K.R. MANGALAM UNIVERSITY
THE COMPLETE WORLD OF EDUCATION

SCHOOL OF EDUCATION

PROGRAMME HANDBOOK

Bachelor in Computer Applications (BCA)

With

Specialization in AI & Data Science [Honours/Honours with Research]

Programme Code:06

(Undergraduate Programme)

(2023-24)

PREAMBLE

At K.R Mangalam University, we believe in the transformative power of education. Our curriculum is designed to equip the learners with the knowledge, skills, and competencies necessary for success in their chosen fields and to prepare them for the challenges of the ever-evolving global landscape. The foundation of our curriculum is rooted in a Learning Outcomes-Based Curricular Framework (LOCF) that ensures that the programmes are designed with clear learning objectives in mind, guiding the teaching and learning process to facilitate learner's growth and achievement. Our goal is to foster a holistic educational experience that not only imparts disciplinary knowledge but also nurtures critical thinking, problem-solving abilities, communication skills, and lifelong learning

The field of Computer Science & Engineering is at the forefront of technological advancements, shaping the world we live in today. It encompasses a diverse range of disciplines, including computer systems, algorithms, software development, networking, artificial intelligence, and more. As technology continues to revolutionize every aspect of our lives, the demand for skilled computer scientists and engineers is ever-increasing.

At K R Mangalam University, BCA with a specialization in AI and Data (Honours/research honours) program is designed to provide students with a comprehensive understanding of the foundational principles and practical skills needed to excel in this dynamic field. During the course, students will delve into subjects such as programming languages, data structures, probability and stastistics, operating systems, database management, java, and software engineering etc.

At our institution, we emphasize on a hands-on approach to learning, combining theoretical knowledge with practical application. Students will have the opportunity to work on real-world projects, engage in laboratory experiments, and participate in internships to gain valuable industry experience. We believe that this experiential learning will not only strengthen technical proficiency but also foster critical thinking, problem-solving abilities, and teamwork skills.

Furthermore, our curriculum is designed to keep pace with the rapidly evolving nature of the computer science and engineering field. We strive to incorporate the latest trends and emerging technologies, ensuring that our graduates are equipped with the knowledge and adaptability necessary to thrive in a competitive industry.

As technology continues to reshape our world, computer scientists and engineers have a pivotal role to play in driving innovation and creating solutions to complex challenges. The BCA with specialization in AI and Data Science program aims to nurture and empower the next generation of professionals who will shape the future of technology.

We are committed to providing a supportive and inclusive learning environment, where students can explore their passions, develop their skills, and unlock their full potential. Through dedicated faculty, state-of-the-art infrastructure, and a vibrant community, we strive to create an enriching educational experience that prepares students for successful careers in the field of Computer Science & Engineering.

We invite aspiring students to embark on this exciting journey with us, as together, we explore the limitless possibilities of computer science and engineering and make a positive impact on the world.

Furthermore, our curriculum is designed to keep pace with the rapidly evolving nature of the computer science and engineering field. We strive to incorporate the latest trends and emerging technologies, ensuring that our graduates are equipped with the knowledge and adaptability necessary to thrive in a competitive industry.

The curriculum is aligned with the needs of the industry and the job market and is flexible enough to adapt to changing trends and technologies. It integrates cross-cutting issues relevant to professional ethics, gender, human values, environment and Sustainable Development Goals (SDGs). All academic programmes offered by the University focus on employability, entrepreneurship and skill development and their course syllabi are adequately revised to incorporate contemporary requirements based on feedback received from students, alumni, faculty, parents, employers, industry and academic experts.

We are committed to implementing the National Education Policy (NEP) 2020 in its entirety, and to creating a more inclusive, holistic, and relevant education system that will prepare our students for the challenges of the 21st century. With the focus on Outcome-Based Education (OBE), our university is continuously evolving an innovative, flexible, and multidisciplinary curriculum, allowing students to explore a creative combination of credit-based courses in variegated disciplines along with value-addition courses, Indian Knowledge Systems, vocational courses, projects in community engagement and service, value education, environmental education, and acquiring skill sets, thereby designing their own learning trajectory.

The Bachelor of Computer Application (BCA) in AI and Data Science program at K.R Mangalam University is a comprehensive four years curriculum built upon the LOCF to prepare aspiring educators to acquire the graduate attributes for a successful career in teaching. The programme consists of a combination of core courses, elective courses, and field experiences. This Programme Handbook serves as a roadmap for students and provides detailed information about the structure, learning outcomes, courses offered, and assessment methods within the BCA programme. We encourage all students to utilize this handbook as a valuable resource throughout their academic journey.

1. UNIVERSITY VISION AND MISSION

K.R. Mangalam University is the fastest-growing higher education institute in Gurugram, India. Since its inception in 2013, the University has been striving to fulfil its prime objective of transforming young lives through ground-breaking pedagogy, global collaborations, and world-class infrastructure.

Recognized for its virtues of quality, equality, inclusiveness, sustainability, and professional ethics, KRMU is synonymous with academic excellence and innovation.

1.1. VISION

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

1.2 MISSION

1. Foster employability and entrepreneurship through a futuristic curriculum and progressive pedagogy with cutting-edge technology.

2. Instil the notion of lifelong learning through stimulating research, outcomes-based education, and innovative thinking.

3. Integrate global needs and expectations through collaborative programs with premier universities, research centers, industries, and professional bodies.

4. Enhance leadership qualities among the youth by having an understanding of ethical values and environmental realities.

2. SCHOOL OF ENGINEERING AND TECHNOLOGY (SOET)

2.1 About the School of Engineering and Technology

Since 2016, the School of Engineering and Technology (SOET) strives to foster and maintain a creative environment with a deep commitment to inculcate excellence in academics and contribute towards students' development. The school brings an attitudinal change in prospective teachers for their advancement into accountable agents of change in society, who are sensitive to local, national, and global concerns and issues vital for human survival, progress, and development. The School of Engineering and Technology offers diverse programs of studies that are designed to develop an insight into the nuances of teaching and learning in terms of theoretical perspectives, and pedagogical techniques that facilitate the student's understanding of the social, emotional, and intellectual ecosystem.

3. SCHOOL VISION AND MISSION

3.1 School Vision

To create, disseminate, and apply knowledge in science and technology to meet the higher education needs of India and the global society, to serve as an institutional model of excellence in scientific and technical education characterized by the integration of teaching, research, and innovation. School of Education aspires to become an internationally recognized department through excellence in the interdisciplinary arena of education, research, and innovation, preparing socially responsible lifelong learners contributing to nation-building.

3.2 School Mission

- To create an environment where teaching and learning are prioritized, with all support activities being held accountable for their success.
- To strengthen the institution's position as the school of choice for students across the State & Nation.
- To promote creative, immersive, and lifelong learning skills while addressing societal concerns.
- To promote co- and extra-curricular activities for the overall personality development of the students.
- To promote and undertake all-inclusive research and development activities.

  Enhance industrial, institutional, national, and international partnerships for symbiotic relationships.
- To help students acquire and develop knowledge, skills, and leadership qualities of the 21st Century and beyond.

4. INTRODUCTION TO BACHELOR OF COMPUTER APPLICATION IN AI AND DS

TheBachelor in Computer Applications (BCA) with Specialization in AI & Data Science [Honours/Honours with Research] is a Four-year program brought in association with Samatrix and IBM, BCA in AI & Data Science focuses on the deepest insights into the science of Big Data Analytics and how Artificial Intelligence is driving the growth. Students will receive both theoretical and practical knowledge in subjects like Machine learning, neural network, Web programming, net frame working, Data Warehousing, Data Mining, Mobile Application Development & handling data.

Note** Students who wish to exit after the first two semesters will undergo a 4-credit work-based learning/internship during the summer term in order to get a UG Certificate.

Note: Students on exit after one year will earn Undergraduate Certificate Program in Computer Applications. Students needs to earn minimum of 40 credits.

4.1. Program Highlights

Building exceptional understanding and expertise to turn ideas into solutions, the Bachelor's in Computer Applications aims at ensuring rigorous pragmatic training and hands-on experience in working on advanced new-age systems and technologies.

- The curriculum is specifically designed in consultation with industry insiders and experts
- Realistic hands-on training for absolute excellence
- Globally accredited certification upon successful completion of the BCA with a Specialization in AI and data science program
- Consistent mentoring by acclaimed academicians and top industry experts
- Highly sophisticated laboratories equipped with cutting-edge tech apparatus
- Ensuring absolute preparedness for successful career progression

4.2. Aims of Bachelor of Computer Application (BCA in AI and DS) Programme

The Bachelor's program in Computer Application with a specialization in AI and Data Science is aligned with the needs and demands of the industry and the current trends in technology. Some key aims of the program are Strong Foundation in Computer Science, Artificial Intelligence fundamentals and Data Science Techniques, Practical Experience, and Hands-on experience that is crucial for preparing students to work in real-world AI and data science.

5. LEARNING OUTCOME-BASED CURRICULUM FRAMEWORK IN BACHELOR OF COMPUTER APPLICATION IN ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

The Learning Outcomes-based Curriculum Framework (LOCF) for the BCA in AI and DS programme provides a framework for the student-teachers to develop a range of knowledge, skills, attitudes, and values that teachers should possess to meet the educational needs of diverse learners, create an engaging and inclusive learning environment and contribute to the overall improvement of the education system. The curriculum has clearly articulated learning outcomes that describe what students should be able to know, understand, and demonstrate by the end of the programme. It integrates theoretical knowledge with practical application and allows for flexibility and adaptation to meet the needs of individual student-teacher and changing educational contexts. It offers elective courses or specialization options, enabling student-teachers to pursue their areas of interest or specialization within the broader field of engineering and its applications. The curriculum includes various assessment methods and tools to measure the attainment of learning outcomes.

6. GRADUATE ATTRIBUTES OF BACHELOR OF COMPUTER APPLICATION IN AI AND DS

Graduate attributes are the qualities, skills, knowledge, and attitudes that students are expected to develop and possess upon completion of a Bachelor of Computer Application programme. The graduate attributes of a Bachelor of Computer Application (BCA) program with a specialization in AI and Data Science are expected to develop students specifically for careers in these cutting-edge fields. With a deep understanding of the principles, methodologies, and techniques of AI and Data Science. They should be proficient in machine learning algorithms, statistical analysis, data mining, natural language processing, and computer vision.

GA1. **Programming Proficiency**: Graduates should be highly skilled in programming languages commonly used in AI and Data Science, such as Python, Java, and other relevant languages. They should be able to implement and optimize AI models and data processing pipelines.

GA2. **Data Handling and Pre-processing:** Graduates should be adept at collecting, cleaning, and pre-processing large datasets. They should know how to handle real-world data and understand the importance of data quality and integrity.

GA3. Data Visualization: Graduates should be capable of visualizing data effectively to communicate insights and findings to various stakeholders. They should be familiar with data visualization libraries and tools.

GA4. **Problem-Solving and Critical Thinking**: Graduates should possess strong problem-solving and critical thinking skills to address complex AI and data science challenges. They should be able to design and develop innovative solutions to real-world problems.

GA5. Ethical AI and Data Usage: Graduates should be aware of the ethical considerations and legal implications of using AI and handling data. They should understand the importance of privacy, fairness, transparency, and bias mitigation in AI systems.

GA6. Research and Experimentation: Graduates should be trained to conduct research and experiments in AI and Data Science. They should be able to analyze experimental results and draw meaningful conclusions.

GA7. **Industry-Relevant Projects:** The program should provide opportunities for students to work on real-world AI and Data Science projects, either through internships or industry collaborations. This practical experience will enhance their employability.

GA8. **Teamwork and Collaboration**: Graduates should be adept at working in multidisciplinary teams, as AI and Data Science projects often involve collaboration with domain experts and other professionals.

GA9. **Lifelong Learning:** The field of AI and Data Science is continuously evolving. Graduates should be equipped with a mind set for lifelong learning to stay updated with the latest advancements and technologies.

GA10. Communication Skills: Graduates should be able to communicate complex technical concepts to both technical and non-technical audiences effectively. Strong communication skills are essential for presenting findings and project outcomes.

GA11. **Domain Knowledge:** Depending on their specific interests and electives, graduates may have domain expertise in areas like computer vision, natural language processing, robotics, healthcare analytics, finance, etc.

7. QUALIFICATION DESCRIPTORS FOR THE BACHELOR OF COMPUTER APPLICATION (BCA) PROGRAMME

The students who complete two years of full-time study will be awarded a Bachelor of Computer Application in AI and DS (BCA-AI&DS) degree. Qualification descriptors for a Bachelor of Computer Application in AI and DS program outline the knowledge, skills, and competencies that students are expected to acquire upon completion of the programme. These descriptors serve as benchmarks for assessing the readiness of graduates to enter the engineering application profession and may include:

1. demonstrate a comprehensive understanding of the theories, principles, and concepts related to engineering.

2. proficient in developing software applications for various platforms, utilizing industry-standard tools and methodologies.
3. will demonstrate the ability to analyze complex problems, identify requirements, and develop effective solutions using appropriate computational and algorithmic techniques.
4. skilled in data handling, including data storage, retrieval, manipulation, and analysis.
5. working knowledge of web development technologies, including HTML, CSS, JavaScript, and server-side scripting languages.
6. importance of information security and be able to apply security measures to protect data and systems from potential threats and vulnerabilities.
7. will possess effective written and oral communication skills, enabling them to convey technical concepts and collaborate with team members and stakeholders.
8. able to work effectively in multidisciplinary teams, contributing their expertise to achieve project goals and objectives.
9. will adhere to high ethical standards in their professional practice, respecting intellectual property rights, privacy, and confidentiality of data.
10. students may have specific knowledge and skills in areas such as artificial intelligence, data science, mobile application development, depending on the elective subjects.

8. PROGRAMME EDUCATIONAL OBJECTIVES (PEO)

- **PEO1 -** Develop expertise in AI and Data Science through research-oriented learning and gain the ability to apply advanced concepts in practical settings.
- **PEO2 -** Pursue a successful career in AI and Data Science by applying knowledge and skills to industry-related problems or further academic pursuits.
- **PEO3 -**Engage in lifelong learning and develop innovative solutions in the domain of AI and Data Science using research and problem-solving skills to contribute to the sustainable development of society.
- **PEO4 -** Demonstrate leadership, teamwork, management, ethical and social responsibility, and communication skills with a commitment to lifelong learning.

PROGRAMME OUTCOMES (PO)

Engineering Graduates will be able to:

- **PO1. Fundamental Knowledge:** Demonstrate a strong foundation in computer science principles, mathematics, and fundamental concepts necessary for the understanding and application of computing.
- **PO2. Problem Solving:** Apply analytical and critical thinking skills to identify, analyze, and solve complex computing problems using appropriate tools, algorithms, and programming languages.
- **PO3. Software Development:** Design, develop, and implement software solutions by applying software engineering principles, programming languages, and best practices.
- **PO4. Database Management:** Design, implement, and manage databases, ensuring efficient data storage, retrieval, and manipulation using database management systems.

- **PO5. Web Technology:** Develop web-based applications and utilize relevant technologies, frameworks, and tools for effective web development and deployment.
- **PO6. Network and Security:** Understand network protocols, security mechanisms, and implement secure network configurations to ensure data integrity, confidentiality, and availability.
- **PO7. Information Management:** Gather, organize, and analyze information from various sources using appropriate technologies and tools for effective decision-making and problem-solving.
- **PO8. Team Collaboration:** Collaborate effectively as a member or leader in multidisciplinary teams, demonstrating effective communication, interpersonal skills, and adaptability to work in diverse professional environments.
- **PO9. Ethical and Professional Practices:** Adhere to ethical, legal, and professional standards in computing, recognizing the social and ethical responsibilities associated with the use of technology.
- **PO10. Lifelong Learning:** Recognize the need for continuous learning, keep up-to-date with emerging trends in technology, and engage in self-directed learning to adapt to evolving computing paradigms.
- **PO11. Industry Relevance:** Apply industry-relevant practices, tools, and technologies to bridge the gap between academia and industry, ensuring the ability to meet industry requirements and contribute effectively to the computing field.
- **PO12. Entrepreneurial Mind set:** Identify opportunities, demonstrate innovation, and apply entrepreneurial thinking to create value, solve problems, and contribute to the growth of businesses and society.

9. PROGRAMME SPECIFIC OUTCOMES (PSO)

PSO1 - Demonstrate a strong foundation in the theoretical concepts and practical applications of AI and Data Science, including machine learning, data analysis, and data visualization**.**

PSO2 - Develop critical thinking, problem-solving, and research skills through hands-on projects, independent study, and engagement in research activities related to AI and Data Science**.**

PSO3 - Apply knowledge and skills to solve industry-related problems through internships, capstone projects, and experiential learning opportunities in collaboration with industry partners.

PSO4 - Demonstrate leadership, teamwork, management, ethical and social responsibility, and communication skills by engaging in team-based projects, leadership activities, and community outreach initiatives related to AI and Data Science.

MAPPING OF SCHOOL VISION, MISSION WITH PROGRAMME OUTCOMES (PO) AND PROGRAMME SPECIFIC OUTCOMES(PSO)

School Vision	School Mission	Programme Outcomes (PO)	Programme Specific Outcomes (PSO)

To create, disseminate, and apply knowledge in science and technology to meet the higher education needs of India and the global society, to serve as an institutional model of excellence in scientific and technical education characterized by the integration of teaching, research, and innovation. School of Education aspires to become an internationally recognized department through excellence in the interdisciplinary arena of education, research, and innovation, preparing socially responsible lifelong learners contributing to nation-building.	**M 1**	**PO1, PO2**	**PSO 1, PSO 2**
	M 2	**PO 3**	
	M 3	**PO 4, PO 5, PO 6, PO 7**	**PSO 6**
	M 4	**PO 8, PO 9** **PO 10,**	**PSO 4,**
	M 5		
	M 6	**PO 11, PO 12**	**PSO 3, PSO 4**
	M 7		
	M 8		

10. PROGRAMME DURATION

Name of the Programme	**Duration**
Bachelor of Computer Application with specialization in AI and DS [honours/honours research]	4Years (8 Semesters)

12. CAREER AVENUES

Students who graduate from BCA with a specialization in Artificial Intelligence and Data Science will have the opportunity to go into various job profiles such as

- Software Engineer
- Business Analyst
- Data Scientist
- Digital Marketer
- Cyber Security Experts
- Go for MCA
- Software Developer

- Block chain Professional

13. ELIGIBILITY CRITERIA

1. Candidate must have passed the 10+2 examination or equivalent in any stream with mathematics/IP and CS as one subject with minimum 50% aggregate marks.

