, and integration with other technologies. The course is divided into 4 modules:

1. GO Programming Language
2. F# Programming Language
3. Clojure Programming Language
4. Kotlin Programming Language

**The Course Outcomes (COs).** On completion of the course the participants will be:

| COs | Statements |
|---|---|
| **CO 1** | **Understanding** principles and paradigms of modern programming languages. |
| **CO 2** | **Developing** proficiency in syntax, data structures, and control flow of each language. |
| **CO 3** | **Exploring** unique features and strengths of each language. |
| **CO 4** | **Applying** development tools to improve code quality and productivity. |
| **CO 5** | **Designing** and implement projects integrating multiple programming languages. |

**CO = Course outcomes.** A student is expected to have learnt concepts and demonstrated/developed abilities or skills related to strategic management at the end of the course.

**Course Outline:**

| **Unit Number: 1** | **Title: GO programming Language** | **No. of hours: 10** |
|---|---|---|
| **Content Summary:**<br>**Overview and Comparison:** Overview of GO, F#, Clojure, and Kotlin, Comparison with traditional programming languages, Installation and setup of development environment,<br>**GO Programming Basics:** Introduction to GO syntax and data types, Control structures in GO, Functions and packages, Arrays, slices, and maps, Structs and custom data types,Pointers and memory management | | |
| **Unit Number: 2** | **Title: F# Programming Language** | **No. of hours: 10** |
| **Content Summary:**<br>Introduction to F# syntax and functional programming concepts, Data Types, Variables, Operators, Decision Making, Loops, Functions, Strings, Options, Immutable data types and pattern matching, Higher-order functions and currying, Asynchronous and parallel programming in F#, Object-Oriented Programming with F#, Database access with F#, Querying and manipulating data using F#, Integration with relational and NoSQL databases | | |
| **Unit Number: 3** | **Title: Introduction to Clojure Programming** | **No. of hours: 10** |
| **Content Summary:**<br>**Introduction to Clojure:** Overview of Clojure and its features, Setting up the development environment,<br>**Basic Syntax and Functional Programming**, Basic syntax and data structures, Functional programming concepts, Immutable data and pure functions, Higher-order functions and recursion, Collections and sequence operations, Restructuring and pattern matching<br>**Error Handling and Testing:** Exception handling and error management in Clojure, Testing strategies and frameworks in Clojure,<br>**Data Manipulation and Transformation:** Data manipulation with Clojure's sequence functions, Data transformation with transducers, Data-driven development with data literals and data readers | | |
| **Unit Number: 4** | **Title: Introduction to Kotlin Programming** | **No. of hours: 10** |

**Content Summary:**

Overview of Kotlin and its advantages, Setting up the development environment, Basic syntax and data types in Kotlin, Conditional statements and loops, Function declarations and parameters, Lambda expressions and higher-order functions,

**Object-Oriented Programming in Kotlin**: Classes, objects, and inheritance, Properties and access modifiers, Interfaces and abstract classes, Understanding nullable and non-nullable types, Safe calls and the Elvis operator, Type inference and smart casting,

**Collections and Functional Programming:** Working with lists, sets, and maps in Kotlin, Collection operations and transformations, Introduction to functional programming concepts in Kotlin, Creating extension functions in Kotlin, Using DSLs for domain-specific problems, Builder pattern and DSL implementation.

**Learning Experiences:**

**Classroom Learning Experience**

1. **Interactive Lectures**: Explore new programming languages through engaging presentations.
2. **Problem-Based Assignments**: Analyze real-world scenarios to enhance programming skills.
3. **Project Labs**: Build and test applications using various new programming languages in hands-on labs.
4. **Collaborative Work**: Collaborate on case studies related to modern programming practices.
5. **Continuous Feedback**: Receive regular assessments and instructor feedback to monitor progress.

**Outside Classroom Learning Experience**

1. **Moodle Access**: Access course materials anytime via Moodle for flexible learning.
2. **Question Bank**: Utilize a question bank and model papers for effective exam preparation.
3. **Real-World Applications**: Apply concepts from new programming languages to practical scenarios.