2. The reservation and relaxation for SC/ST/OBC/PwD and other categories shall be as per the rules of the Central Government/ State Government, whichever is applicable.

14. CLASS TIMINGS

The class will be held from Monday to Friday from 9.10 A.M. to 4.00 P.M.

15. TEACHING- LEARNING PROCESS

The teaching and pedagogy at the School of Engineering and Technology focus on preparing students to become skilled professionals in various fields of engineering.

1. Lectures

Many courses begin with lectures in which professors introduce important concepts, theories, and frameworks to the class. In order to improve comprehension and application, lectures are frequently augmented with visual aids, real-world examples, and case studies.

2. Professional Development

In order to prepare students for the workforce, the School of Engineering and Technology frequently incorporates professional development events. Including workshops, seminars, guest lectures, and industrial visits. Students have the chance to meet with professionals in the field and learn about current industry trends and practices.

3. **Guest Speakers and Industry Experts**

Students are exposed to real-world experiences and ideas from professionals by inviting guest lecturers and industry experts to provide lectures or take part in panel discussions. This offers students insights into the industry, chances to network, and practical knowledge outside of the classroom.

4. Theory and Practice Integration

Engineering and technology programmes work to combine theoretical ideas with real-world applications. Fundamental ideas and theories are taught to students, who subsequently use them practically. With this method, teachers can be sure that students not only comprehend the underlying theory but also acquire the abilities necessary to use their newfound knowledge in practical ways.

16. ASSESSMENT METHODS

Continuous assessment is employed to monitor student progress and provide feedback for improvement. Assessments include examinations, project evaluations, presentations, and practical demonstrations. Constructive feedback from instructors allows students to identify areas of strength and areas that require further development, facilitating their overall growth. Formative assessments such as class discussions, group activities, projects, quizzes, assignments, and presentations are conducted throughout the teaching-learning process, enabling teachers to monitor student progress continuously. Teachers provide oral or written feedback, engage in one-on-one discussions, and use rubrics and checklists to communicate student performance.

7. MINIMUM ACCEPTABLE LEVEL OF ACADEMIC STANDARDS

The minimum acceptable level of achievement that a student must demonstrate to be eligible for the award of academic credit or qualification is the minimum acceptable level of academic standards. The Letter Grades and Grade Points which shall be used to reflect the outcome of the assessment process of the student’s performance is indicated in Table 1.

Table 1

Marks Range (%)	Letter Grade	Grade Points	Description of the Grade
>90	O	10.0	Outstanding

80-90	A+	9.0	Excellent
70-80	A	8.0	Very Good
60-70	B+	7.0	Good
55-60	B	6.0	Above Average
50-55	C	5.5	Average
40-50	P	5.0	Pass
<40	F	0	Fail
-	AB	0	Absent
% marks≥ 50	S	-	Satisfactory
% marks <50	US	-	Unsatisfactory
	W	0	Withdrawal

18. PROGRAMME STRUCTURE

THREE-YEAR BCA in AI & DS PROGRAMME AT A GLANCE

	Semester I	**Semester II**	**Semester III**	**Semester IV**	**Semester V**	**Semester VI**	**Total**
Courses	9	8	10	10	10	9	56
Credits	24	21	26	25	26	23	169

19 Scheme of Studies for BCA (AI&DS) Programme

SOET	SCHEME OF STUDIES (2023-26)																BCA
YEAR	**ODD SEMESTER (I)**								**EVEN SEMESTER (II)**								
	S. NO	**COURSE CODE**	**COURSE TITLE**	**Course Type**	**L**	**T**	**P**	**C**	**S. NO**	**COURSE CODE**	**COURSE TITLE**	**Course Type**	**L**	**T**	**P**	**C**	
FIRST	1	ENCA101	Web Designing Using HTML,CSS, Java Script & PHP	Major	4	0	-	**4**	1	ENCA102	Fundamentals of Object Oriented Programming using C++	Major	3	1	-	**4**	
	2	ENMA 103	Basics of Mathematics	Major	3	1	-	**4**	2	ENCA104	Discrete Structure	Major	3	1	-	**4**	
	3	ENSP101	Clean Coding with Python	Minor	4	0	0	**4**	3	ENSP102	Overview of AI, Data Science, Ethics and Foundation of Data Analysis	Minor	4	0	0	**4**	
	4	ENCA103	Essentials of Software Engineering	Major	3	1	0	**4**	4	SEC039	R Programming for Data Science and Data Analytics Lab	SEC	0	0	4	**2**	
	5	ENCA 151	Web Design Lab	Major	-	-	2	**1**	5	ENCA152	Object Oriented Programming Lab using C++	Major	-	-	2	**1**	
	6	ENSP151	Clean Coding with Python Lab	Minor	0	0	2	**1**	6	ENSP152	Overview of AI, Data Science, Ethics and Foundation of Data Analysis Lab	Minor	0	0	2	**1**	
	7		Environmental Studies & Disaster Management (Online Moodle)	VAC I	2	-	-	**2**	7		***Open Elective -1***	Open Elective	3	-	-	**3**	
	8	ENCA153	Essentials of Software Engineering Lab	Major	0	0	1	**2**	8	ENCA202	Extension Activities(community engagement service)	VAC II	2	-	-	**2**	
	9	SEC037	Data Visualization using Power BI	SEC	0	0	4	**2**									
			TOTAL		**16**	**2**	**9**	**24**			**TOTAL**		**15**	**2**	**8**	**21**	

SOET	**SCHEME OF STUDIES (2023-26)**												**BCA**			
YEAR	**ODD SEMESTER (I)**								**EVEN SEMESTER (II)**							
	S. NO	**COURSE CODE**	**COURSE TITLE**	**Course Type**	**L**	**T**	**P**	**C**	**S. NO**	**COURSE CODE**	**COURSE TITLE**	**Course Type**	**L**	**T**	**P**	**C**
SECOND	1	ENCA201	Fundamentals of Data Structures	Major	3	1	0	**4**	1	ENCA202	Fundamentals of Operating Systems	Major	4	-	-	**4**
	2	ENCA203	Fundamentals of Java Programming	Major	3	1	0	**4**	2	ENCA204	Fundamentals of Database Management Systems	Major	3	1	0	**4**
	3	ENSP205	Probabilistic Modelling and Reasoning	Minor	0	0	4	**2**	3	ENSP212	Foundation of Machine Learning	Minor	4	0	0	**4**
	4	ENCA205	Fundamentals of Artificial Intelligence	Major	3	1	0	**4**	4	ENCA252	Fundamentals of Operating System Lab	Major	-	-	2	**1**
	5	ENCA251	Fundamentals of Java Programming Lab	Major	0	0	2	**1**	5	ENSP262	Foundation of Machine Learning Lab	Minor	0	0	2	**1**
	6	ENCA253	Fundamentals of Data Structures Lab	Major	0	0	2	**1**	6	ENCS254	Fundamentals of Database Management Systems Lab	Major	0	0	2	**1**
	7		Open Elective - II	Open Elective	3	0	0	**3**	7		***Open Elective -III***	Open Elective	3	0	0	**3**
	8		VAC III	VAC	2	0	0	**2**	8	AEC012	Life Skills for Professionals-II	AEC	3	0	0	**3**
	9	AEC011	Life Skills for Professionals-I	AEC	3	0	0	**3**	9		VAC IV	VAC	2	0	0	**2**
	10		Summer Internship /Project	INT	0	0	0	**2**	10	ENSI252	***Minor project-I***		0	0	0	**2**
			TOTAL		**17**	**3**	**8**	**26**			**TOTAL**		**19**	**1**	**6**	**25**

SOET	SCHEME OF STUDIES (2023-26)																BCA
YEAR	ODD SEMESTER (I)									EVEN SEMESTER (II)							
	S.NO	COURSE CODE	COURSE TITLE	Course Type	L	T	P	C		S.NO	COURSE CODE	COURSE TITLE	Course Type	L	T	P	C
THIRD	1	ENCA301	Design and Analysis of Algorithms	Major	3	1	0	**4**		1		Department Elective I	Minor	4	-	-	**4**
	2	ENCA303	Theory of Automata	Major	3	1	0	**4**		2	ENCA302	Introduction to Computer Organization & Architecture	Major	3	1	-	**4**
	3	ENSP302	Introduction to Natural Language Processing	Minor	4	0	0	**4**		3	ENCA304	Introduction to Computer Networks	Major	4	-	-	**4**
	4	ENSP309	Big Data Analysis with Scala and Spark	Minor	4	-	-	**4**		4	ENCA306	Basics of Neural Networks and Deep Learning	Major	4	-	-	**4**
	5	SEC040	Data Science - Tools and Techniques Lab	SEC	0	0	4	**2**		5		Department Elective I Lab	Minor	-	-	2	**1**
	6	AEC013	Life Skills for Professionals-III	AEC	3	0	0	**3**		6	ENCA352	Computer Networks Lab	Major	-	-	2	**1**
	7	ENCA351	Design & Analysis of Algorithms Lab	Major	0	0	2	**1**		7	ENCA354	Neural Networks and Deep Learning Lab	Major	-	-	2	**1**
	8	ENSP352	Natural Language Processing Lab	Minor	0	0	2	**1**		8	SEC036	Competitive Coding	SEC	-	-	4	**2**
	9	ENSP359	Big Data Analysis with Scala and Spark Lab	Minor	-	-	2	**1**		9	ENSI352	Minor Project-II	Proj	-	-	-	**2**
	10		Summer Internship /Project	INT	0	0	0	**2**									
			TOTAL		**17**	**2**	**10**	**26**				**TOTAL**		**15**	**1**	**10**	**23**

SEMESTER-WISE STRUCTURE FOR BCA in AI & DS PROGRAMME

YEAR	ODD SEMESTER (I)							
	S.NO	COURSE CODE	COURSE TITLE	Course Type	L	T	P	C
FIRST	1	ENCA101	Web Designing Using HTML,CSS, Java Script & PHP	Major	4	0	-	**4**
	2	ENMA 103	Basics of Mathematics	Major	3	1	-	**4**
	3	ENSP101	Clean Coding with Python	Minor	4	0	0	**4**
	4	ENCA103	Essentials of Software Engineering	Major	3	1	0	**4**
	5	ENCA 151	Web Design Lab	Major	-	-	2	**1**
	6	ENSP151	Clean Coding with Python Lab	Minor	0	0	2	**1**
	7		Environmental Studies & Disaster Management (Online Moodle)	VAC I	2	-	-	**2**
	8	ENCA153	Essentials of Software Engineering Lab	Major	0	0	1	**2**
	9	SEC037	Data Visualization using Power BI	SEC	0	0	4	**2**
			TOTAL		**16**	**2**	**9**	**24**

Department:	**Department of Computer Science and Engineering**		
Course Name: Web Designing Using HTML,CSS, Java Script & PHP	**Course Code ENCA101**	**L-T-P**	**Credits**
		3-0-2	4
Type of Course:	Major		
Pre-requisite(s), if any: Basic computer skills and familiarity with HTML and CSS are recommended pre-requisites for this syllabus.			

COURSE OBJECTIVES

The course will enable the student-teacher to:

COs	**Statements**
CO1	Understand: Demonstrate a clear understanding of the fundamental concepts and principles of web designing using HTML, CSS, JavaScript, and PHP.
CO2	Express: Express ideas and concepts effectively through the implementation of HTML, CSS, JavaScript, and PHP in web design projects.
CO3	Determine appropriate strategies and techniques to solve web design problems using HTML, CSS, JavaScript, and PHP.
CO4	Identify: Identify and analyze the different components and elements required for effective web design, including HTML structure, CSS styles, JavaScript functions, and PHP scripts.
CO5	Articulate: Articulate the purpose, functionality, and interaction of various web design elements, including HTML tags, CSS selectors, JavaScript events, and PHP database operations.
CO6	Design: Design and develop visually appealing and responsive web pages using HTML, CSS, JavaScript, and PHP, considering user experience and accessibility.

CATALOG DESCRIPTION

Brief Syllabus:
The syllabus for "Web Designing Using HTML, CSS, JavaScript, and PHP" covers the essential elements of web design and development. It starts with an introduction to HTML, teaching students about the structure of HTML documents, tags, and text formatting. CSS is then introduced, covering selectors, styling text and backgrounds, and working with colors and fonts. The course progresses to JavaScript, where students learn about variables, data types, conditional statements, loops, functions, and events. They also gain knowledge of DOM manipulation, allowing them to interact with HTML elements dynamically. The next unit focuses on PHP, teaching students server-side scripting. They learn about PHP syntax, variables, handling forms and user input, working with files and directories, and integrating databases using MySQL. The final unit involves a web design project. Students plan and design a website, implementing their skills in HTML, CSS, JavaScript, and PHP. They learn about integrating front-end and back-end development, testing, debugging, and deploying the project to a hosting server. Overall, this syllabus equips students with the necessary skills to create visually appealing and interactive websites using HTML, CSS, JavaScript, and PHP.

UNIT WISE DETAILS

Unit Number: 1	**Introduction to HTML (Hypertext Markup Language) and CSS**	**No. of hours: 4**

Content Summary:
Introduction to HTML (Hypertext Markup Language), Basic structure of HTML documents HTML tags and elements , Text formatting and links , Images, Introduction to CSS (Cascading Style Sheets), Internal, External and Embedded CSS, selectors Styling text and backgrounds Working with colors and fonts.

Unit Number: 2	**Advanced HTML and CSS**	**No. of hours: 8**

Content Summary:
HTML forms and input elements, HTML tables, CSS layout techniques, Box model and positioning Responsive design principles, CSS transitions and animations, Working with media (images, audio, and video).

Unit Number: 3	**JavaScript Fundamentals**	**No. of hours: 8**

Content Summary:
Introduction to JavaScript, JavaScript variables and data types, Java operators, Conditional statements and loops, JS Functions and events, JavaScript Arrays, DOM, Data Validation using JS, Accessing and modifying HTML elements.

Unit Number: 4	**Dynamic Web Development with PHP**	**No. of hours: 8**

Content Summary:
Introduction to server-side scripting, PHP syntax and variables, PHP data types, working with forms and user input, Handling files and directories, working with databases (MySQL), Database connections and queries, Inserting, updating, and retrieving data.
***Self-Learning Components:**
1. Codecademy Web Development Skill Path: https://www.codecademy.com/learn/paths/web-development
2. FreeCodeCamp Responsive Web Design Certification: https://www.freecodecamp.org/learn/responsive-web-design/
3. Codecademy Web Development Skill Path: https://www.codecademy.com/learn/paths/web-development

Reference Books:
1. JavaScript and JQuery: Interactive Front-End Web Development" by Jon **Duckett**

Assessment & Evaluation

Components	Assignment	Mid Term Examination	Attendance	End Term Examination
Weightage (%)	20	20	10	50

Programme and Course Mapping

Course Code and Title	Course outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENCA101 /Web Designing Using	CO1	3	2	3	2	2	2	-	-	1	1	1	1	3	2	3	2
	CO2	3	2	3	2	2	1	1	1	1	2	1	2	3	3	3	2
	CO3	3	3	3	3	3	2	2	1	2	1	2	1	3	3	3	3
	CO4	2	2	3	3	3	1	-	-	1	2	1	1	3	3	3	2
	CO5	3	2	3	3	3	2	-	1	2	1	1	1	3	3	3	3
	CO6	3	3	2	3	3	2	1	1	2	1	1	1	3	3	3	2

Department:	**Department of Computer Science and Engineering**		
Course Name: **Web Designing Using HTML,CSS, Java Script & PHP Lab**	**Course Code ENCA151**	**L-T-P**	**Credits**
		0-0-2	1
Type of Course:	Major		
Pre-requisite(s), if any: Basic computer skills and familiarity with HTML and CSS are recommended pre-requisites for this syllabus.			

Course Outcomes

COs	Proficiency in HTML, CSS, JavaScript, and PHP.
CO 1	Ability to Design and Implement Interactive Web Elements.
CO 2	Problem-solving and Critical Thinking Skills.
CO 3	Application of Web Development Concepts
CO 4	Effective Collaboration and Communication Skills.