**Text Books:**

1. The Go Programming Language, Alan A. A. Donovan and Brian W. Kernighan, Addison-Wesley Professional.
2. An Introduction to Programming in Go, Caleb Doxsey, CreateSpace Independent Publishing.

**References**

1. Real-World Functional Programming: With Examples in F# and C#, Tomas Petricek and Jon Skeet, Manning.
2. Programming F# 3.0: A Comprehensive Guide for Writing Simple Code to Solve Complex Problems, Chris Smith, O'Reilly Media.

3. Getting Clojure: Build Your Functional Skills One Idea at a Time, Russ Olsen, O′Reilly.
4. The Joy of Clojure, Michael Fogus and Chris Houser, Manning Publication.
5. Atomic Kotlin, Bruce Eckel and Svetlana Isakova, Mindview LLC.
6. Kotlin in Action, Dmitry Jemerov and Svetlana Isakova, Manning Publication.

**Additional Readings:**

**Online Learning Resources for New-Age Programming Languages**

a) **Go (Golang)**
   1. **Coursera: Programming with Google Go**
      1. **Description:** An introductory course to Go programming, covering language syntax, data structures, and more.
      2. **Link:** Coursera - Programming with Google Go
   2. **Go by Example**
      1. **Description:** A hands-on introduction to Go using annotated example programs.
      2. **Link:** Go by Example

b) **F#**
   1. **Microsoft Learn: Introduction to F#**
      1. **Description:** A series of modules introducing the F# language, its syntax, and functional programming concepts.
      2. **Link:** Microsoft Learn - Introduction to F#

c) **Clojure**
   1. **ClojureBridge**
      1. **Description:** Free Clojure workshops for beginners, including resources and exercises.
      2. **Link:** ClojureBridge
   2. **Learn Clojure: Clojure for the Brave and True**
      1. **Description:** A beginner-friendly book that teaches Clojure through real-world projects and examples.
      2. **Link:** Clojure for the Brave and True

d) **Kotlin**
   1. **Kotlin Lang: Kotlin Documentation**
      1. **Description:** Official Kotlin documentation and tutorials by JetBrains.
      2. **Link:** Kotlin Documentation
   2. **Udacity: Kotlin for Android Developers**

1. **Description:** A course by Udacity focusing on Kotlin for Android development.
2. **Link:** Udacity - Kotlin for Android Developers

## NEW AGE PROGRAMMING LANGUAGES LAB

| | | | |
|---|---|---|---|
| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
| **Course Name:** **New Age Programming languages Lab** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSP465** | 0-0-2 | 1 |
| **Type of Course:** | DSE-8 | | |
| **Pre-requisite(s), if any: Basics of Programming** | | | |

**Course Outcomes (CO)**

| COs | **Statements** |
|---|---|
| CO1 | **Understanding** the fundamental principles and paradigms of modern programming languages. |
| CO2 | **Developing** proficiency in using the syntax, data structures, and control flow constructs of each language. |
| CO3 | **Exploring** the unique features and strengths of each language, such as Go's focus on concurrency, F#'s functional programming capabilities, Clojure's emphasis on immutability and simplicity, and Kotlin's interoperability with existing Java code. |
| CO4 | **Applying** the languages' respective development tools and best practices. |
| CO5 | **Implementing** projects that utilize the strengths of each language to tackle complex problems or tasks. |