Ex. No	Experiment Title	Mapped CO/COs
1	Creating a Basic HTML Page: Learn to create a simple HTML page with headings, paragraphs, and basic elements.	CO 2
2	CSS styling sheets: Create a web page to differentiate the use of internal, external and Embedded Styling sheets.	CO 1
3	Styling with CSS: Apply CSS styles to enhance the appearance of HTML elements, such as fonts, colors, and backgrounds.	CO 1
4	Designing a Web Page: Working with Frames.	CO 1
5	Designing a Web Page: Working with Images and Hyperlinks.	CO 1, CO 3
6	Create HTML table: Create a Table using HTML and CSS properties like border, outline, and margin.	CO 1, CO 3
7	Create Lists: Create a webpage using lists in HTML and CSS properties.	CO 1, CO 3

8	Responsive Web Design: Design a responsive webpage that adjusts its layout based on different screen sizes using media queries.	CO 1, CO 3
9	Image Gallery: Develop an image gallery using HTML and CSS, with features like thumbnails and light box effects.	CO 1
10	Create Form: Design a form in a web page using HTML & CSS properties.	CO 1, CO 3
11	Form Validation: Use JavaScript to validate form inputs, such as required fields, email format, and password strength.	CO 1, CO 3
12	Implementing data types, variables, expression, and operator in JavaScript.	CO 1, CO 3
13	Use of conditional statements & looping statements in Java Script.	CO 1
14	Write a JavaScript Program to check whether the given positive number is a multiple of 3.	CO 1, CO 3
15	JavaScript Operators: Write a JavaScript program to calculate multiplication and division of two numbers.	CO 3,CO 4
16	JavaScript Arrays: Write a JavaScript program to compute the sum of elements of given array of integers.	CO 3,CO 4
17	Dropdown Menus: Design dropdown menus using HTML, CSS, and JavaScript to provide a hierarchical navigation structure.	CO 1, CO 3
18	Introduction to PHP: Learn the basics of PHP and create a simple PHP script to display dynamic content on a webpage.	CO 3,CO 4
19	Database Connectivity: Connect PHP with a MySQL database to retrieve and display data on a webpage.	CO 3,CO 4
20	User Registration and Login System: Implement a user registration and login system using PHP and MySQL.	CO 3,CO 4
21	Addition Operations: Perform addition operations using PHP and MySQL to manage database records.	CO 3,CO 4
22	Retrieve/View Operations: Perform Retrieve/View Operations using PHP and MySQL to manage database records.	CO 1, CO 3
23	Delete Operation: Perform delete operations using PHP and MySQL to manage database records.	CO 3,CO 4
24	Update Operation: Perform update operations using PHP and MySQL to manage database records.	CO 3,CO 4
25	Content Management System (CMS): Build a simple CMS using PHP and MySQL to manage website content dynamically.	CO 3,CO 4

26	Website Deployment: Learn how to deploy a web project on a server, configure domain and hosting settings, and make the website live on the internet.	CO 3, CO 4

Department	**Department of Computer Science and Engineering**		
Course Name: Clean Coding with Python	**Course Code**	**L-T-P**	**Credits**
	ENSP101	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s), if any: NA	**NA**		

Course Outcomes

CO1	Understand Python syntax and semantics and be fluent in the use of Python flow control and Functions.
CO2	Implement Python programs using core data structures like Lists, Dictionaries, and use of Strings Handling methods.
CO3	Apply Machine Learning Algorithms to real-world problems.
CO4	Interpretation of Data, Data Handling and Use Cases.

Brief Syllabus:

Python is a language with a simple syntax and a powerful set of libraries. It is an interpreted language, with a rich programming environment, including a robust debugger and profiler. While it is easy for beginners to learn, it is widely used in many scientific areas for data exploration. This course is an introduction to the Python programming language for students without prior programming experience. This course covers data types, control flow, object-oriented programming, and graphical user interface-driven applications. The examples and problems used in this course are drawn from diverse areas such as text processing, simple graphics creation and image manipulation, HTML and web programming, and genomics.

UNIT WISE DETAILS

Introduction Clean Coding

Unit Number: 1

Content Summary:

What is Bad Code? What is Clean Code? Purpose of Clean Code, Thought of experienced programmers, Meaningful Names, Intention Revealing Names, Make Meaningful Distinctions, Use Pronounceable Names , Avoid Encodings and Mental Mappings , Difference between smart and professional programmer , Class and Method Names, Function Size Matters, Blocks and Indenting
Do only one thing within a function, One level of abstraction per function, Use Descriptive Names
Function Arguments, Advantages of Having Less Arguments, Command Query Separation, Prefer Exceptions to Returning Error Codes , Extract Try/Catch Blocks ,Error Handling Is One Thing

Unit Number: 2 Introduction to Python

Content Summary:
What is Python?, Advantages and disadvantages, Downloading and installing, Which version of Python, Running Python Scripts, Using the interpreter interactively, Using variables, String types: normal, raw and Unicode String operators and expressions, Math operators and expressions, Writing to the screen, Reading from the keyboard, Indenting is significant, The if and elif statements, While Loops, Using List, Dictionaries, Using the for statement, Opening, reading and writing a text file, Using Pandas, the python data analysis library and data frames, Grouping, aggregating and applying, merging and joining, Dealing with syntax errors, Exceptions, Handling exceptions with try/exception.

Unit Number: 3 Data Handling and Use Cases

Content Summary:
RE Pattern Matching, Parsing Data, Introduction to Regression, Types of Regression, Use Cases, Exploratory data analysis, Correlation Matrix, Visualization using Matplotlib, Implementing linear regression.

Unit Number: 4 Advance Concepts

Content Summary:
Machine Learning - Algorithm Algorithms – Random forest Super vector Machine Random Forest Build your own model in Python Comparison between random forest and decision tree

***Self-Learning Components:**
- Object-oriented programming concepts,
- Numpy
- File Handling
- Jupyter Notebook
- PyCharm

Reference Books:
1. IBM Material

Assessment & Evaluation

Components	Assignment	Mid Term Examination	Attendance	End Term Examination
Weightage (%)	20	20	10	50

Program and course Mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENSP101/Clean Coding with Python	CO1	1	2	3	2	-	-	-	-	2	-	-	2	2	-	2	-
	CO2	1	2	3	2	2	-	-	-	2	-	-	2	2	-	2	-
	CO3	1	2	3	2	2	-	-	-	2	-	-	2	3	3	2	-
	CO4	1	2	3	2	2	-	-	-	2	-	-	2	3	3	2	-

Department:	**Department of Computer Science and Engineering**		
Course Name: Clean Coding with Python Lab	**Course Code**	**L-T-P**	**Credits**
	ENSP151	0-0-2	1
Type of Course:	Minor		
Pre-requisite(s), if any: NA			

Defined Course Outcomes

COs	
CO 1	Develop solutions to simple computational problems using Python programs.
CO 2	Solve problems using conditionals and loops in Python. Develop Python programs by defining functions and calling them.
CO 3	Implement Python lists, tuples, and dictionaries for representing compound data.
CO 4	Implementation of Machine Learning Algorithms.

Ex. No	Experiment Title	Mapped CO/COs
1	Develop programs to understand the control structures of python	CO 1
2	Develop programs to implement list	CO 3
3	Develop programs to implement Dictionary	CO 3
4	Develop programs to implement tuples	CO 3
5	Develop programs to implement function with stress on scoping	CO 2
6	Develop programs to implement classes and objects	CO 3
7	Develop programs to implement exception handling.	CO 1
8	Develop programs to implement linear search and binary search.	CO 2
9	Develop programs to implement insertion sort	CO 2
10	Develop programs to implement bubble sort.	CO 2
11	Develop programs to implement quick sort.	CO 2
12	Develop programs to implement heap sort.	CO 2

Department:	**Department of Computer Science and Engineering**		
Course Name: Essentials of Software Engineering	**Course Code**	**L-T-P**	**Credits**
	ENCA103	**3-1-0**	**4**
Type of Course:	Major		
Pre-requisite(s), if any: NA			

CO1	Identify and describe different software development methodologies.
CO2	Summarize the stages and activities involved in the software development life cycle.
CO3	Analyze and evaluate software requirements to identify potential risks and constraints.
CO4	Evaluate software documentation and code quality against industry standards.
CO5	Design and develop a complex software system that meets specified requirements.

Brief Syllabus:
This course covers the fundamentals of software engineering, including understanding system requirements, finding appropriate engineering compromises, effective methods of design, coding, and testing, team software development, and the application of engineering tools, Requirements Engineering technique, Development of UML Diagrams, Software Architecture and Design patterns, Software Testing- Black Box and White Box, Developing Test cases using Equivalence and Boundary value partitioning techniques, Test Driven Development with Junit in Eclipse, Software Refactoring.

UNIT WISE DETAILS

Unit Number: 1 Introduction to Models and SRS **No. of hours: 8**

Content Summary:
Introduction to Software Engineering: The evolving role of software, changing nature of software, Software Crisis, Software Processes & Characteristics.
Process models: The Waterfall model, Agile model, Evolutionary process model, Spiral model.
Software Requirements analysis & specifications: Requirement engineering, requirement elicitation techniques, Requirements analysis using DFD, ER Diagrams, Requirement documentation, Nature of SRS, Characteristics & organization of SRS.

Unit Number: 2 Software Metrics and System Design **No. of hours: 12**

Content Summary:
Software Metrics: Size Metrics like LOC, Token Count, Function Count, Design Metrics, Data Structure Metrics, Information Flow Metrics. Cost Estimation Models: COCOMO, COCOMO-II.

System Design: Design Concepts, design models for architecture, components, data, and user interfaces; Problem Partitioning, Abstraction, Cohesiveness, Coupling, Top-Down, and Bottom-Up design approaches; Functional Versus Object Oriented Approach, Design Specification.

Unit Number: 3 Unified Modeling Language and Software Reliability No. of hours: 10

Content Summary:

Unified Approach and Unified Modeling Language: The Unified Approach: Layered Approach to OO Software Development, UML: UML Diagrams for Structure Modeling, UML Diagrams for Behavior Modeling, UML Diagram for Implementation and deployment modeling.
Software Reliability: Importance, Hardware Reliability & Software Reliability, Failure and Faults, Reliability Models, Basic Model, Logarithmic Poisson Model, Software Quality Models, CMM & ISO 9001.

Unit Number: 4 Software Testing and Maintenance No. of hours: 10

Content Summary:

Software Testing: Testing process, Design of test cases, Functional testing: Boundary value analysis, Equivalence class testing, Decision table testing, Cause effect graphing, Structural testing, Path Testing, Data flow, and mutation testing, Unit Testing, Integration and System Testing, Debugging, Alpha & Beta Testing, Testing Tools & Standards.
Software Maintenance: Management of Maintenance, Maintenance Process, Maintenance Models, Regression Testing, Reverse Engineering, Software Re-engineering.

Text Books

1. K. K. Aggarwal & Yogesh Singh, “Software Engineering”, New Age International.
2. R. S. Pressman, “Software Engineering – A practitioner’s approach”, McGraw Hill
3. W.S. Jawadekar, “Software Engineering – Principles and Practices”, McGraw Hill

Reference Books/Materials

1. Stephen R. Schach, “Classical & Object-Oriented Software Engineering”, IRWIN, TMH.
2. James Peter, W. Pedrycz, “Software Engineering: An Engineering Approach”, John Wiley & Sons.
3. I. Sommerville, “Software Engineering”, Addison Wesley.
4. K. Chandrasehakhar, “Software Engineering & Quality Assurance”, BPB.

Program and Course Outcome Mapping

Course Code and Title		**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
ENCA103 Essentials of Software Engineering	**CO1**	3	2	3	-	-	-	-	-	2	2	2	1	1	2	1	2
	CO2	2	3	3	-	-	-	-	2	2	2	2	-	2	3	2	2
	CO3	2	3	3	1	1	-	2	-	2	-	-	-	2	2	3	2
	CO4	3	2	3	-	-	-	-	2	3	-	2	-	2	2	1	2
	CO5	3	3	3	-	-	-	-	2	2	-	3	1	3	3	3	2

Department:	**Department of Computer Science and Engineering**		
Course Name: Essentials of Software Engineering Lab	**Course Code**	**L-T-P**	**Credits**
	ENCA153	**0-0-1**	**2**
Type of Course:	Major		
Pre-requisite(s), if any: NA			

Course Outcomes

CO 1	Apply knowledge of software engineering principles and concepts to formulate clear problem statements.
CO 2	Apply software engineering lifecycle models and methodologies to develop and maintain software systems.
CO 3	Design software development processes that align with technical understanding and meet specified requirements.
CO 4	Analyze software requirements using appropriate modeling techniques and tools.
CO 5	Generate test case specifications and implement test cases based on given software requirements.

Ex. No	**Experiment Title**	**Mapped CO/COs**
1	Student Result Management System	
2	Library management system	
3	Inventory control system	
4	Accounting system	
5	Fast food billing system	
6	Bank loan system	
7	Blood bank system	
8	Railway reservation system	
9	Automatic teller machine	
10	Video library management system	
11	Hotel management system	
12	Hostel management system	
13	E-ticking	
14	Share online trading	
15	Hostel management system	
	Complete the following tasks for any five mentioned topics from the above list.	
1	Write the complete problem statement	CO1
2	Write the software requirement specification document	CO1, CO3
3	Draw the entity relationship diagram	CO2, CO4
4	Draw the data flow diagrams at level 0 and level 1	CO2, CO4
5	Draw a use case diagram	CO2, CO4
6	Draw an activity diagram of all use cases.	CO2, CO3

7	Draw a state chart diagram of all use cases	CO2. CO3
8	Draw a sequence diagram of all use cases	CO2, CO3
9	Draw a collaboration diagram of all use cases	CO2, CO3
10	Assign objects in sequence diagram to classes and make class diagram	CO2, CO3
11	Create test cases for the testing of the modules	CO1, CO5

EVEN SEMESTER (II)							
S. NO	**COURSE CODE**	**COURSE TITLE**	**Course Type**	**L**	**T**	**P**	**C**
1	ENCA102	Fundamentals of Object Oriented Programming using C++	Major	3	1	-	**4**
2	ENCA104	Discrete Structure	Major	3	1	-	**4**
3	ENSP102	Overview of AI, Data Science, Ethics and Foundation of Data Analysis	Minor	4	0	0	**4**
4	SEC039	R Programming for Data Science and Data Analytics Lab	SEC	0	0	4	**2**
5	ENCA152	Object Oriented Programming Lab using C++	Major	-	-	2	**1**
6	ENSP152	Overview of AI, Data Science, Ethics and Foundation of Data Analysis Lab	Minor	0	0	2	**1**
7		***Open Elective -1***	Open Elective	3	-	-	**3**
8	ENCA202	Extention Activities(community engagement service)	VAC II	2	-	-	**2**
		TOTAL		**15**	**2**	**8**	**21**

Department:	Department of Computer Science and Engineering		
Course Name: Fundamentals of Object-Oriented Programming using C++	Course Code	L-T-P	Credits
	ENCA102	3-1-2	4
Type of Course:	Major		
Pre-requisite(s), if any: Basics of C programming			

CO1	Understand object-oriented programming concepts.
CO2	Applying the concepts of object-oriented paradigm (Classes, Objects, inheritance, polymorphism etc.) for designing solution of a given programming problem
CO3	Developing applications that can manipulate data stored in files
CO4	Developing applications by considering all possible scenarios thereby employing appropriate exception handling.

Brief Syllabus:
The objective of this course is to introduce object-oriented programming. To explore and implement the various features of OOP such as inheritance, polymorphism, Exceptional handling using programming language C++. After completing this course student can easily identify the basic difference between the programming approaches like procedural and object oriented.

UNIT WISE DETAILS

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

Content Summary:
Procedure Oriented and Object-Oriented Approach. Basic Concepts: Objects, classes, Principals like Abstraction, Encapsulation, Inheritance and Polymorphism. Dynamic Binding, Message Passing. Characteristics of Object-Oriented Languages, Functions, Returning values from functions, Data Types

Unit Number: 2 **CLASSES AND OBJECTS** **No. of hours: 10**

Content Summary:
Abstract data types, Object & classes, attributes, methods, C++ class declaration, Local Class and Global Class, State identity and behaviour of an object, Local Object and Global Object, Scope resolution operator, Friend Functions, Inline functions, Constructors and destructors,

instantiation of objects, Types of Constructors, Static Class Data, Array of Objects, Constant member functions and Objects, Memory management Operators.

Unit Number: 3 **INHERITANCE & POLYMORPHISM** **No. of hours: 12**

Content Summary:

Inheritance, Types of Inheritance, access modes – public, private & protected, Abstract Classes, Ambiguity resolution using scope resolution operator and Virtual base class, Aggregation, composition vs classification hierarchies, overriding inheritance methods, Constructors in derived classes, Nesting of Classes

Polymorphism, Type of Polymorphism – Compile time and runtime, Function Overloading, Operator Overloading (Unary and Binary) Polymorphism by parameter, Pointer to objects, this pointer, Virtual Functions, pure virtual functions.

Unit Number: 4 **STRINGS AND EXCEPTION HANDLING** **No. of hours: 10**

Content Summary:

Manipulating strings, String Manipulation Functions, formatted and Unformatted Input output. Exception handling, reshowing exception, Exception Handling Techniques

***Self-Learning Components*:**

Students should explore Platforms like LeetCode, HackerRank for C++.

Students can refer the following courses as per the Open Source University Curriculum

1. Introduction to C++" and "C++ Programming for C Programmers" offered by edX
   "C++ Programming for Beginners," and "Learn Advanced C++ Programming." offered by Udemy

Reference Books:

1. E. Balagurusamy ,“Object Oriented Programming with C++”, Mc Graw Hill,6th Edition,2013.
2. Schildt Herbert, “C++: The Complete Reference”, Wiley DreamTech, 2005.Parasons, “Object Oriented Programming with C++”, BPB Publication, 1999.
3. Steven C. Lawlor, “The Art of Programming Computer Science with C++”, Vikas Publication, 2002.
4. Yashwant Kanethkar, “Object Oriented Programming using C++”, BPB, 2004

Program and Course Outcome Mapping

Course Code and Title		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENCA102 Fundamentals of Object-Oriented Programming using C++	CO1	1	-	3	-	3	-	-	-	2	2	1	2	3	2	2	1
	CO2	1	-	3	-	3	-	-	-	2	2	1	2	2	3	2	1
	CO3	1	-	3	2	3	-	2	-	2	3	1	2	2	3	2	1
	CO4	1	-	3	2	3	-	2	-	2	3	1	2	2	3	2	1

Department:	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Object Oriented Programming Lab using C++	**ENCA152**	3-1-2	4
Type of Course:	Major		
Pre-requisite(s), if any: Basics of C programming			

Ex. No	Experiment Title	Mapped CO/COs
1	Write a program for Functions with default arguments	
2	Simple Classes for understanding objects, member functions and Constructors i.Classes with primitive data members	
3	Write a program for Classes with constant data members, Classes with static member functions	
4	Write a program for Classes with pointers as data members – String Class	
5	Write a program for Classes with arrays as data members	
6	Implementation of Call by Value, Call by Address and Call by Reference	
7	Write a Program to illustrate New and Delete Keywords for dynamic memory allocation	
8	Write a Program Containing a Possible Exception. Use a Try Block to Throw it and a Catch Block to Handle it Properly.	
9	Project 1: interactive Basic Calculator: Create a calculator that accepts two numbers and an operator (+, -, /, *,&, <,>,// etc) using keyboard. Depending on operator, calculator must calculate the appropriate answer	
10	Write a Program to Demonstrate the Catching of All Exceptions.	
11	Write a program fir passing object as argument to a function with help of a program to add marks of two students in two different subjects respectively. Marks of first student in “sub1” should be added with marks of second student in “sub1” and respectively for marks of “sub2” added for both students and then displayed.	
12	Write a program to illustrate the concept of one class with two objects by taking student data.	
13	Write a program to show the relationship of class and object to display roll no., grade and fee paid by student.	
14	Write a program to define the member function outside and inside the class.	
15	Write a program to read and display the information of N persons to illustrate the concept of array of objects.	
16	Write a program to add two numbers to illustrate the use of friend function.	

17	Write a program to assign and copy values to illustrate the concept of parametrized and copy constructor.	
18	Write a program to show the order of constructor and destructor.	
19	Write a program to add two numbers using binary operator overloading.	
20	Write a program to illustrate the assignment operator overloading.	
21	Sample Programs using inheritance in and accessing objects of different derived classes (a) Write a program to compute the marks explaining the concept of multiple inheritance.	
22	Write a program to find the factorial of a number using inheritance	
23	Sample Programs using polymorphism and virtual functions (using pointers) (a) Write a program to find the volume of cylinder and cuboid using function overloading. (b) Write a program to reverse a string using pointers.	
24	Write a program to explain the relationship of inheritance and virtual function.	
25	Project2: Create Tic Tac Toe game using C++ concepts	CO4
26	Project 3: Quiz Game: Design a quiz game program where users can answer multiple-choice questions from various topics. The program should keep track of the score and provide feedback on the user's performance.	

Department:	**Department of Computer Science and Engineering**		
Course Name: **Discrete Mathematics**	**Course Code**	**L-T-P**	**Credits**
	ENCS203	3-1-0	4
Type of Course:	Programme Core		
Pre-requisite(s), if any: Basic of Mathematics			

CO1	Understand foundational concepts: Gain a solid understanding of fundamental concepts in discrete mathematics, including logic, sets, relations, and functions
CO2	Express proficiency in logical reasoning and constructing mathematical proofs using various proof techniques such as direct proofs, proof by contradiction, and mathematical induction.
CO3	Determine methods to Explore various discrete structures, such as sets, sequences, functions, relations, and formal languages. Understand the properties and applications of these structures.
CO4	Identify and develop problem-solving skills by applying discrete mathematics concepts to solve mathematical problems and real-world scenarios. Enhance logical thinking and analytical reasoning abilities.
CO5	Articulate real-world applications of discrete mathematics in computer science, cryptography, network analysis, optimization problems, scheduling, and decision-making.

Brief Syllabus:
This course will discuss fundamental concepts and tools in discrete mathematics with emphasis on their applications to computer science. Topics include logic and Boolean circuits, sets, functions, relations, deterministic algorithms and randomized algorithms, analysis techniques based on counting methods and recurrence relations, trees and graphs etc.

UNIT WISE DETAILS

Unit Number: 1 **Propositional Logics & Relations** **No. of hours: 12**

Mathematical Logic: Introduction to Mathematical Thinking, Propositional and Predicate Logic, Propositional Equivalences, Sets, Binary Relation, Equivalence Relation, Logical operations, Conditional Statements, Tautologies, Contradictions, Logical Equivalence, The use of Quantifiers, Normal Forms, Predicates and Quantifiers, Nested Quantifiers, Rules of Inference. Sets **and Relations**: Set Operations, Representation, and Properties of Relations & Functions, Equivalence Relations, Partially Ordering.

Unit Number: 2 **Title: Counting, Mathematical Induction, and Discrete Probability** **No. of hours: 12**

Basics of Counting, Pigeonhole Principle, Permutations and Combinations, Inclusion-Exclusion Principle, Mathematical Induction, Probability, Bayes' Theorem, Discrete Probability Theory, Discrete Structures in Computing, Counting Principles, Permutations and Combinations, Probability Theory, Discrete Random Variables, Discrete Optimization - Optimization Problems and Algorithms, Linear Programming, Integer Programming, Algebraic Structures - Groups (Definition, Properties, Subgroups, Cyclic Groups), Rings (Definition, Properties, Integral Domains, Fields), Isomorphisms and Homomorphisms, Counting and combinatorics.

Unit Number: 3 **Title: Group Theory & Discrete Probability** **No. of hours: 8**

Groups, Subgroups, Semi Groups, Product and Quotients of Algebraic Structures, Isomorphism, Homomorphism, Automorphism, Rings, Integral Domains, Fields, Applications of Group Theory, Combinatorial optimization: basic concepts and algorithms, Sample spaces, events, and probability axioms, Conditional probability and Bayes' theorem.

Unit Number: 4 **Title: Graph Theory** **No. of hours: 8**

Simple Graph, Multigraph, Weighted Graph, Paths and Circuits, Shortest Paths in Weighted Graphs, Eulerian Paths and Circuits, Hamiltonian Paths and Circuits, Planner graph, Graph Coloring, Bipartite Graphs, Trees and Rooted Trees, Prefix Codes, Tree Traversals, Spanning Trees and Cut-Sets, digraphs, Graph Coloring, Euler's formulae, Graph Theory, Networks and Flows.

***Self-Learning Components**:

Topics (with book references):

1. Applications of Graph Coloring: Timetable Scheduling ("Discrete Mathematics and Its Applications" by Kenneth H. Rosen: Chapter 10.3: Graph Coloring)
2. Network Analysis, Routing & Optimization, using graph theory.(Introduction to Graph Theory" by Richard J. Trudeau)
3. Combinatorial Optimization & Error Detection & correction using The Pigeonhole Principle ("Combinatorial Optimization: Algorithms and Complexity" by Christos H. Papadimitriou and Kenneth Steiglitz)
4. Scheduling and Task Prioritization, using Partial ordering. ("Introduction to Scheduling" by Yves Robert and Frederic Vivien)
5. Rules-based system and Algorithm design using conditional statements. (Chapters 10, 22, 23, of Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig).

Online Certification Courses for Discrete Mathematics (With Links):

1. Discrete Mathematics: https://www.coursera.org/learn/discrete-mathematics
2. Mathematics For Computer Science, https://ocw.mit.edu/courses/6-042j-mathematics-for-computer-science-fall-2010/
3. Introduction to Discrete Mathematics for Computer Science Specialization, https://www.coursera.org/specializations/discrete-mathematics
4. Discrete Math Series : Propositional Logic masterclass https://www.udemy.com/course/discretemathematics/
5. Master Discrete Mathematics: Sets, Math Logic, and More: https://www.udemy.com/course/master-discrete-mathematics/
6. Master Math by Coding in Python: https://www.udemy.com/course/math-with-python/
7. Discrete Mathematics for Computer Science in C, Java, Python: https://www.udemy.com/course/discrete-mathematics-and-its-applications/
8. Discrete Mathematics - Complete Course: https://www.udemy.com/course/discrete-mathematics-complete-course/
9. Discrete Optimization: https://www.coursera.org/learn/discrete-optimization
10. Introduction to Discrete Mathematics for Computer Science Specialization: https://www.coursera.org/specializations/discrete-mathematics

NPTEL Lecture Links for Discrete Mathematics (With Links):

1. Discrete Mathematics _ IIITB, IIIT Bangalore, Prof. Ashish Choudhury: https://nptel.ac.in/courses/106108227

2. Discrete Mathematics, IIT Ropar: https://nptel.ac.in/courses/106106183

Reference Books of Discrete Mathematics:

1. Elements of Discrete Mathematics, C. L Liu, McGraw-Hill Inc, 1985. Applied Combinatorics, Alan Tucker.
2. Concrete Mathematics, Ronald Graham, Donald Knuth, and Oren Patashnik, 2nd Edition - Pearson Education Publishers.
3. Combinatorics: Topics, Techniques, Algorithms by Peter J. Cameron, Cambridge University Press.
4. Topics in Algebra, I.N. Herstein, Wiley.
5. Kenneth H. Rosen, Discrete Mathematics and its Applications, Tata McGraw – Hill
6. Satinder Bal Gupta: A Text Book of Discrete Mathematics and Structures, University Science Press, Delhi.

E-Books of Discrete Mathematics (with Links):

1. Discrete Mathematics: An open Introduction, by Oscar Levin, 3rd Edition: https://discrete.openmathbooks.org/pdfs/dmoi-tablet.pdf
2. Lecture Notes on Discrete Mathematics, IITK, https://home.iitk.ac.in/~arlal/book/mth202.pdf
3. Mathematical Foundations And Aspects of Discrete Mathematics, Jean Gallier and Jocelyn Quaintance, https://www.cis.upenn.edu/~jean/discmath-root-b.pdf
4. Discrete Mathematics for Computer Science, Gary Haggard, John Schlipf, Sue Whitesides, https://www2.cs.uh.edu/~arjun/courses/ds/DiscMaths4CompSc.pdf
5. DISCRETE MATHEMATICS FOR COMPUTER SCIENCE, Herbert Edelsbrunner and Brittany Fasy, https://courses.cs.duke.edu/spring09/cps102/Lectures/Book.pdf
6. Discrete Mathematics and its Applications, Rosen, https://faculty.ksu.edu.sa/sites/default/files/rosen_discrete_mathematics_and_its_applications_7th_edition.pdf

Program and Course mapping

Course Code and Title	**PO**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
ENCS203/ Discrete Mathematics	CO1	3	2	2	-	2	-	2	-	-	-	-	2	3	2	2	1
	CO2	1	2	-	1	3	2	1	-	-	-	-	2	2	3	2	1
	CO3	-	-	-	1	3	-	2	-	-	3	-	2	2	3	2	1
	CO4	-	2	-	-	3	1	2	-	-	3	-	2	2	3	2	1
	CO5	-	2	-	-	3	-	2	-	-	3	-	2	2	3	2	1

Department:	**Department of Computer Science and Engineering**		
Course Name: Overview of AI, Data Science, Ethics and Foundation of Data Analysis	**Course Code**	**L-T-P**	**Credits**
	ENSP102	**4-0-0**	**4**
Type of Course:	Programme Core		
Pre-requisite(s), if any: Basic knowledge of Excel.			

COs	Statements
CO1	Outline the key concepts of AI and how AI has evolved
CO2	Identify the key concepts of Machine Learning and will be able to differentiate between key algorithms such as supervised learning and unsupervised learning
CO3	Distinguish key Data Science concepts such as structured and unstructured data, SQL, and NoSQL Database
CO4	Examine the process required the successfully execute a Machine Learning or Data Science project
CO5	Infer the large-scale data using Excel

Brief Syllabus:
The students will be studying about Introduction to Data Science, Natural Language, Machine generated Data, Graph-based or Network Data, Audio, Image, Video, and Streaming data. Also, six steps of data science processes define research goals, data retrieval, cleansing data, and correct errors as early as possible, integrating – combining data from different sources, transforming data, exploratory data analysis, Data modeling, model and variable selection, presentation, and automation would be taught to the students. Introduction to Machine Learning and Introduction to Data Analytics is also included in the syllabus.

UNIT WISE DETAILS

Unit Number: 1	**Introduction to Data Science**	**No. of hours: 8**

Content Summary: Defining Data Science and Big Data, Benefits and Uses of Data Science and Big Data, Facets of Data, Structured Data, Unstructured Data, Natural Language, Machine generated Data, Graph based or Network Data, Audio, Image, Video, Streaming data, Data Science
Process, Big data ecosystem and data science, distributed file systems, Distributed programming framework, data integration framework, machine learning framework, No SQL Databases, scheduling tools, benchmarking tools, system deployments

Unit Number: 2	**Data Science Processes**	**No. of hours: 8**

Content Summary: Six steps of data science processes define research goals, data retrieval, cleansing data, and correct errors as early as possible, integrating – combine data from different sources, transforming data, exploratory data analysis, Data modelling, model and variable selection, model execution, model diagnostic and model comparison, presentation and automation.

Unit Number: 3	Introduction to Machine Learning	No. of hours: 8

Content Summary: What is Machine Learning, Learning from Data, History of Machine Learning, Big Data for Machine Learning, Leveraging Machine Learning, Descriptive vs Predictive Analytics, Machine Learning and Statistics, Artificial Intelligence and Machine Learning, Types of Machine Learning – Supervised, Unsupervised, Semi-supervised, Reinforcement Learning, Types of Machine Learning Algorithms, Classification vs Regression Problem, Bayesian, Clustering, Decision Tree, Dimensionality Reduction, Neural Network and Deep Learning, Training machine learning systems

Unit Number: 4	Introduction to AI	No. of hours: 8

Content Summary: What is AI, Turing test, cognitive modeling approach, the law of thoughts, the relational agent approach, the underlying assumptions about intelligence, techniques required to solve AI problems, level of details required to model human intelligence, successfully building an intelligent problem, history of AI

Unit Number: 5	Introduction to Data Analytics	No. of hours: 4

Content Summary: Working with Formula and Functions, Introduction to Power BI & Charts, Logical functions using Excel, Analysing Data with Excel.

***Self-Learning Components**:

1. Artificial Intelligence Professional Program, https://online.stanford.edu/programs/artificial-intelligence-professional-program
2. Artificial Intelligence (AI), https://www.edx.org/course/artificial-intelligence-ai#!
3. 21st-Century Teaching & Learning: Data Science, https://online.stanford.edu/courses/xeduc315n-21st-century-teaching-learning-data-science
4. Artificial Intelligence: Principles and Techniques, https://online.stanford.edu/courses/xcs221-artificial-intelligence-principles-and-techniques
5. Data Visualization, https://online.stanford.edu/courses/cs448b-data-visualization

Reference Books:

1. Artificial Intelligence 3e: A Modern Approach Paperback – By Stuart J Russell & Peter Norvig; Publisher – Pearson
2. Artificial Intelligence Third Edition By Kevin Knight, Elaine Rich, B. Nair – McGraw-Hill
3. Artificial Intelligence Third Edition By Patrick Henry Winston – Addison-Wesley Publishing Company

Program and Course Outcome Mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENSP102 /Overview of AI, Data Science, Ethics and Foundation of Data Analysis	CO1	3	2	2	-	2	-	2	-	-	-	-	2	3	2	2	1
	CO2	1	2	-	1	3	2	1	-	-	-	-	2	2	3	2	1
	CO3	-	-	-	1	3	-	2	-	-	3	-	2	2	3	2	1
	CO4	-	2	-	-	3	1	2	-	-	3	-	2	2	3	2	1
	CO5	-	2	-	-	3	-	2	-	-	3	-	2	2	3	2	1

Department:	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Overview of AI, Data Science, Ethics and Foundation of Data Analysis	**ENSP152**	**0-0-2**	**1**
Type of Course:	**Programme Core**		
Pre-requisite(s), if any: Basic knowledge of Excel.			

COs	After performing the practical in Overview of AI, Data Science, Ethics and Foundation of Data Analysis lab, the students would be able to:
CO 1	Learn the basics of primary data structures.
CO 2	Perform various operations over these data structures.
CO 3	Learn the basics of Data Science & Analytics.
CO 4	Implement the basics of Data Science & Analytics.

Ex. No	Experiment Title	Mapped CO/COs
1	Write a program that uses functions to perform the following operations on singly linked list: i.Creation ii.Insertion iii.Deletion iv.Traversal	CO1, CO2
2	Write a program that uses functions to perform the following operations on doubly linked list: i.Creation ii.Insertion iii.Deletion iv.Traversal	CO1, CO2
3	Write a program that uses functions to perform the following operations on circular linked list: i.Creation ii.Insertion iii.Deletion iv.Traversal	CO1, CO2
4	Write a program that implement stack and its operations using: i.Arrays ii.Pointers	CO1, CO2
5	Write a program that implement queue and its operations using: i.Arrays	CO1, CO2

	ii.Pointers	
6	Write a program that implements the following sorting methods to sort a given list of integers in ascending order: i.Bubble sort ii.Selection sort iii.Insertion sort	CO1, CO2
7	Write a program that use both recursive and non-recursive functions to perform the following searching operations for a Key value in a given list of integers: i.Linear search ii.Binary search	CO1, CO2
8	Write a program to implement the tree traversal methods.	CO1, CO2
9	Write a program to implement the graph traversal methods.	CO1, CO2
10	Program on Comparative Analysis of Matching Algorithms	CO3, CO4
11	Analyzing the Impact of COVID-19 using Data Science: A Comprehensive Case Study	CO3, CO4
12	Program for Enhancing Data Visualization with Conditional Formatting	CO3, CO4
13	Exploring Pivot Tables in Data Science	CO3, CO4
14	Data Visualization with Power Map	CO3, CO4
15	Write a program for Data Science with Power BI	CO3, CO4
16	Write a program for Building Predictive Models in data science	CO3, CO4
17	Analyzing Sales Wallet Transactions using Data Science: Extracting Insights and Driving Business Growth	CO3, CO4
18	Harnessing the Power of Power Query in Data Science: Extract, Transform, and Analyze Data with Efficiency and Precision	CO3, CO4
19	"Exploring Correlation Methods in Data Science: Unveiling Relationships and Patterns in Complex Datasets	CO3, CO4

ODD SEMESTER (III)							
SNo	**Course Code**	**Course Title**	**Category**	**L**	**T**	**P**	**C**
1	ENCA201	Fundamentals of Data Structures	Major	3	1	0	4
2	ENCA203	Fundamentals of Java Programming	Major	3	1	0	4
3	ENSP205	Probabilistic Modelling and Reasoning	Minor	0	0	4	2
4	ENCA205	Fundamentals of Artificial Intelligence	Major	3	1	0	4
5	ENCA251	Fundamentals of Java Programming Lab	Major	0	0	2	1
6	ENCA253	Fundamentals of Data Structures Lab	Major	0	0	2	1
7		Open Elective -II	Open Elective	3	0	0	3
8		VAC III	VAC	2	0	0	2
9	AEC011	Life Skills for Professionals-I	AEC	3	0	0	3
		Summer Internship/Project	INT	0	0	0	2
		Total		17	3	8	26

Department:	**Department of Computer Science and Engineering**		
1. Course Name: Fundamentals of Data Structure	**Course Code**	**L-T-P**	**Credits**
	ENCA201	3-1-2	4
Type of Course:	Major		
Pre-requisite(s), if any: Basics of Computer Programming			

COs	Statements
CO1	Evaluate the efficiency of different data structures in terms of time and space complexity.
CO2	Implement a given Search problem (Linear Search and Binary Search).
CO3	Demonstrate an understanding of how data structures are implemented and their logical organization.
CO4	Design & implement the algorithm for Selection Sort, Bubble Sort, Insertion Sort, Quick Sort, Merge Sort, Heap sort. Compare their performance in term of Space and time complexity

Brief Syllabus:
Solving computational problems requires the knowledge of efficient data organization and the ability to make effective choices among multiple solutions. In this course, we will explore several fundamental data structures in computer science and learn to implement them. The course aims to teach the fundamentals of data structures, their design, implementation and effective use in problem solving approach. With the knowledge of data structures and practical experience in implementing them, students can become much more effective designer and developer. The course will start with the basic introduction of linear such as arrays, stack and queues as well as non-linear data structures such as trees and graphs. They will further proceed with the programming intensive task of implementing them.