## Lab Experiments

| S. N | Experiment Title | CO |
|---|---|---|
| 1 | Develop a RESTful API for a simple blog application in Go. The API should allow users to create, read, update, and delete blog posts. Use Go's built-in net/http package and struct types for handling blog post data | CO1 |
| 2 | Create a command-line tool in Go that fetches and displays current weather information for a specified city. Use a public weather API and Go's JSON parsing capabilities to implement this tool. | CO1 |
| 3 | Set up the F# development environment. Create a simple F# program to demonstrate basic syntax, data types, and variables. | CO1 |
| 4 | Develop a functional calculator application in F#. The calculator should support basic arithmetic operations, as well as more advanced functions like trigonometry and logarithms. Use pattern matching and immutable data structures to handle calculations. | CO2 |
| 5 | Create a small web application in F# using Suave (a lightweight web server library). The application should allow users to register, log in, and create simple posts. Implement basic session management and data storage. | CO2 |
| 6 | Build a financial portfolio tracker in F#. The application should allow users to input and track their investments, calculate current value, and generate reports. Use F#'s asynchronous programming capabilities to fetch real-time stock prices from a financial API. | CO2 |
| 7 | Develop a to-do list application in Clojure. The application should allow users to add, remove, and mark tasks as complete. Use Clojure's sequence operations and immutable data structures to manage tasks. | CO3 |
| 8 | Create a simple web scraper in Clojure. The scraper should fetch data from a specified website, parse the HTML content, and extract specific information. Use Clojure's libraries for HTTP requests and HTML parsing. | CO3 |
| 9 | Develop a Kotlin-based Android application for tracking fitness activities. The app should allow users to log their workouts, view statistics, and set goals. Use Kotlin's object-oriented features and Android SDK for development. | CO4 |

| | | |
|---|---|---|
| 10 | Create a Kotlin DSL (Domain-Specific Language) for generating HTML pages. The DSL should allow users to define HTML structures using Kotlin syntax and generate the corresponding HTML code. | CO4 |

## Summer Internship-III

| **Program Name:** | **B. Tech (Computer Science and Engineering)** | | |
|---|---|---|---|
| **Course Name: Summer Internship-III** | **Course Code** | **L-T-P** | **Credits** |
| | **ENSI451** | **---** | 2 |
| **Type of Course:** | INT-3 | | |
| **Pre-requisite(s), if any: NA** | | | |

**Course Outcomes (CO)**

| | |
|---|---|
| CO1 | **Applying** theoretical knowledge from core subjects to real-world problems in an industry or academic setting. |
| CO2 | **Demonstrating** the acquisition of new technical skills relevant to the field of computer science and engineering during the internship. |
| CO3 | **Developing** a comprehensive case study, project, or research paper that reflects the practical application of internship experiences. |
| CO4 | **Presenting** internship outcomes effectively, showcasing professional growth, technical competencies, and communication skills. |

**Duration:**

The internship will last for six weeks. It will take place after the completion of the 6th semester and before the commencement of the 7th semester.

**Internship Options:**

Students can choose from the following options:

- **Industry Internship (Offline) or Internship in Renowned Academic Institutions (Offline):**
    - Students must produce a joining letter at the start and a relieving letter upon completion.

**Report Submission and Evaluation:**

1. **Report Preparation:**
    - Students must prepare a detailed report documenting their internship experience and submit it to the department. A copy of the report will be kept for departmental records.
2. **Case Study/Project/Research Paper:**

- Each student must complete one of the following as part of their internship outcome:
    1. A case study
    2. A project
    3. A research paper suitable for publication

3. **Presentation:**
    - Students are required to present their learning outcomes and results from their summer internship as part of the evaluation process.

**Evaluation Criteria for Summer Internship (Out of 100 Marks):**

| **valuation Criteria** | **Maximum Marks** |
|---|---|
| **Relevance to Learning Outcomes** | **30 Marks** |
| - Case Study/Project/Research Paper Relevance | 15 Marks |
| - Application of Theoretical Knowledge | 15 Marks |
| **Skill Acquisition** | **40 Marks** |
| - New Technical Skills Acquired | 20 Marks |
| - Professional and Soft Skills Development | 20 Marks |
| **Report Quality** | **15 Marks** |
| - Structure and Organization | 8 Marks |
| - Clarity and Comprehensiveness | 7 Marks |
| **Presentation** | **15 Marks** |
| - Content Delivery | 8 Marks |
| - Visual Aids and Communication Skills | 7 Marks |

| **Total** | **100 Marks** |

Detailed View:

1. **Relevance to Learning Outcomes (30 Marks)**
    - **Case Study/Project/Research Paper Relevance (15 Marks):**
        1. Directly relates to core subjects: 15 marks
        2. Partially relates to core subjects: 10 marks
        3. Minimally relates to core subjects: 5 marks

4. Not relevant: 0 marks

- **Application of Theoretical Knowledge (15 Marks):**
  1. Extensive application of theoretical knowledge: 15 marks
  2. Moderate application of theoretical knowledge: 10 marks
  3. Minimal application of theoretical knowledge: 5 marks
  4. No application of theoretical knowledge: 0 marks

2. **Skill Acquisition (40 Marks)**
   - **New Technical Skills Acquired (20 Marks):**
     1. Highly relevant and advanced technical skills: 20 marks
     2. Moderately relevant technical skills: 15 marks
     3. Basic technical skills: 10 marks
     4. No new skills acquired: 0 marks
   - **Professional and Soft Skills Development (20 Marks):**
     1. Significant improvement in professional and soft skills: 20 marks
     2. Moderate improvement in professional and soft skills: 15 marks
     3. Basic improvement in professional and soft skills: 10 marks
     4. No improvement: 0 marks
3. **Report Quality (15 Marks)**
   - **Structure and Organization (8 Marks):**
     1. Well-structured and organized report: 8 marks
     2. Moderately structured report: 6 marks
     3. Poorly structured report: 3 marks
     4. No structure: 0 marks
   - **Clarity and Comprehensiveness (7 Marks):**
     1. Clear and comprehensive report: 7 marks
     2. Moderately clear and comprehensive report: 5 marks
     3. Vague and incomplete report: 2 marks
     4. Incomprehensible report: 0 marks
4. **Presentation (15 Marks)**
   - **Content Delivery (8 Marks):**
     1. Clear, engaging, and thorough delivery: 8 marks
     2. Clear but less engaging delivery: 6 marks
     3. Somewhat clear and engaging delivery: 3 marks
     4. Unclear and disengaging delivery: 0 marks

- **Visual Aids and Communication Skills (7 Marks):**
    1. Effective use of visual aids and excellent communication skills: 7 marks
    2. Moderate use of visual aids and good communication skills: 5 marks
    3. Basic use of visual aids and fair communication skills: 2 marks
    4. No use of visual aids and poor communication skills: 0 marks

**Total: 100 Marks**

## Applied Programming and Problem-Solving Skills for Campus Interviews

| Program | B.Tech (Computer Science & Engineering) | | |
|---|---|---|---|
| Course Name: **Applied Programming and Problem-Solving Skills for Campus Interviews** | Course Code | L-T-P | Credits |
| | | --- | 2 |
| Type of Course: | MOOC-2 | | |
| Pre-requisite(s), if any: Basics of Java/C++ Programming, DBMS, Data Structure | | | |

Preface:

This comprehensive course is meticulously designed to equip students with essential skills in Data Structures, Algorithms, Object-Oriented Programming, Java, and Database Management Systems, along with essential soft skills and aptitude preparation. Through a combination of self-paced online modules, hybrid learning experiences, and offline assessments, students will gain a profound understanding of foundational and advanced concepts crucial for software development and data management. The course places a strong emphasis on practical applications, problem-solving techniques, and the development of robust, maintainable code.

In addition to the core technical skills, students will be prepared for real-world scenarios through aptitude exams, soft skills training, and mock interviews, ensuring they are well-rounded and industry-ready. A key feature of this course is the integration of free certifications from Infosys Springboard, providing students with recognized credentials that enhance their employability and align with industry standards.

To successfully complete the course, students are required to pass each individual component, contributing to the final mark out of 100. This course is an integral part of the Practical Training Module (Bootcamp training) in the curriculum, and upon successful completion, students will earn 2 credits. The rigorous evaluation process ensures that students not only acquire the necessary knowledge but also demonstrate their ability to apply these skills in professional settings.

Students must obtain specific free certifications from Infosys Springboard (https://infytq.onwingspan.com/web/en/page/home).