UNIT WISE DETAILS

Unit Number: 1 Introduction to Data Structure **No. of hours: 8**

Content Summary:

Introduction to Data Structures: Definition of data structures and abstract data types, Static and Dynamic implementations, Examples and real-life applications; Arrays: ordered lists, representation of arrays in memory

Basic Analysis: Differences among best, average, and worst-case behaviors of an algorithm, Asymptotic analysis of upper and expected complexity bounds, Big O notation: formal definition and use, big omega and big theta notation, Complexity classes, such as constant, logarithmic, linear, quadratic, and exponential, Time and space trade-offs in algorithms.

Unit Number: 2 Stacks, Queues and Linked List **No. of hours: 12**

Content Summary:
Stacks: ADT Stack and its operation, Array based implementation of stacks, Examples: Infix, postfix, prefix representation, Conversions, Evaluation of postfix expression using stacks.

Queues: ADT Queue and its operation, Array based implementation of linear Queues, Circular Queues, Priority queues

Linked List: Definition, Components of linked list, Representation of linked list, Advantages and Disadvantages of linked list. Types of linked list: Singly linked list, doubly linked list, Circular linked list and circular doubly linked list. Operations on singly linked list: creation, insertion, deletion, search and display (based on the different position as specified by the user). Linked representation of Stacks & Queues.

Unit Number: 3 Trees and Graphs **No. of hours: 12**

Trees: Basic Terminology, Binary Trees and their representation, expression evaluation, Complete Binary trees, traversing binary trees, Searching, Insertion and Deletion in binary search trees.

Graphs: Terminology and Representations, Directed Graphs, Sequential representation of graphs, Adjacency matrices, Transversal Connected Component and Spanning trees, algorithms and their analysis.

Unit Number: 4 Sorting and Searching **No. of hours: 8**

Content Summary:
Sorting Algorithms: Introduction, insertion, selection, bubble, quick, merge, heap sort, algorithms and their analysis
Searching Algorithms: Straight Sequential Search, Binary Search (recursive & non–recursive Algorithm
***Self-Learning Components:**
1. Students should explore Platforms like LeetCode, HackerRank for Data structure
2. Students can refer the following courses as per the **Open-Source University Curriculum**
   - "Algorithms, Part I" by Robert Sedgewick and Kevin Wayne (available on Coursera)
   - "Algorithms, Part II" by Robert Sedgewick and Kevin Wayne (available on Coursera)

Reference Books:

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.s
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.

Program and Course Outcome Mapping

Course Code and Title		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENCA201 Fundamentals of Data Structure	CO1	3	3	3	-	-	-	-	-	-	3	2	-	3	2	3	-
	CO2	3	3	2	-	-	-	-	-	-	3	1	-	3	3	3	-
	CO3	3	3	3	-	-	-	-	-	-	3	1	-	3	-	2	-
	CO4	3	3	3	-	-	-	-	-	-	3	1	-	3	-	3	-

Department:	**Department of Computer Science and Engineering**		
Course Name: Fundamentals of Data Structure LAB	**Course Code**	**L-T-P**	**Credits**
	ENCA253	3-1-2	4
Type of Course:	Major		
Pre-requisite(s), if any: Basics of Computer Programming			

COs	List of CO's
CO 1	Apply the concepts learned of operators, if-else, loops and arrays to java-based application development.
CO 2	Demonstrate the use of various types of inheritances, polymorphisms, class objects, inheritances, packages and other concepts to basic and complex java programming problems.
CO 3	Demonstrate graphical applications based on java applets, swings and event handling
CO 4	Apply knowledge of event handling and AWT controls to create some new dynamic graphical applications.

Ex No	Experiment Title	Mapped CO/COs
1	Sample Programs using Objects and classes, Variable Types, Modifier Types, operators, Loops Decision Making, Strings and Arrays, (a) WAP to display "Hello, it's a first program in java". (b) WAP to find sum of two integers taken as input from user at runtime. (c) WAP to find sum of two float numbers taken as command line arguments (d) WAP to find changed case of entered character. (e) WAP to find maximum of 3 integer numbers taken as input from user at runtime.	CO1
2	Sample Programs using Inheritance, Overriding, Polymorphism, Interfaces, Packages a. WAP in java to illustrate the concept of interfaces. b. Write a program in java to showcase uses of super keyword	CO1
3	Sample Programs using exception handling and threads	CO2

	a) Write a program to demonstrate the use of nesting of try-catch block b) WAP in java to illustrate the concept of using multiple catch clauses to handle different types of exceptions. c) WAP in java to create a user defined Exception and throw it explicitly.	
4	Sample Programs using event handling and AWT controls	CO1
5	Sample Programs using swings Write an applet which will display “HAPPY”and “DEEPAVALI” as: The word “HAPPY” will roll from top to bottom and “DEEPAVLI” from bottom to “top” . Both will run at the same speed and stop simultaneously at the center of the applet.	CO3
6	WAP in java to create a frame with various AWT controls (like choice, list, TextField and Buttons) and handle the events thrown by them.	CO3
7	WAP in java to create a frame with AWT controls (like label, push buttons, Checkbox, Checkbox Group) and handle various events generated by them.	CO4
8	WAP to create a package as MyPack having a class with three methods: max, fact and show. Use it in other folder with setting classpath and without setting class path.	CO2
9	WAP to create a frame and illustrate the concept of using an adapter class in place of interfaces for handling various mouse events generated over frame window.	CO3
10	Write a program to display “hello” in different color where user clicks left mouse button and “world” where right mouse button is clicked. Use black background.	CO2
11	a) Demonstrate thread using Thread class and Runnable interface b) Demonstrate various thread methods using a program	CO3
12	Write a java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea( ) that prints the area of the given shape.	CO4
13	(a) WAP to create class with “name” as String and “age” as integer data members. The class should have two methods to take input from user and display the data. (b) WAP to find factorial of a number using class and object.	CO3
14	Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even,	CO4

	second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.	
15	Create an Frame with one single button with caption “Click”. On clicking the button will open a new Frame with title “Factorial”. The frame will have two three controls :TextField, Label and button. On clicking button calculate the factorial entered in TextField control.	CO4
16	Project 1: Simple Calculator: Build a basic calculator application that performs arithmetic operations like addition, subtraction, multiplication, and division. You can add a user interface using Java Swing or JavaFX for a more interactive experience.	CO4
17	Project 2: Tic-Tac-Toe Game: Implement the classic Tic-Tac-Toe game where two players take turns marking X or O on a 3x3 grid. Allow players to play against each other.	CO4
18	Project 3: Quiz Application: Design a quiz application that presents multiple-choice questions to users and keeps track of their scores. Include features like a timer, question randomization, and a scoring system.	CO4
19	Project 4: Hangman Game: Create a Hangman game where players guess letters to uncover a hidden word. Include features such as displaying the word's progress, tracking incorrect guesses, and providing hints.	CO4

Department:	**Department of Computer Science and Engineering**		
Course Name: Fundamentals of Java Programming	**Course Code**	**L-T-P**	**Credits**
	ENCA203	**3-1-0**	**4**
Type of Course:	Programme Core		
Pre-requisite(s), if any: C++ Programming			

COs	Statements
CO1	Recognize features of object-oriented design such as encapsulation, polymorphism inheritance and composition of systems based on object identity.
CO2	Articulate re-usable programming components using Abstract Class, Interfaces and other permitted ways in packages.
CO3	Apply access control mechanism to safeguard the data and functions that can be applied by the object.
CO4	Design GUI applications using pre-built frameworks available in Java.

Brief Syllabus:

The objective is to impart programming skills used in this object-oriented language java. The course explores all the basic concepts of core java programming like object, classes, data types, features, operators, control structures, interfaces, packages, applets, AWT, Swings. The students are expected to learn it enough so that they can develop the basic applications as well as web solutions like creating applets etc.

11. UNIT WISE DETAILS

Unit Number: 1 Introduction to Java No. of hours: 12

Content Summary:
Concepts of OOP, Features of Java, How Java is different from C++, Environmental setup, Basic syntax, Objects and classes, Basic Data Types, Variable Types, Modifier Types, Basic operators, Loop Control, Decision Making, Strings and Arrays, Methods, I/O. Introducing classes, objects and methods: defining a class, adding variables and methods, creating objects, constructors.

Unit Number: 2 Arrays and Strings No. of hours: 8

Content Summary:
Classes: String and String Buffer classes, Wrapper classes: Basics types, using super, Multilevel hierarchy, abstract and final classes, Object class, Access protection, Inheritance, Overriding, Polymorphism, Abstraction, Encapsulation, Interfaces, Packages, Exploring java.util package.

Unit Number: 3 Exceptional Handling & Multithreading No. of hours: 12

Content Summary:
Exception Hierarchy, Exception Methods, Catching Exceptions, Multiple catch Clauses, Uncaught Exceptions Java's Built-in Exception. Creating, Implementing and Extending thread, thread priorities, synchronization suspending, resuming and stopping Threads, Multi-threading.

Unit Number: 4 Input/output Programming & File handling No. of hours: 8

Basics Streams, Byte and Character Stream, predefined streams, Reading and writing from console and files. Reading data from files using input streams, writing data to files using output streams.

***Self-Learning Components*:**
Students should explore Platforms like LeetCode, HackerRank for JAVA and JAVA IDE like eclipse, Netbeans etc.
Students can refer the following courses as per the Open-Source University Curriculum

1. "Java Programming Masterclass for Software Developers" on Udemy by Tim Buchalka
2. "Java Fundamentals: The Java Language" on Pluralsight by Jesse Liberty,

Reference Books:

1. Herbert Schildt, ―Java – The Complete Reference‖, Oracle Press.
2. Cay S. Horstmann, ―Core Java Volume – I Fundamentals‖, Pearson.

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENCA203/ Fundamentals of Java Programming	CO1	3	2	2	-	2	-	2	-	-	-	-	2	3	2	2	1
	CO2	1	2	-	-	3	-	1	-	-	-	-	2	2	3	2	1
	CO3	-	-	-	-	3	-	2	1	-	3	-	2	2	3	2	1
	CO4	-	-	-	-	3	-	2	-	-	3	-	2	2	3	2	1
	CO5	3	2	2	-	2	-	2	-	-	-	-	2	3	2	2	1

Department:	**Department of Computer Science and Engineering**		
Course Name: Fundamentals of Java Programming Lab	**Course Code**	**L-T-P**	**Credits**
	ENCA251	**3-1-0**	**4**
Type of Course:	Programme Core		
Pre-requisite(s), if any: C++ Programming			

Ex No	Experiment Title	Mapped CO/COs
1	Sample Programs using Objects and classes, Variable Types, Modifier Types, operators, Loops Decision Making, Strings and Arrays, a. WAP to display “Hello, it’s a first program in java”. b. WAP to find sum of two integers taken as input from user at runtime. c. WAP to find sum of two float numbers taken as command line arguments d. WAP to find changed case of entered character. e. WAP to find maximum of 3 integer numbers taken as input from user at runtime.	CO1
2	Sample Programs using Inheritance, Overriding, Polymorphism, Interfaces, Packages a. WAP in java to illustrate the concept of interfaces. b. Write a program in java to showcase uses of super keyword	CO1
3	Sample Programs using exception handling and threads a. Write a program to demonstrate the use of nesting of try-catch block b. WAP in java to illustrate the concept of using multiple catch clauses to handle different types of exceptions. c. WAP in java to create a user defined Exception and throw it explicitly.	CO2
4	Sample Programs using event handling and AWT controls	CO1
5	Sample Programs using swings Write an applet which will display “HAPPY” and “DEEPAVALI” as: The word “HAPPY” will roll from top to bottom and “DEEPAVLI” from bottom to “top” . Both will run at the same speed and stop simultaneously at the center of the applet.	CO3
6	WAP in java to create a frame with various AWT controls (like choice, list, TextField and Buttons) and handle the events thrown by them.	CO3
7	WAP in java to create a frame with AWT controls (like label, push buttons, Checkbox, Checkbox Group) and handle various events generated by them.	CO4

8	WAP to create a package as MyPack having a class with three methods: max, fact and show. Use it in other folder with setting classpath and without setting class path.	CO2
9	WAP to create a frame and illustrate the concept of using an adapter class in place of interfaces for handling various mouse events generated over frame window.	CO3
10	Write a program to display “hello” in different color where user clicks left mouse button and “world” where right mouse button is clicked. Use black background.	CO2
11	a. Demonstrate thread using Thread class and Runnable interface b. Demonstrate various thread methods using a program	CO3
12	Write a java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea( ) that prints the area of the given shape.	CO4
13	a. WAP to create class with “name” as String and “age” as integer data members. The class should have two methods to take input from user and display the data. b. WAP to find factorial of a number using class and object.	CO3
14	Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.	CO4
15	Create an Frame with one single button with caption “Click”. On clicking the button will open a new Frame with title “Factorial”. The frame will have two three controls :TextField, Label and button. On clicking button calculate the factorial entered in TextField control.	CO4
16	Project 1: Simple Calculator: Build a basic calculator application that performs arithmetic operations like addition, subtraction, multiplication, and division. You can add a user interface using Java Swing or JavaFX for a more interactive experience.	CO4
17	Project 2: Tic-Tac-Toe Game: Implement the classic Tic-Tac-Toe game where two players take turns marking X or O on a 3x3 grid. Allow players to play against each other.	CO4
18	Project 3: Quiz Application: Design a quiz application that presents multiple-choice questions to users and keeps track of their scores. Include features like a timer, question randomization, and a scoring system.	CO4
19	Project 4: Hangman Game: Create a Hangman game where players guess letters to uncover a hidden word. Include features such as displaying the word's progress, tracking incorrect guesses, and providing hints.	CO4

Department:	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
	ENSP205	0-0-4	2
Type of Course:	Probabilistic Modelling and Reasoning		
Pre-requisite(s), if any: Basic knowledge of Statistics			

Course Outcomes (COs)

- Help students understand the importance and implementation of various random sampling techniques
- Describe probability and various probability distributions such as normal distribution, beta, gamma, students -t, and bivariate distributions
- Introduce the concepts of estimation techniques that cover both point and interval estimation
- Teach the concepts of hypothesis testing, p-value, and Bayesian statistics

CO 1	Explain the data-gathering techniques
CO 2	Inspect the data using descriptive statistics
CO 3	Illustrate the probability and conditional probability concepts
CO 4	Distinguish between various probability distributions and analyze the data following different probability distributions
CO5	Solve inferential statistics problems using point and interval estimation techniques. Infer the statistical problems using hypothesis testing and p value

Brief Syllabus:

- Help students understand the importance and implementation of various random sampling techniques
- Describe probability and various probability distributions such as normal distribution, beta, gamma, students -t, and bivariate distributions
- Introduce the concepts of estimation techniques that cover both point and interval estimation
- Teach the concepts of hypothesis testing, p-value, and Bayesian statistics

UNIT WISE DETAILS

Unit Number: 1 Introduction to Statistics No. of hours: 8

Content Summary: Introduction to Statistics. Role of statistics in scientific methods, current applications of statistics
Scientific data gathering: Sampling techniques, scientific studies, observational studies, data management.
Displaying data on a single variable (graphical methods, measure of central tendency, measure of spread), displaying the relationship between two or more variables, a measure of association between two or more variables.

Unit Number: 2 Probability Theory No. of hours: 8

Content Summary: Sample space and events, probability, axioms of probability, independent events, conditional probability, Bayes' theorem.

Random Variables: Discrete and continuous random variables. Probability distribution of discrete random variables, binomial distribution, Poisson distribution. Probability distribution of continuous random variables, The uniform distribution, normal (Gaussian) distribution, exponential distribution, gamma distribution, beta distribution, t-distribution, χ'' distribution. Expectations, variance, and covariance. Probability Inequalities. Bivariate distributions.