Course Outcomes (COs):

**CO1**: **Demonstrating** understanding of basic data structures (arrays, strings, linked lists) and their operations to solve simple computational problems

**CO2**: **Applying** advanced data structures (stacks, queues, trees, graphs) to develop efficient algorithms for complex problem-solving.

**CO3**: **Designing** and implement software solutions using Object-Oriented Programming principles, ensuring code reusability and maintainability.

| | | |
|---|---|---|
| **CO4**: **Developing** robust Java applications, utilizing object-oriented features, exception handling, and file I/O operations effectively<br>**CO5**: **Analyzing** and optimize database systems using SQL for efficient data retrieval, transaction management, and ensuring data integrity. | | |
| SESSION WISE DETAILS | | |
| Module: 1 | Data Structures and Algorithms - Part 1 | No. of hours: 30<br>(Online, Self-Paced) |
| Content Summary: Module 1 covers foundational data structures including arrays, strings, and linked lists. It introduces key operations and practical applications. This foundational knowledge is crucial for advancing to more complex data structures. | | |
| Module: 2 | Data Structures and Algorithms - Part 2 | No. of hours: 30<br>(Online, Self-Paced) |
| Content Summary: Module 2 focuses on advanced data structures such as stacks, queues, trees, and graphs. It covers essential operations, real-world applications, and their role in solving complex problems. Understanding these structures is vital for effective problem-solving and algorithm optimization. | | |
| Module: 3 | Object Oriented Programming | No. of hours:45<br>(Online, Self-Paced) |
| Content Summary: covers the fundamental concepts of OOP, including classes, objects, inheritance, polymorphism, and encapsulation. It emphasizes designing and implementing software using these principles to enhance code reusability and maintainability. | | |
| Module: 4 | Programming using Java | No. of hours: 100<br>(Online, Self-Paced) |
| Content Summary: basics of Java, including syntax, data types, operators, and control structures. It covers object-oriented principles specific to Java, such as classes, objects, inheritance, and polymorphism, along with advanced topics like exception handling and file I/O. The module aims to build strong foundational skills in Java for developing robust applications. | | |
| Module: 5 | Database management Systems (part I) | No. of hours: 60<br>(Online, Self-Paced) |
| Content Summary: fundamental concepts of database systems, including database models, relational databases, and SQL. It covers key topics such as entity-relationship modeling, normalization, and basic query operations. The | | |

| | | |
|---|---|---|
| module provides a solid foundation in designing and managing databases, focusing on effective data retrieval and manipulation using SQL. | | |
| Module: 6 | Database management Systems (part II) | No. of hours: 40<br>(Online, Self-Paced) |
| Content Summary: advanced database concepts, including transaction management, concurrency control, and database security. This module covers complex SQL queries, stored procedures, and triggers, as well as performance optimization techniques. It focuses on enhancing your skills in managing and optimizing large-scale databases, ensuring data integrity, and implementing robust security measures. | | |
| Module: 7 | Aptitude Exam | No. of hours: Online |
| Module: 8 | Independent Evaluation<br>through 3rd party | No. of hours: Offline |
| Module: 9 | Softs Skills | No. of hours: Online |
| Module: 10 | MOCK Interview | No. of hours: Hybrid |

Reference Books:

- "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
- "Cracking the Coding Interview" by Gayle Laakmann McDowell
- "Elements of Programming Interviews" by Adnan Aziz, Tsung-Hsien Lee, and Amit Prakash
- "Head First Java" by Kathy Sierra and Bert Bates
- "Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan
- "Introduction to the Theory of Computation" by Michael Sipser
- "Programming Challenges: The Programming Contest Training Manual" by Steven S. Skiena and Miguel A. Revilla
- "The Algorithm Design Manual" by Steven S. Skiena
- "Algorithms" by Robert Sedgewick and Kevin Wayne
- "Effective Java" by Joshua Bloch