Unit Number: 3 **Point Estimations** **No. of hours: 8**

Content Summary: Methods of finding estimators, method of moments, maximum likelihood estimators, Bayes estimators. Methods of evaluating estimators, mean squared error, best-unbiased estimator, sufficiency and unbiasedness

Interval Estimations: Confidence interval of means and proportions, Distribution free confidence interval of percentiles

Unit Number: 4 **Test of Statistical Hypothesis and P-values** **No. of hours: 8**

Content Summary: Tests about one mean, tests of equality of two means, tests about proportions, p-values, likelihood ratio test, Bayesian tests

Bayesian Statistics: Bayesian inference of discrete random variable, Bayesian inference of binomial proportion, comparing Bayesian and frequentist inferences of proportion, comparing Bayesian and frequentist inferences of mean

Univariate Statistics using Python: Mean, Mode. Median, Variance, Standard Deviation, Normal Distribution, t-distribution, interval estimation, Hypothesis Testing, Pearson correlation test, ANOVA F-test

about intelligence, techniques required to solve AI problems, level of details required to model human intelligence, successfully building an intelligent problem, history of AI

Self-Learning Components: mention 4-5 topics for students in bullet points

- Advanced topics on Statistics and probability from the reference books given
- Learn the concepts from https://learning.samatrix.io further
- Download different datasets from Github and practice the EDA, probability distributions
- Participate in Kaggle Competitions on Statistical data analysis

Please Note:

1)Students are supposed to learn the components on self-basis

2)Mention open-source tools/ new concepts/technologies that students will be required to learn and present through presentations in class

3) At least 5-10 % syllabus will be asked in end term exams from self-learning components

- Reference Books: Achim Klenke, (2014), Probability Theory A Comprehensive Course Second Edition, Springer, ISBN 978-1-4471-5360-3
- Christian Heumann, Michael Schomaker Shalabh (2016), Introduction to Statistics and Data Analysis with Exercises, Solutions and Applications in R, Springer International Publishing, ISBN 978-3-319-46160-1

Douglas C. Montgomery, (2012), Applied Statistics and Probability for Engineers, 5th Edition, , Wiley India, ISBN: 978-8-126-53719-8

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO1 1	PO1 2	PSO1	PSO2	PSO3	PSO 4
ENSP205/ Probabilistic Modelling and Reasoning	**CO1**	3	2	2	-	2	-	2	-	-	-	-	2	3	2	2	1
	CO2	1	2	-	-	3	-	1	-	-	-	-	-	2	3	2	3
	CO3	-	-	-	-	3	-	2	1	-	3	-	2	2	3	1	1
	CO4	-	-	-	-	3	-	2	-	-	3	-	-	2	3	2	3
	CO5	-	-	-	-	3	-	2	-	-	3	-	2	2	3	2	1

Department:	**Department of Computer Science and Engineering**		
Course Name: Fundamentals of Artificial Intelligence	**Course Code**	**L-T-P**	**Credits**
	ENCA205	**3-1-0**	**4**
Type of Course:	**Major**		
Pre-requisite(s), if any: NA			

COs	Statements
CO1	**Understand** will learn about intelligent agents, problem-solving, knowledge representation, machine learning, and other core areas of AI
CO2	**Express** proficiency in developing algorithms and models to solve complex problem and develop AI solutions that can automate tasks, improve efficiency, and make informed decisions.
CO3	**Determine** methods analyze data, identify patterns and trends, and make data-driven decisions.
CO4	**Identify** making decisions based on data and learning from patterns, which enhance critical thinking and decision-making skills.
CO5	**Articulate** to abstract complex problems, identify essential features, and generalize solutions across different scenarios.

UNIT WISE DETAILS

Unit Number: 1 Introduction **No. of hours: 4**

Definition of Intelligence, Knowledge, Artificial Intelligence; importance, real-time applications, Turing test, the importance of Artificial Intelligence in today’s era, Difference between Human Brain & Computer; Chinese Room Argument, Types of Knowledge, Knowledge Pyramid, Merits and Demerits of Artificial Intelligence.

Unit Number: 2 Brute Force & Heuristic Search Algorithms **No. of hours: 8**

Characteristics of AI Problems, Problem Representation Techniques, Declarative and Procedural Representation, Introduction to Brute Force Search: Breadth First Search & Depth First Search; Introduction to Heuristic Search: Hill Climbing, A* Algorithm, Best First Search.

Unit Number: 3 Introduction to Machine Learning **No. of hours: 8**

An introduction to Machine Learning, Definition of Machine Learning, Learning, Classification of Machine Learning, Supervised, Unsupervised, and Reinforcement Learning; 7 Types of Reasoning (With Definitions and Examples), Machine Learning Applications, Life Cycle of Machine Learning, Introduction to Fuzzy Logic, Linear regression, and classification, Decision trees, random forests.

Unit Number: 4 Fusion of AI with other technologies and case studies **No. of hours: 8**

Fusion of AI with IoT, Case Studies: Artificial Intelligence & Machine Learning, Definition: Expert Systems, Neural Networks, Natural Language Processing, Expert System Life Cycle, Futuristic trends in Artificial Intelligence & Machine Learning, Computer Vision

***Self-Learning Components:**

1. Case Studies of Fuzzy Logic ("Fuzzy Logic with Engineering Applications" by Timothy J. Ross)
2. Problems associated with Hill Climbing Algorithm ("Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig)
3. Monotonic Vs. Non-Monotonic Reasoning ("Artificial Intelligence: Foundations of Computational Agents" by David L. Poole and Alan K. Mackworth)
4. Case studies of Fusion of AI with IoT ("Artificial Intelligence for the Internet of Things" by Amita Kapoor)
5. Ethical Considerations in AI ("Ethics of Artificial Intelligence and Robotics" edited by Vincent C. Müller and Nick Bostrom)

Online Certification Courses for Fundamentals of Artificial Intelligence (With Links):

1. Stanford & Deeplearning.AI "Machine Learning Specialization", https://www.coursera.org/specializations/machine-learning-introduction
2. "IBM AI Engineering Professional Certificate", https://www.coursera.org/professional-certificates/ai-engineer
3. Artificial Intelligence A-Z™ 2023: Build an AI with ChatGPT4, https://www.udemy.com/course/artificial-intelligence-az/
4. Artificial Intelligence for Business, Solve Real World Business Problems with AI Solutions, https://www.udemy.com/course/ai-for-business/
5. "Artificial Intelligence: an Overview", https://www.coursera.org/specializations/artificial-intelligence-overview
6. TensorFlow Developer Certificate in 2023: Zero to Mastery, https://www.udemy.com/course/tensorflow-developer-certificate-machine-learning-zero-to-mastery/
7. IBM "AI Foundations for Everyone Specialization", https://www.coursera.org/specializations/ai-foundations-for-everyone
8. Introduction to Generative AI, https://www.coursera.org/learn/introduction-to-generative-ai
9. Artificial Intelligence: An Overview, https://www.coursera.org/learn/artificial-intelligence-an-overview

NPTEL Lecture Links for Fundamentals of Artificial Intelligence (With Links):

1. An Introduction to Artificial Intelligence By Prof. Mausam, IIT Delhi, https://onlinecourses.nptel.ac.in/noc22_cs56/preview
2. NPTEL Video Course : Artificial Intelligence (Prof. Deepak Khemani), https://www.digimat.in/nptel/courses/video/106106126/L01.html
3. NPTEL Video Course NOC:An Introduction to Artificial Intelligence, https://www.digimat.in/nptel/courses/video/106102220/L01.html
4. NPTEL Video Course : NOC:Artificial Intelligence: Knowledge Representation and Reasoning, https://www.digimat.in/nptel/courses/video/106106140/L01.html
5. NPTEL Video Course : NOC: Introduction to Machine Learning, Prof. Sudeshna Sarkar, https://www.digimat.in/nptel/courses/video/106105152/L01.html

K. R. Mangalam University, LMS Lecture Links for Fundamentals of Artificial Intelligence (With Links):

Fundamentals of Artificial Intelligence, by Dr. Amar Saraswat, K. R. Mangalam University, https://lms.krmangalam.edu.in/course/view.php?id=1769

Reference Books of Fundamentals of Artificial Intelligence:

1. Artificial Intelligence, by Elaine Rich, Kevin Knight, Shivashankar B Nair, Mc graw Hill Publications.
2. "Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig

E-Books of Fundamentals of Artificial Intelligence (with Links):

1. Artificial Intelligence, A Morden Approach, Third Edition, Stuart Russell and Peter Norvig, PRENTICE HALL SERIES IN ARTIFICIAL INTELLIGENCE, https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf

Program and Course Outcome Mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO1 1	PO1 2	PSO1	PSO2	PSO3	PSO 4
ENCA205/ Fundamentals of Artificial Intelligence	CO1	3	2	2	-	2	-	2	-	-	-	-	2	3	2	2	1
	CO2	1	2	-	-	3	-	1	-	-	-	-	-	2	3	2	3
	CO3	-	-	-	-	3	-	2	1	-	3	-	2	2	3	1	1
	CO4	-	-	-	-	3	-	2	-	-	3	-	-	2	3	2	3
	CO5	-	-	-	-	3	-	2	-	-	3	-	2	2	3	2	1

Department:	**Department of Computer Science and Engineering**		
Course Name: Life Skills for Professionals-I	**Course Code**	**L-T-P**	**Credits**
	AEC011	**3-0-0**	**3**
Type of Course:	**AEC**		
Pre-requisite(s) Basic life skills			

COs	Statements
CO1	Perform calculations related to number systems, percentages and averages, quickly and accurately.
CO2	Exhibit confidence in tackling multiple-choice questions, time-constrained tests and competitive examinations.
CO3	Demonstrate active listening techniques, including attentive listening and reflection
CO4	Articulate and speak with confidence and express ideas clearly and coherently.
CO5	Improve confidence and display open and positive non-verbal communication.

Brief Syllabus:

Through this comprehensive course, the learners will develop a solid foundation in communication skills, enabling them to express themselves confidently, listen actively, and build strong relationships in personal and professional contexts.

UNIT WISE DETAILS

Unit Number: 1 Communication: An Introduction No. of hours: 6

Definition, Nature and Scope of Communication, Importance and Purpose of Communication, Process of Communication, Types of Communication, Barriers to Communication, Essentials of Effective Communication

Unit Number: 2 Non-Verbal Communication No. of hours: 6

Personal Appearance, Gestures, Postures, Facial Expression, Eye Contacts, Body Language (Kinesics) Time language, Tips for Improving Non-Verbal Communication

Unit Number: 3 Basic number system No. of hours: 6

Divisibility, Unit digit, Last two digit, Remainder, Number of zero, Factor, LCM & HCF, Simplification, Mixture, Average, Ratio, and Partnership.

Unit Number: 4 **Number system** **No. of hours: 6**

Factor, LCM & HCF, Simplification, Mixture, Average, Ratio, and Partnership.

Unit Number: 5 **Time Management** **No. of hours: 6**

Time management strategies, setting goals, organizing, and planning ahead, Making the most of your time Dealing with distractions, Procrastination, and Avoiding distractions

*Self-Learning Components:

https://onlinecourses.nptel.ac.in/noc21_hs02/preview

Please Note:

1)Students are supposed to learn the components on self-basis

2) At least 5-10 % syllabus will be asked in end term exams from self-learning components

Reference Books:

1. Aggarwal, R. S. (2014). Quantitative aptitude (Revised edition).
2. Gladwell, M. (2021). Talking to strangers.
3. Scott, S. (2004). Fierce conversations.

Program and Course Outcome Mapping

Course Code and Title	**PO**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO 10**	**PO1 1**	**PO1 2**	**PSO1**	**PSO2**	**PSO3**	**PSO 4**
AEC011/ life Skills for Professionals-I	**CO1**	3	2	2	-	2	-	2	-	-	-	-	2	3	2	2	1
	CO2	1	2	-	-	3	-	1	-	-	-	-	-	2	3	2	3
	CO3	-	-	-	-	3	-	2	1	-	3	-	2	2	3	1	1
	CO4	-	-	-	-	3	-	2	-	-	3	-	-	2	3	2	3
	CO5	-	-	-	-	3	-	2	-	-	3	-	2	2	3	2	1

Department:	**Department of Computer Science and Engineering**		
Course Name: Summer Internship-II	**Course Code**	**L-T-P**	**Credits**
		0-0-0	2
Type of Course:	**INT**		
Pre-requisite(s), if any:			

The duration of the internship will be two weeks. It will be after the completion of 3rd Semester and before the commencement of Semester 5th Semester.

The following options can be opted for by the students:

1. Offline internship in industry - The student is supposed to produce a joining letter and relieving letter once the internship is over in case of an Offline internship in any industry.

2. Online internships – with organizations /institutions that are approved /supported/recommended by the All-India Council of Technical Education for Internship (like SWAYAM, NPTEL, Internshala etc.)

Report Submission and Evaluation Guidelines:

• Student must prepare a detailed report and submit the report. A copy of the report can be kept in the departments for record.

• Each student must be assigned a faculty as a mentor from the university and an Industry Expert as External Guide or Industry Mentor.

• The presentation by student for Internship/ project should in the presence of all students is desirable.

• Student should produce successful completion certificate in case of summer internship in industry.

Course Outcomes:

At the end of the course, students will be able to:

1. Get exposure to the industrial environment, which cannot be simulated in the classroom and hence creating competent professionals for the industry.

2. Get possible opportunities to learn, understand and sharpen the real time technical / managerial skills required at the job(s).

3. Gain experience in writing technical reports / projects and presentation of it.

4. Learn and gain exposure to the engineer’s responsibilities and ethics.

5. Understand the social, economic, and administrative considerations that influence the working environment of industrial organizations.

	EVEN SEMESTER (IV)						
SNo	Course Code	Course Title	Category	L	T	P	C
1	ENCA202	Fundamentals of Operating Systems	Major	4	-	-	4
2	ENCA204	Fundamentals of Database Management Systems	Major	3	1	0	4
3	ENSP212	Foundation of Machine Learning	Minor	4	0	0	4
4	ENCA252	Fundamentals of Operating System Lab	Major	-	-	2	1
5	ENSP262	Foundation of Machine Learning Lab	Minor	0	0	2	1
6	ENCS254	Fundamentals of Database Management Systems Lab	Major	0	0	2	1
7		Open Elective -III	Open Elective	3	0	0	3
8	AEC012	Life Skills for Professionals-II	AEC	3	0	0	3
9		VAC IV	VAC	2	0	0	2
10	ENSI252	Minor project-I		0	0	0	2
TOTAL				19	1	6	25

Department:	**Department of Computer Science and Engineering**		
Course Name: Fundamentals of Operating Systems	**Course Code**	**L-T-P**	**Credits**
	ENCA202	4-0-0	4
Type of Course:	MAJOR		
Pre-requisite(s), if any: Basics of programming			

Define Course Outcomes (CO)

COs	**Statements**
CO1	To outline various concepts and features of Operating systems.
CO2	Compare various operating systems with respect to characteristics and features
CO3	Implement algorithm of CPU Scheduling, Memory Scheduling and disk scheduling.
CO4	Apply appropriate memory and file management schemes
CO5	Illustrate various disk scheduling algorithms.

Brief Syllabus:
Operating systems course is intended as a general introduced to the techniques used to implement operating systems and related kinds of systems software. The topics covered will be functions and structure of operating systems, process management (creation, synchronization, and communication); processor scheduling; deadlock prevention, avoidance, and recovery; main-memory management; virtual memory management (swapping, paging, segmentation and page-replacement algorithms); control of disks and other input/output devices; file-system structure and implementation; and protection and security.

UNIT WISE DETAILS

Unit Number: 1 **Introduction to OS** **No. of hours: 6**

Introduction: Concept of Operating Systems, Generations of Operating systems, Types of Operating Systems, OS Services, System Calls, Layered System, Kernel, Types of Kernels (Monolithic/Macro Kernel and Micro Kernel), Virtual Machine.

Unit Number: 2 **Processes and Threads** **No. of hours: 12**

Processes: Definition, Process Relationship, Different states of a Process, Process State transitions, Process Control Block (PCB), Context switching.
Thread: Definition, Various states, Benefits of threads, Types of threads, Concept of multithreads.

Process Scheduling: Basic Concept, Type of Scheduling (Preemptive Scheduling, Non-preemptive Scheduling), Scheduling criteria: CPU utilization, Throughput, Turnaround Time, Waiting Time, Response Time.
Scheduling algorithms: Pre-emptive and Non-preemptive.

Unit Number: 3 Verification and Validation No. of hours: 12

Memory Management: Basic concept, Logical and Physical address map, Memory allocation: Contiguous Memory allocation – Fixed and variable partition–Internal and External fragmentation and Compaction; Paging.
Virtual Memory: Basics of Virtual Memory – Hardware and control structures – Locality of reference, Page fault, Working Set, Dirty page/Dirty bit– Demand paging, Page Replacement algorithms: Optimal, First in First Out (FIFO), Second Chance (SC), Not recently used (NRU) and Least Recently used (LRU).
File Management: Concept of File, Access methods, File types, File operation, Directory structure, Allocation methods (contiguous, linked, indexed).

Unit Number: 4 Software Project Management No. of hours: 10

Process-Synchronization & Deadlocks: Inter-process Communication: Critical Section, Race Conditions, Mutual Exclusion, Peterson's Solution, The Producer\ Consumer Problem, Semaphores, Event Counters, Message Passing, Classical IPC Problems: Reader's & Writer Problem, Dinning Philosopher Problem etc.
Definition of Deadlocks, Necessary and sufficient conditions for Deadlock, Deadlock Prevention, Deadlock Avoidance: Banker's algorithm, Deadlock detection and Recovery.

***Self-Learning Components:**
1. Case study on UNIX and WINDOWS Operating System.

2. Practice of System calls

3. Students can refer the following book as well:

Operating Systems: Three Easy Pieces by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau

https://pages.cs.wisc.edu/~remzi/OSTEP/

4. Students can refer the following courses as per the Open-Source University Curriculum

- "Operating system courses" on Udemy.
- " Introduction to Operating Systems Specialization" Coursera.
- "Introduction to Operating Systems" by Udacity.