Evaluation Criteria:

| Module | Link | Hrs | Marks |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Data Structures and Algorithms - Part 1 | https://infytq.onwingspan.com/web/en/app/toc/lex_auth_0125409699132620801065_shared/overview | 30 | 10 |
| Data Structures and Algorithms - Part 2 | https://infytq.onwingspan.com/web/en/app/toc/lex_auth_0127667384693882883448_shared/overview | 30 | 10 |
| Object Oriented Programming | https://infytq.onwingspan.com/web/en/app/toc/lex_auth_0125409722749255681063_shared/overview | 40 | 10 |
| Programming using Java | https://infytq.onwingspan.com/web/en/app/toc/lex_auth_012880464547618816347_shared/overview | 100 | 10 |
| Database management Systems (part I) | https://infytq.onwingspan.com/web/en/app/toc/lex_auth_01275806667282022456_shared/overview | 60 | 10 |
| Database management Systems (Part II) | https://infytq.onwingspan.com/web/en/app/toc/lex_auth_0127673005629194241_shared/overview | 40 | |
| Aptitude Exam | Online Sessions to be conducted by KRMU | NA | 10 |
| AMCAT (Aspiring Minds Computer Adaptive Test) | To be organized by KRMU through External agency | NA | 20 |
| Softs Skills | Online Sessions to be conducted by KRMU | NA | 10 |
| MOCK Interview | To be organized by KRMU | NA | 10 |
| | | | 100 |

*Please Note: No end term evaluations will be done

**Student Learning Experiences**

**Inside Classroom Learning Experience:**

1. **Foundational Mastery:** Students will gain a strong foundation in data structures and algorithms, essential for advanced problem-solving in computer science.
2. **Hands-On Programming:** Through extensive Java programming exercises, students will develop robust applications, reinforcing object-oriented principles.

3. **Practical Problem Solving:** Students will apply advanced data structures to solve complex problems, preparing them for real-world challenges.
4. **Real-World Readiness:** Soft skills training and mock interviews will equip students with the interpersonal and professional skills needed for career success.

**Outside Classroom Learning Experience:**

1. **Advanced Database Skills:** The course will enhance students' ability to manage and optimize databases, ensuring data integrity and effective transaction management.
2. **Self-Paced Learning:** The online, self-paced modules allow students to learn at their own pace, ensuring a deep and thorough understanding of each topic.
3. **Comprehensive Evaluation:** Independent evaluations and aptitude exams will rigorously assess students' knowledge and readiness for the tech industry.
4. **Certification and Credibility:** Earning certifications from Infosys Springboard will enhance students' resumes and demonstrate their competence in industry-relevant skills.

## SEMESTER: VIII

## Industrial Project/R&D Project/Start-up Project

| | | | |
|---|---|---|---|
| Program | B.Tech (Computer Science & Engineering) | | |
| Course Name:<br>Industrial Project/R&D Project/Start-up Project | Course Code | L-T-P | Credits |
| | ENSI452 | | 12 |
| Type of Course: | PROJ-4 | | |
| Pre-requisite(s), if any: | | | |

Preface:

The **B.Tech Final Semester Full-Time Project Work** is a culmination of the academic journey for engineering students at the School of Engineering & Technology, K.R. Mangalam University. This detailed Standard Operating Procedure (SOP) is designed to guide students through their project, ensuring a comprehensive, practical, and outcome-driven approach that aligns with the principles of the **National Education Policy (NEP) 2020**.

The SOP provides a framework for students to choose from three types of projects—**Industrial Projects**, **Research & Development (R&D) Projects**, and **Start-up Projects**. It emphasizes experiential learning, real-world problem-solving, and interdisciplinary collaboration, reflecting NEP 2020's focus on holistic development, innovation, and entrepreneurship. Students will work under the mentorship of both internal faculty and external experts, ensuring they are equipped with the skills and knowledge required to excel in industry, research, or entrepreneurship.

This document outlines each stage of the project work, from proposal submission to final evaluation, and offers clear guidelines for successful completion. By adhering to this SOP, students will not only demonstrate their technical proficiency but also contribute meaningfully to industry, academia, and society.

**Standard Operating Procedure (SOP) for B.Tech Final Semester Full-Time Project Work**

### 1. Introduction

The **B.Tech Final Semester Full-Time Project Work** is an essential academic requirement aimed at providing students with the opportunity to apply theoretical knowledge to practical challenges. The project is designed to foster critical thinking, problem-solving, innovation, and research-oriented learning, with a focus on real-world industrial, research, and entrepreneurial domains. Students may choose from:

- **Industrial Project**: Solving real industrial problems in collaboration with an industry partner.

- **Research & Development (R&D) Project**: Contributing to academic and applied research, with external guidance from academic/research institutions.
- **Start-up Project**: Developing and launching innovative start-up ideas with entrepreneurial mentors.

The SOP ensures that the project aligns with **NEP 2020 guidelines**, emphasizing interdisciplinary, practical, and outcome-based learning.

---

## 2. Objectives

The primary objectives of the full-time project are:

- **Application of Theoretical Knowledge**: Enabling students to apply their academic learning to practical problems.
- **Holistic Development**: Promoting interdisciplinary learning, critical thinking, creativity, and problem-solving.
- **Research and Innovation**: Encouraging innovative solutions, leading to publications, patents, or prototypes.
- **Industry Collaboration**: Fostering partnerships with industries for real-world problem-solving.
- **Entrepreneurship Development**: Developing entrepreneurial skills and creating viable start-ups.
- **Global Competency**: Ensuring students develop the skills required to excel in global environments through research, innovation, and collaboration.

---

## 3. Types of Projects

### a) Industrial Project

Students working on **Industrial Projects** will:

- Collaborate with an industry partner.
- Identify specific, real-world challenges faced by the company.
- Propose and implement a solution that provides value to the industry.
- Develop a final product or prototype that can be implemented in the industrial setting.

**Project Proposal**:

- Problem Statement and Objectives: Identify the industrial problem and outline the objectives.
- Proposed Solution: Present a detailed methodology for solving the problem.
- Deliverables: Define tangible deliverables, including prototypes, software, or hardware.
- Expected Impact: Outline the expected impact on the industry.

**Evaluation Criteria**:

- Practical implementation and solution viability (40%)

- Project innovation (20%)
- Industrial applicability and impact (20%)
- Final presentation and report quality (20%)

**b) Research & Development (R&D) Project**

The **R&D Project** focuses on creating innovative research outcomes through collaborations with academic or research institutions. This can result in publications, research reports, or new discoveries.

**Project Proposal**:

- Literature Review: Detailed research on existing work related to the chosen topic.
- Hypothesis/Research Questions: Define the specific research problem or question.
- Methodology: Include data collection, experimental design, and analysis techniques.
- Research Timeline: Step-by-step phases of research with milestones.

**External Mentor**: Collaboration with an **external academic expert** is mandatory for research projects. The external mentor must be a research professional with expertise in the specific field of study.

**Internal Mentor**: Each student will also be assigned an **internal faculty member** who will supervise the project. The internal mentor will ensure that the research meets academic standards and deadlines.

**Evaluation Criteria**:

- Quality of Research and Novelty (30%)
- Research Methodology (25%)
- Contributions to the field (20%)
- Final Report, Presentation, and Publication (25%)

**c) Start-up Project**

The **Start-up Project** involves developing a business model or creating a start-up venture. Students work on a product/service idea that addresses a significant market need or societal problem.

**Project Proposal**:

- Start-up Idea: Explain the business or product idea.
- Market Research: Detailed research on the market, target customers, competitors, and potential revenue streams.
- Business Plan: Define the steps needed to take the idea to market, including funding, development phases, marketing, and operational plans.
- Product Prototype: If applicable, develop a working prototype.

**Mentorship**:

- **External Mentor**: An industry/start-up expert will guide the student in refining the idea, business model, and market strategy.

- **Internal Faculty Mentor**: An internal mentor will provide academic guidance and ensure the start-up idea is feasible and innovative.