Reference Books :

1. Silbersachatz and Galvin, "Operating System Concepts", Pearson
2. Tannenbaum, "Operating Systems", PHI, 4th Edition.
2. William Stallings, "Operating Systems Internals and Design Principles", PHI
3. HallMadnick, J. Donovan, "Operating Systems", Tata McGraw Hill.
4. W. Tomasi, "Electronic Communication Systems" Pearson Education, 5th Edition

Program and Course Outcome Mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENCA202/ **Fundamentals of Operating Systems**	CO1	3	-	2	-	-	-	-	-	-	-	2	1	3	2	1	1
	CO2	3	-	-	-	-	-	1	-	-	-	3	1	2	2	1	1
	CO3	-	3	3	-	-	-	1	-	-	-	2	1	3	3	2	1
	CO4	-	3	3	2	-	-	-	-	1	-	-	1	2	2	2	2
	CO5	1	2	2	-	-	-	-	-	-	-	-	-	2	1	1	1

Department:	**Department of Computer Science and Engineering**		
Course Name: **Fundamentals of Operating Systems Lab**	**Course Code**	**L-T-P**	**Credits**
	ENCA252	0-0-2	1
Type of Course:	MAJOR		
Pre-requisite(s), if any: Basics of programming			

Proposed Lab Experiments

Defined Course Outcomes

COs	Statements
CO 1	Recall the concepts and principles of CPU scheduling algorithms used in operating systems.
CO 2	Compare and contrast different CPU scheduling algorithms and their advantages and disadvantages.
CO 3	Implement CPU scheduling algorithms, such as Round Robin and Priority, using Python programming.
CO 4	Evaluate the performance of CPU scheduling algorithms by analyzing and interpreting the generated Gantt charts and calculating average waiting time and turnaround time.
CO 5	Design Python programs to simulate various file allocation strategies and memory management techniques, such as sequential, indexed, linked, and paging.

List of Programs

Ex No	Experiment Title	Mapped CO/COs
1	Write Python programs to simulate the following CPU Scheduling algorithm: First-Come, First-Served (FCFS)	CO1
2	Write Python programs to simulate the following CPU Scheduling algorithm: Shortest Job First (SJF)	CO1
3	Write Python programs to simulate the following CPU Scheduling algorithms: Round Robin	CO1
4	Write Python programs to simulate the following CPU Scheduling algorithms: Priority	CO1
5	Given the list of processes, their CPU burst times, and arrival times, write a Python program to display/print the Gantt chart for Priority and Round Robin scheduling	CO4

	algorithms. Compute and print the average waiting time and average turnaround time for each scheduling policy.	
6	Write a Python program to simulate the following file allocation strategies like Sequential	CO5
7	Write a Python program to simulate the following file allocation strategies like Indexed	CO5
8	Write a Python program to simulate the following file allocation strategies like linked.	CO5
9	Write Python programs to simulate the following contiguous memory allocation techniques: a) Worst-fit b) Best-fit c) First-fit	CO5
10	Write Python programs using the I/O system calls of UNIX/Linux operating system (open, read, write, close, fcntl, seek, stat, opendir, readdir).	CO1
11	Write a Python program to simulate the MVT (Multiple Variable Tasks) memory management technique.	CO5
12	Write a Python program to simulate the MFT (Multiple Fixed Tasks) memory management technique.	CO5
13	Write a Python program to simulate the Banker's Algorithm for Deadlock Avoidance and Prevention.	CO5
14	Write a Python program to implement the Producer-Consumer problem using semaphores using UNIX/Linux system calls.	CO3
15	Write Python programs to illustrate the following IPC (Inter-Process Communication) mechanisms: a) Pipes	CO3
16	Write Python programs to illustrate the following IPC (Inter-Process Communication) mechanisms: a) FIFOs (Named Pipes)	CO3
17	Program to implement process synchronization using semaphores in Python.	CO4
18	Program to implement a basic Fo5ile allocation strategy like sequential file allocation in Python.	CO5
19	Program to demonstrate the use of signals in Python for process management.	CO1
20	Program to create and manipulate threads in Python.	CO3
21	Program to implement memory management techniques (e.g., paging, segmentation) in Python.	CO5
22	Program to simulate file system operations (e.g., open, read, write, close) in Python.	CO1
23	Program to implement process synchronization using mutex locks in Python.	CO4

24	Program to simulate the working of virtual memory in Python.	CO5
25	Program to simulate disk file management operations (e.g., allocation, deallocation) in Python.	CO5
26	Program to implement file locking mechanisms (e.g., advisory, mandatory) in Python.	CO5
27	Write a Python program to simulate the following file organization techniques Two level directories	CO5
28	Write Python programs to simulate the paging in memory management techniques	CO5
29	Write Python programs to simulate the segmentation in memory management techniques	CO5
30	Write a Python program to simulate the following file organization techniques Single level directory	CO5

Department:	**Department of Computer Science and Engineering**		
Course Name: Fundamentals of Database Management Systems	**Course Code**	**L-T-P**	**Credits**
	ENCA204	3-1-0	4
Type of Course:	MAJOR		
Pre-requisite(s), if any: Basics of programming			

Define Course Outcomes (CO)

COs	Statements
CO1	Analyze the key components and concepts of DBMS, including data independence, architecture, schemas and various DBMS models.
CO2	Apply data modeling techniques using ER model and understanding the concepts of keys
CO3	Evaluate the principles and techniques of relational modeling and the fundamental operations of relational algebra.
CO4	Design and implement effective database designs by analyzing functional dependencies and normalization.
CO5	Explain transaction processing, concurrency control and database recovery protocols in databases.

Brief Syllabus:
This course introduces the basic concept of database, Database modelling languages, E-R modelling, Transaction Processing and Database security.

UNIT WISE DETAILS

Unit Number: 1 **Introduction to database** **No. of hours: 8**

Overview of DBMS, DBMS system vs file system, Data independence and abstraction level, Architecture of DBMS, Schemas, Instances and various DBMS models.

Unit Number: 2 **Data Modelling and Languages** **No. of hours: 10**

Data Modelling: Data modeling using Entity relationship Model: ER Model Concepts, notation of ER diagram, mapping constraints, Keys, concept of super key, candidate key, primary key, generalization and specialization
Relational Modelling: Concepts, constraints, Language, Relational Database Design by ER and EER mapping, Relational Algebra, Relational Calculus, relational Algebra and its fundamental operations.
Mini project: Draw ER diagram to design a database to manage university course registration, including student records, courses, instructors, prerequisites, and enrolment.

Unit Number: 3 **Database design and Transaction Processing** **No. of hours: 8**

Database design: Functional Dependencies, lossless decomposition and Normalization (1NF, 2NF, 3NF, BCNF, 4NF)
Transaction management: transaction concept, ACID properties, state of transaction, serializability, checkpoints and deadlock handling.

Mini project: Design a database to manage a library's catalog, including books, authors, genres, and borrower information. Normalize the database to eliminate data duplication and maintain consistency.

Unit Number: 4 **Introduction to SQL** **No. of hours: 12**

Introduction to SQL: characteristics and advantages of SQL, SQL data types, SQL commands and operators, Tables, views and indexes, Queries and sub-queries, aggregate function, insert, alter and update operations
Mini project : Create Client_master with the following fields(ClientNO, Name, Address, City, State, bal_due)
(a) Insert five records
(b) Find the names of clients whose bal_due> 5000 .
(c) Change the bal_due of ClientNO " C123" to Rs. 5100
(d) Change the name of Client_master to Client12.
(e) Display the bal_due heading as "BALANCE"

***Self-Learning Components**:

- **PostgreSQL**
- **MongoDB**

Note: Students will give presentations and submit projects based on self-learning components for evaluation.
Reference Books:
1."Database System Concepts", 6th Edition by Abraham Silberschatz, Henry F. Korth, S. Sudarshan, McGraw-Hill.
2. "Principles of Database and Knowledge – Base Systems", Vol 1 by J.D. Ullman, Computer Science Press.
3. https://github.com/ossu/computer-science#databases.(OSSU computer science curriculum)

Program and Course Outcome Mapping

Course Code and Title	**PO**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO 10**	**PO 11**	**PO 12**	**PSO1**	**PSO2**	**PSO3**	**PS O4**
ENCA204/ Fundamentals of Database	**CO1**	3	-	-	3	-	-	-	-	-	2	-	1	2	-	1	1
	CO2	-	2	2	3	2	-	-	-	-	-	-	-	2	2	2	-
	CO3	2	2	1	3	-	-	-	-	-	-	-	-	1	2	1	-
	CO4	3	2	3	2	-	-	-	-	-	-	-	-	1	2	2	-
	CO5	-	-	-	3	-	2	2	-	2	-	-	-	1	1	2	1

Department:	**Department of Computer Science and Engineering**		
Course Name: Fundamentals of Database Management Systems Lab	**Course Code**	**L-T-P**	**Credits**
	ENCS254	0-0-2	1
Type of Course:	MAJOR		
Pre-requisite(s), if any: Basics of programming			

Defined Course Outcomes

COs	Statements
CO 1	Define and apply mapping constraints to transform an ER model into a relational schema
CO 2	Demonstrate an understanding of keys (super key, candidate key, primary key) and their roles in database design
CO 3	Perform data manipulation operations such as insertion, deletion, and updating using SQL commands
CO 4	Create and manage database objects like tables, views, and indexes using SQL statements

Ex. No	Experiment Title	Mapped CO/COs
1	Consider following databases and draw ER diagram and convert entities and relationships to relation table for a given scenario: COLLEGE DATABASE: STUDENT (USN, SName, Address, Phone, Gender) SEMSEC (SSID, Sem, Sec) CLASS (USN, SSID) SUBJECT (Subcode, Title, Sem, Credits) IAMARKS (USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)	CO1, CO2
2	Consider following databases and draw ER diagram and convert entities and relationships to relation table for a given scenario: COMPANY DATABASE: EMPLOYEE (SSN, Name, Address, Sex, Salary, SuperSSN, DNo) DEPARTMENT (DNo, DName, MgrSSN, MgrStartDate) DLOCATION (DNo,DLoc) PROJECT (PNo, PName, PLocation, DNo) WORKS_ON (SSN, PNo, Hours)	CO1, CO2
3	Consider the below Database: Movies (title, director, making_year, rating), actors (actor, acting_year), acts(actor, title), directors (director, director_year) Write relation algebra queries for given relations: 1. Find movies made after 1997 2. Find movies made by Hanson after 1997 3. Find all movies and their ratings 4. Find all actors and directors 5. Find Coen’s movies with McDormand	CO3, CO4
4	**Database Schema for a customer-sale scenario** Customer(<u>**Cust id : integer**</u>, cust_name: string) Item(<u>**item_id: integer**</u>, item_name: string, price: integer) Sale(<u>**bill_no: integer**</u>, bill_data: date, **cust_id: integer**, **item_id: integer**, qty_sold: integer) For the above schema, perform the following— i) Create the tables with the appropriate integrity constraints. ii) Insert around 10 records in each of the tables. iii) List all the bills for the current date with the customer names and item numbers.	CO3, CO4

iv) List the total Bill details with the quantity sold, price of the item and the final amount.
v) List the details of the customer who have bought a product which has a price>200.
vi) Give a count of how many products have been bought by each customer
vii) Give a list of products bought by a customer having cust_id as 5.
viii) List the item details which are sold as of today.
ix) Create a view which lists out the bill_no, bill_date, cust_id, item_id, price, qty_sold, amount.
x) Create a view which lists the daily sales date wise for the last one week

5 **Database Schema for a Student Library scenario** CO3, CO4

Student(**Stud_no : integer,** Stud_name: string)
Membership(**Mem_no: integer**, **Stud_no: integer**)
Book(**book_no: integer**, book_name:string, author: string)
Iss_rec**(iss_no:integer**, iss_date: date, **Mem_no: integer**, **book_no: integer**)
For the above schema, perform the following—

i) Create the tables with the appropriate integrity constraints
ii) Insert around 10 records in each of the tables
iii) List all the student names with their membership numbers
iv) List all the issues for the current date with student and Book names
v) List the details of students who borrowed book whose author is CJDATE
vi) Give a count of how many books have been bought by each student
vii) Give a list of books taken by student with stud_no as 5
viii) List the book details which are issued as of today
ix) Create a view which lists out the iss_no, iss _date, stud_name, book name
x) Create a view which lists the daily issues-date wise for the last one week

6 **Database Schema for a Employee-pay scenario** CO3, CO4

employee(**emp_id : integer,** emp_name: string)
department(**dept_id: integer**, dept_name:string)
paydetails(**emp_id : integer**, **dept_id: integer**, basic: integer, deductions: integer, additions: integer, DOJ: date)
payroll**(emp_id : integer**, pay_date: date)
For the above schema, perform the following—

i) Create the tables with the appropriate integrity constraints
ii) Insert around 10 records in each of the tables
iii) List the employee details department wise
iv) List all the employee names who joined after particular date
v) List the details of employees whose basic salary is between 10,000 and 20,000
vi) Give a count of how many employees are working in each department
vii) Give a names of the employees whose netsalary>10,000
viii) List the details for an employee_id=5
ix) Create a view which lists out the emp_name, department, basic, deductions, netsalary
x) Create a view which lists the emp_name and his netsalary

7 **Database Schema for a Video Library scenario** CO3, CO4

Customer(cust_no: integer,cust_name: string)
Membership(**Mem_no: integer**, **cust_no: integer**)
Cassette(**cass_no:integer**, cass_name:string, Language: String)
Iss_rec(**iss_no: integer**, iss_date: date, **mem_no: integer**, **cass_no: integer**)
For the above schema, perform the following—

i) Create the tables with the appropriate integrity constraints

ii) Insert around 10 records in each of the tables
iii) List all the customer names with their membership numbers
iv) List all the issues for the current date with the customer names and cassette names
v) List the details of the customer who has borrowed the cassette whose title is " The Legend"
vi) Give a count of how many cassettes have been borrowed by each customer
vii) Give a list of book which has been taken by the student with mem_no as 5
viii) List the cassettes issues for today
ix) Create a view which lists outs the iss_no, iss_date, cust_name, cass_name
x) Create a view which lists issues-date wise for the last one week

8 **Database Schema for a student-Lab scenario** CO3, CO4

Student(**stud_no: integer**, stud_name: string, **class: string**)
Class(**<u>class: string,</u> descrip: string**)
Lab(**<u>mach_no: integer</u>**, Lab_no: integer, description: String)
Allotment(**Stud_no: Integer, mach_no: integer, dayof week: string)**
For the above schema, perform the following—

i) Create the tables with the appropriate integrity constraints
ii) Insert around 10 records in each of the tables
iii) List all the machine allotments with the student names, lab and machine numbers.
iv) List the total number of lab allotments day wise
v) Give a count of how many machines have been allocated to the 'CSIT' class
vi) Give a machine allotment details of the stud_no 5 with his personal and class details
vii) Count for how many machines have been allocated in **Lab_no 1** for the day of the week as "Monday"
viii) How many students class wise have allocated machines in the labs
ix) Create a view which lists out the stud_no, stud_name, mach_no, lab_no, dayofweek
x) Create a view which lists the machine allotment details for "Thursday".

9 **Consider the following table:** CO3, CO4

Table: CLASS

Id	Name
1	Bravo
2	Alex
4	Cheng

Give the output of the following SQL script:

```
> INSERT INTO class VALUES (5,'Rahul');
> COMMIT;
> UPDATE class SET name = 'Abhijeet' WHERE id= '5';
> SAVEPOINT A;
> INSERT INTO class VALUES (6, 'Chris');
> SAVEPOINT B;
> INSERT INTO class VALUES (7, 'Bravo');
> SAVEPOINT C
> SELECT * FROM class;
> ROLLBACK TO B;
> SELECT * FROM class;
> ROLLBACK TO A;
```

10 **Consider the following two tables: SHOP and ACCESSORIES** CO3, CO4

Table: SHOP

ID	ShopName	Area
S01	ABC Computronics	CP
S02	All Infotech Media	GK II
S03	Tech Shoppe	CP
S04	Geek Tenco Soft	Nehru Place
S05	Hitech Tech Store	Nehru Place

Table: ACCESSORIES

No	Name	Price	Id
A01	Motherboard	12000	S01
A02	Hard Disk	5000	S01
A03	Keyboard	500	S02
A04	Mouse	300	S01
A05	Motherboard	13000	S02
A06	Keyboard	400	S03
A07	LCD	6000	S04
T08	LCD	5500	S05
T09	Mouse	350	S05
T10	Hard Disk	450	S03

i. Perform Cartesian product or Cross join of both tables.
ii. To display the Name and Price of all the Accessories in ascending order of their price.
iii. To display ID and ShopName of all shops located in Nehru Place.
iv. To display minimum and maximum price of all accessories.
v. To display Name, Price of all accessories and their respective ShopName where they are available.

11 **In continuation with experiment no. 10, find the output of the following SQL queries based on above mentioned tables:** CO3, CO4

i. SELECT DISCTINCT NAME FROM ACCESSORIES WHERE PRICE >= 5000;
ii. SELECT AREA, COUNT(*) FROM SHOP GROUP BY AREA;
iii. SELECT COUNT(DISTINCT AREA) FROM SHOP;
iv. SELECT NAME, PRICE*0.05 DISCOUNT FROM ACCESSORIES WHERE ID IN (‘S02’, ‘S03’);

12 **Consider the following two tables: PRODUCT and CLIENT.** CO3, CO4

Table: Product

P_ID	ProdName	Manufacturer	Price	ExpiryDate
TP01	Talcom Powder	LAK	40	2011-06-26
FW05	Face Wash	ABC	45	2010-12-01
BS01	Bath Soap	ABC	55	2010-09-10
SH06	Shampoo	XYZ	120	2012-04-09
FW12	Face Wash	XYZ	95	2010-08-15

Note:

- P_ID is the primary key.