**Evaluation Criteria**:

- Start-up viability and market potential (30%)
- Product or service innovation (30%)
- Prototype/Business Model Development (20%)
- Final Pitch/Presentation and Start-up Plan (20%)

---

**4. Roles and Responsibilities**

**a) Student's Responsibilities:**

- Select a suitable project topic based on interests (industrial, R&D, or start-up).
- Draft and submit a detailed proposal with objectives, methodology, timelines, and deliverables.
- Coordinate with both external and internal mentors regularly for feedback and guidance.
- Maintain a weekly progress report for both mentors.
- Submit a final comprehensive report and present the project.

**b) Internal Supervisor:**

- Guide the student throughout the project.
- Provide academic input and ensure that the project aligns with the program outcomes.
- Conduct progress reviews and ensure timelines are adhered to.
- Evaluate the project at the mid-term and final stages.

**c) External Mentor:**

- Offer specialized industrial, research, or entrepreneurial guidance.
- Provide real-world problem insights for industrial and start-up projects.
- Ensure the project is relevant to the chosen industry, research domain, or start-up ecosystem.
- Participate in the final evaluation of the project.

---

**5. Project Phases**

**Phase 1: Proposal Submission and Approval**

- Students will submit a project proposal during the first two weeks of the final semester.
- The proposal must include the problem statement, objectives, literature review (for R&D projects), methodology, and expected outcomes.
- The proposal is subject to review and approval by the internal supervisor and external mentor.

**Phase 2: Planning and Resource Allocation**

- Once approved, the student will develop a project plan that includes:

- **Project Milestones**: Break down the project into smaller tasks with defined milestones.
- **Resource Requirements**: Identify any software, hardware, lab resources, or tools required for the project.
- **Team Roles**: For group projects, define the roles of each team member.
- **Risk Assessment**: Highlight potential risks and the corresponding mitigation strategies.

**Phase 3: Mid-term Review**

- A mid-term review will be conducted halfway through the project to assess progress.
- Students will present their work to a committee consisting of the internal supervisor, external mentor, and department head.
- The review will assess the progress against the timeline and suggest course corrections if needed.

**Phase 4: Final Execution and Evaluation**

- **Industrial Projects**: Students must submit a prototype or industrial report, demonstrating the solution's applicability to the industry.
- **R&D Projects**: Students must submit a final research report or publish findings in academic journals.
- **Start-up Projects**: Students must present a business plan, along with a working prototype, market analysis, and revenue model.

**Phase 5: Final Report Submission and Presentation**

- **Final Report**: The project report should contain a title page, abstract, introduction, problem statement, objectives, methodology, results, discussion, conclusions, future scope, references, and appendices.
- **Presentation**: Students will deliver a final presentation to a panel of evaluators, showcasing their work, findings, or product.
- **Evaluation**: Based on the final report and presentation, students will be awarded marks in accordance with the evaluation rubrics.

---

## 6. Collaboration and Mentorship

For **Research Projects**, the mentorship will involve both:

- **External Mentor**: An academic expert outside the institution, preferably from a reputed university or research institute.
- **Internal Mentor**: A faculty member from the student's department to provide academic and administrative guidance.

For **Industrial Projects**:

- External mentorship will come from industry professionals, preferably from the partnering company.

For **Start-up Projects**:

- External mentorship will involve experienced entrepreneurs, start-up founders, or investors.

Mentors will:

- Provide critical inputs on the technical, business, or research aspects of the project.
- Offer feedback and advice during each phase of the project.

---

## 7. NEP 2020 Guidelines

The project structure is designed to ensure interdisciplinary learning and foster entrepreneurial and research innovation, in line with the **NEP 2020** guidelines:

- **Interdisciplinary Approach**: Students are encouraged to explore projects that bridge different fields of study.
- **Flexibility**: Students have the flexibility to choose between industrial, research, or start-up projects.
- **Experiential Learning**: Real-world problem-solving and hands-on project work are at the core of this initiative.
- **Collaboration**: The integration of external mentors ensures industry and academic collaboration.

---

## 8. Documentation and Submission Requirements

Students are required to:

- Submit their proposal, mid-term report, final report, and any supporting documents via the **Learning Management System (LMS)**.
- Maintain detailed project logs and weekly reports.