Table: Client

C_ID	ClientName	City	P_ID
1	Cosmetic Shop	Delhi	FW05
6	Total Health	Mumbai	BS01
12	Live Life	Delhi	SH06

15	Pretty One	Delhi	FW05
16	Dreams	Bengaluru	TP01
14	Expressions	Delhi	NULL

Note:

- C_ID is the primary key.
- P_ID is the foreign key referencing P_ID of Client Table.

i. To display the ClientName and City of all Mumbai and Delhi based clients in Client table.
ii. Increase the price of all the products in Product Table by 10%.
iii. To display the ProdName, Manufacturer, ExpiryDate of all the products that expired on or before '2010-12-31'.
iv. To display C_ID, ClientName, City of all the clients including the ones that have not purchased a product and their corresponding ProdName sold.
v. Display the distinct Manufacturer from Product table.
vi. Display the ClientName, C_ID who belong to a city starts with 'M'

13 Consider the following schema for a Library Database: BOOK(Book_id, Title, Publisher_Name, Pub_Year) BOOK_AUTHORS(Book_id, Author_Name) PUBLISHER(Name, Address, Phone) BOOK_COPIES(Book_id, Programme_id, No-of_Copies) BOOK_LENDING(Book_id, Programme_id, Card_No, Date_Out, Due_Date) LIBRARY_PROGRAMME(Programme_id, Programme_Name, Address) CO3, CO4

Write SQL queries to

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each Programme, etc.
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.
3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.
4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
5. Create a view of all books and its number of copies that are currently available in the Library

14 Consider the following schema for Order Database: SALESMAN(Salesman_id, Name, City, Commission) CUSTOMER(Customer_id, Cust_Name, City, Grade, Salesman_id) ORDERS(Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id) CO3, CO4

Write SQL queries to

1. Count the customers with grades above Bangalore's average.
2. Find the name and numbers of all salesman who had more than one customer.
3. List all the salesman and indicate those who have and do not have customers in their cities (Use UNION operation.)
4. Create a view that finds the salesman who has the customer with the highest order of a day.
5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

15 Consider the schema for Movie Database: ACTOR(Act_id, Act_Name, Act_Gender) DIRECTOR(Dir_id, Dir_Name, Dir_Phone) CO3, CO4

	MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id) MOVIE_CAST(Act_id, Mov_id, Role) RATING(Mov_id, Rev_Stars) Write SQL queries to 1. List the titles of all movies directed by ‘Hitchcock’. 2. Find the movie names where one or more actors acted in two or more movies. 3. List all actors who acted in a movie before 2000 and in a movie after 2015 (use JOIN operation). 4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title. 5. Update rating of all movies directed by ‘Steven Spielberg’ to 5.	
16	Consider the schema for College Database: STUDENT(USN, SName, Address, Phone, Gender) SEMSEC(SSID, Sem, Sec) CLASS(USN, SSID) COURSE(Subcode, Title, Sem, Credits) IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA) Write SQL queries to 1. List all the student details studying in fourth semester ‘C’ section. 2. Compute the total number of male and female students in each semester and in each section. 3. Create a view of Test1 marks of student USN ‘1BI15CS101’ in all Courses. 4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students. 5. Categorize students based on the following criterion: If FinalIA = 17 to 20 then CAT = ‘Outstanding’ If FinalIA = 12 to 16 then CAT = ‘Average’ If FinalIA< 12 then CAT = ‘Weak’ Give these details only for 8th semester A, B, and C section students.	CO3, CO4
17	Consider the schema for Company Database: EMPLOYEE(SSN, Name, Address, Sex, Salary, SuperSSN, DNo) DEPARTMENT(DNo, DName, MgrSSN, MgrStartDate) DLOCATION(DNo,DLoc) PROJECT(PNo, PName, PLocation, DNo) WORKS_ON(SSN, PNo, Hours) Write SQL queries to 1. Make a list of all project numbers for projects that involve an employee whose last name is ‘Scott’, either as a worker or as a manager of the department that controls the project. 2. Show the resulting salaries if every employee working on the ‘IoT’ project is given a 10 percent raise. 3. Find the sum of the salaries of all employees of the ‘Accounts’ department, as well as the maximum salary, the minimum salary, and the average salary in this department. 4. Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator). 5. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.	CO3, CO4

Department:	**Department of Computer Science and Engineering**		
Course Name: **Foundation of Machine Learning**	**Course Code**	**L-T-P**	**Credits**
	ENSP212	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s), if any: Basics of programming			

Define Course Outcomes (CO)

COs	**Statements**
CO1	Explain the use of Machine Learning Models in business and understand machine learning models can be used to solve business problems.
CO2	Compare machine learning algorithms such as supervised, unsupervised, and reinforcement learning models.
CO3	Identify the performance of different machine learning models and compare them to optimize the results.
CO4	Make use continuous and discrete data set to fit regression and classification models.

Brief Syllabus:
Help student understand what machine learning is. How business can use machine learning in different domains to gain competitive advantage. Student is able to differentiate between different learning algorithms. To understand different data science processes, tools and techniques. Gain a fundamental understanding of the concepts and techniques that underpin machine learning algorithms

UNIT WISE DETAILS

Unit Number: 1 **Introduction to Machine Learning** **No. of hours: 10**

Learning systems, real world applications of machine learning, why machine learning, variable types and terminology, function approximation,

Types of machine learning: Supervised learning, unsupervised learning, Reinforcement learning

Unit Number: 2 **Important concepts of machine learning** **No. of hours: 10**

Parametric vs non-parametric models, the trade-off between prediction accuracy and model interpretability, the curse of dimensionality, measuring the quality of fit, bias-variance trade off, overfitting, model selection, no free lunch theorem.

Unit Number: 3 **Linear Regression** **No. of hours: 10**

Linear regression, estimating the coefficients, accessing the accuracy of coefficient estimates, accessing the accuracy of the model, multiple linear regression, qualitative predictors.

Unit Number: 4 **Classification** **No. of hours: 10**

Logistic regression, estimating regression coefficients, making predictions, multiple logistic regressions, linear discriminant analysis, bayes' theorem of classification, LDA for p=1, LDA for p>1, quadratic discriminant analysis.

***Self-Learning Components**:
1)Students are supposed to learn the components on self-basis

2)Mention open-source tools/ new concepts/technologies that students will be required to learn and present through presentations in class

3) At least 5-10 % syllabus will be asked in end term exams from self-learning components

Reference Books:

1. Machine Learning by Tom M. Mitchell - McGraw Hill Education; First edition.
2. Pattern Recognition and Machine Learning (Information Science and Statistics) by Christopher M. Bishop - Springer; 1st ed. 2006. Corr. 2nd printing 2011 edition.

The Elements of Statistical Learning: Data Mining, Inference, and Prediction by Trevor Hastie, Robert Tibshirani, Jerome Friedman - Springer; 2nd ed. 2009, Corr. 9th printing 2017 edition.

Program and Course Outcome Mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12	PSO1	PSO2	PSO3	PS O4
ENSP212/ Foundation of Machine	CO1	2	3	1	1			1			1	1	1	2	3	2	2
	CO2	2	3	2	1	1		1				2	1	3	3	3	2
	CO3	2		2	1			2			1	1	1	2	2	3	2
	CO4	2	1	2	2			1				1	1	2	3	2	3
	CO5	2	3	1	1			1			1	1	1	2	3	2	2

Department:	**Department of Computer Science and Engineering**		
Course Name: **Foundation of Machine Learning Lab**	**Course Code**	**L-T-P**	**Credits**
	ENSP262	0-0-2	1
Type of Course:	Minor		
Pre-requisite(s), if any: Basics of programming			

Proposed Lab Experiments

Defined Course Outcomes

COs	Statements
CO 1	Explain the use of Machine Learning Models in business and understand machine learning models can be used to solve business problems.
CO 2	Compare machine learning algorithms such as supervised, unsupervised, and reinforcement learning models.
CO 3	Identify the performance of different machine learning models and compare them to optimize the results.
CO 4	Make use continuous and discrete data set to fit regression and classification models.

List of Programs

Ex No	Experiment Title	Mapped CO/COs
1	Prediction using simple linear regression	CO1
2	Prediction using multiple linear regression	CO1
3	Classification using Logistics regression	CO1
4	Classification using linear discriminant analysis	CO1
5	Classification using support vector machine.	CO2
6	Classification using Guassian Naïve Bayes	CO2
7	Classification using decision Tree	CO2
8	Classification using Random Forest.	CO1
9	Classification using K nearest neighbour.	CO4
10	Write a program to Retrieve Data for a machine Learning project.	CO3
11	Write a program to Conduct Exploratory Data Analysis using Python	CO3
12	Write a program to Clean the Data using Python	CO4
13	Write a program for Data Modeling using Python	CO4
14	Write a program to analyze and solve Null Value problem.	CO2
15	Write a program to analyze and solve zero values	CO2

16	Write a program to analyze the categorical values	CO2
17	Write a program for graphical representation of data.	CO1
18	Write a program for logistic regression using statsmodel.	CO3
19	Write a program to implement multiple logistic regression.	CO4
20	Write a program to scale the data and implement linear regression using sklearn.	CO3
21	Write a program to implement multiple linear regression.	CO2

ODD SEMESTER (V)							
SNo	Course Code	Course Title	Category	L	T	P	C
1	ENCA301	Design and Analysis of Algorithms	Major	3	1	0	4
2	ENCA303	Theory of Automata	Major	3	1	0	4
3	ENSP302	Introduction to Natural Language Processing	Minor	4	0	0	4
4	ENSP309	Big Data Analysis with Scala and Spark	Minor	4	-	-	4
5	SEC040	Data Science - Tools and Techniques Lab	SEC	0	0	4	2
7	AEC013	Life Skills for Professionals-III	AEC	3	0	0	3
8	ENCA351	Design & Analysis of Algorithms Lab	Major	0	0	2	1
9	ENSP352	Natural Language Processing Lab	Minor	0	0	2	1
10	ENSP359	Big Data Analysis with Scala and Spark Lab	Minor	-	-	2	1
11		Summer Internship /Project	INT	0	0	0	2
TOTAL				17	2	10	26

Department	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Analysis and Design of Algorithms	**ENCA301**	**3-1-0**	**4**
Type of Course:	**Major**		
Pre-requisite(s), if any: - Data Structure			

COs	Statements
CO1	Understand fundamental algorithmic concepts and how to analyze Complexities.
CO2	Analyze and evaluate algorithm performance.
CO3	Design efficient algorithms in terms of space and time.
CO4	Apply algorithmic problem-solving strategies.
CO5	Develop algorithm implementation skills.

Brief Syllabus:
The analysis and design of algorithm course introduce students to the design of computer algorithms, as well as analysis of sophisticated algorithms. Students will learn how to analyse the asymptotic performance of algorithms as well as provides familiarity with major algorithms and data structures. This course introduces basic methods for the design and analysis of efficient algorithms emphasizing methods useful in practice. Different algorithms for a given computational task are presented and their relative merits evaluated based on performance measures. The following important computational problems will be discussed: sorting, searching, elements of dynamic programming and greedy algorithms, advanced data structures, graph algorithms (shortest path, spanning trees, tree traversals), string matching, elements of computational geometry, NP completeness.

UNIT WISE DETAILS

Unit Number: 1 Introduction to Algorithms No. of hours: 8

Characteristics of algorithm. Analysis of algorithm: Asymptotic analysis of complexity bounds – best, average and worst-case behaviour, Performance measurements of Algorithm, Time and Time and space trade- offs, Analysis of recursive algorithms through recurrence relations: Substitution method, Recursion tree method and Masters' theorem.

Unit Number: 2 Fundamental Algorithmic Strategies No. of hours: 4

Brute -Force, Greedy, Dynamic Programming, Branch-and-Bound and Backtracking methodologies for the design of algorithms; Illustrations of these techniques for Problem-Solving, Bin Packing, Knap Sack. Heaps and priority queues, Hash tables and hash functions. String matching

Unit Number: 3 Graph and Tree Algorithms **No. of hours: 8**

Traversal algorithms: Depth First Search (DFS) and Breadth First Search (BFS); Shortest path algorithms, Minimum Spanning Tree, Topological sorting, Network Flow Algorithm. Graph Colouring algorithms.

Unit Number: 4 Tractable and Intractable Problems **No. of hours: 4**

Computability of Algorithms, Computability classes – P, NP, NP complete and NP-hard. Cook's theorem, Standard NP-complete problems and Reduction techniques.

Self-Learning Components
Container loading problem, stable marriage problem, Coin Change problem

Reference Books

1. Introduction to Algorithms, 4TH Edition, Thomas H Cormen, Charles E Lieserson, Ronald L Rivest and Clifford Stein, MIT Press/McGraw-Hill.
2. Fundamentals of Algorithms – E. Horowitz et al.

Program and Course Outcome Mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12	PSO1	PSO2	PSO3	PS O4
ENCA301/ Design and Analysis of Algorithms	CO1	3	3	-	-	-	-	-	-	-	-	-	2	3	3	-	-
	CO2	-	-	-	3	2	-	-	-	-	-	-	1	-	3	-	3
	CO3	-	-	3	-	-	-	-	-	-	-	-	3	3	2	-	-
	CO4	-	-	-	-	2	-	-	-	2	-	-	-	-	3	3	-
	CO5	-	-	-	3	-	-	-	-	-	2	-	-	-	2	3	-

Department:	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Analysis and Design of Algorithms Lab	**ENCA351**	**0-0-2**	**1**
Type of Course:	**Major**		
Pre-requisite(s), if any: -			

Defined Course Outcomes

COs	Statements
CO 1	Analyze the time and space complexities of algorithms and evaluate their performance
CO 2	Apply algorithmic problem-solving strategies to solve complex computational problems
CO 3	Design and develop innovative algorithms for solving complex computational problems.
CO 4	Generate algorithmic solutions that consider trade-offs between time complexity, space complexity, and problem constraints.

Ex. No	Experiment Title	Mapped CO/COs
1	Apply quick sort algorithm.	CO1
2	Design an algorithm to find the maximum and minimum elements in an unsorted array.	CO1
3	Implement the Largest Common Subsequence.	CO1
4	Find the Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm.	CO1
5	Find the Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm.	CO2
6	To Implement Optimal Binary Search Tree.	CO2
7	To Implement Strassen’s matrix multiplication Algorithm	CO2
8	Design an algorithm to find the maximum subarray sum in an array.	CO2
9	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.	CO2
10	Implement 0/1 Knapsack Problem using Dynamic algorithm concepts.	CO2
11	To implement Bellman Ford’s Algorithm.	CO2
12	To implement Depth First Search and Breadth First Search Algorithm.	CO2
13	To implement Naïve String-matching Algorithm.	CO3
14	Implement N Queen's problem using Back Tracking.	CO3
15	Design an algorithm to check if a given graph is acyclic (a DAG).	CO3
16	Obtain the Topological ordering of vertices in a given digraph.	CO3

17	Design an algorithm to find the nth Fibonacci number using dynamic programming.	CO3
18	Implement the brute-force algorithm to solve the Subset Sum Problem.	CO4

Department:	**Department of Computer Science and Engineering**		
Course Name: Theory of Automata	**Course Code**	**L-T-P**	**Credits**
	ENCA303	3-1-0	4
Type of Course:	**Major**		
Pre-requisite(s), if any:			

COs	**Statements**
CO1	To solve the problems related to regular expression, regular grammar, and Finite Automata
CO2	To write a formal notation for strings, languages and machines
CO3	To identify the phases of compilers for a programming language and construct the parsing table for a given syntax
CO4	To discover syntax directed translation rules for a given context free grammar by examining S-attributed and L-attributed grammars
CO5	To construct grammars and machines for a context free and context sensitive languages

Brief Syllabus:
This course provides a formal connection between algorithmic problem solving and the theory of languages and automata and develop them into a mathematical view towards algorithmic design and in general computation itself. The course should in addition clarify the practical view towards the applications of these ideas in the engineering part of computer science.

Unit Number: 1 Introduction to Finite automata No. of hours: 12

Finite automata: Review of Automata, its types and regular expressions, Equivalence of NFA, DFA and €-NFA, Conversion of automata and regular expression, Applications of Finite Automata to lexical analysis

Unit Number: 2 PDA and Parser No. of hours: 10

PDA and Parser: Parse Trees, Ambiguity in grammars and languages, Push down automata, Context Free grammars, Top down and Bottom-up parsing. Closure Properties of CFL.

Unit Number: 3 Chomsky hierarchy and Turing Machine No. of hours: 08

Chomsky hierarchy and Turing Machine: Chomsky hierarchy of languages and recognizers, Context Sensitive features like type checking, Turing Machine as language acceptors and its design.

Unit Number: 4 **Code generation and optimization** **No. of hours: 10**

Code generation and optimization: Syntax-directed translation, S-attributed and L-attributed grammars, Intermediate code generation, type conversions, and equivalence of type expression, Code generation, and optimization.

Text Books

1. J.E. Hopcroft, R. Motwani and J.D. Ullman, "Introduction to Automata Theory, Languages and Computations", second Edition, Pearson Education.

Reference Books/Materials

1. H.R. Lewis and C.H. Papadimitriou, "Elements of the theory of Computation", Second Edition, Pearson Education.
2. Thomas A. Sudkamp," An Introduction to the Theory of Computer Science, Languages and Machines", Third Edition, Pearson Education.
3. Raymond Greenlaw an H.James Hoover, "Fundamentals of Theory of Computation, Principles and Practice", Morgan Kaufmann Publishers.
4. MichealSipser, "Introduction of the Theory and Computation", Thomson Brokecole.
5. J. Martin, "Introduction to Languages and the Theory of Computation" Third Edition, Tata Mc Graw Hill.

Program and Course Outcome Mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENCA301/ Design and Analysis of Algorithms	CO1	3	3	-	-	-	-	-	-	-	-	-	2	3	1	2	1
	CO2	2	2	-	3	-	-	3	-	-	-	-	1	3	2	2	2
	CO3	2	3	3	3	-	-	3	3	3		-	3	3	3	2	2
	CO4	3	3	3	3	3		1	-	-	3	-	-	2	3	2	3
	CO5	1	-	-	2	-	-	-	-	-	-	-	-	3	3	2	3

Department:	**Department of Computer Science and Engineering**		
Course Name: NATURAL LANGUAGE PROCESSING	**Course Code**	**L-T-P**	**Credits**
	ENCA303	**4-0-0**	**4**
Type of Course:	**Major**		
Pre-requisite(s)- Strong programming skills, particularly in Python.			

COs	Statements
CO1	Understand the fundamentals of Natural Language Processing (NLP).
CO2	Analyse and represent text data using various techniques.
CO3	Implement text classification and information extraction techniques
CO4	Apply NLP techniques to analyse social media data
CO5	Develop practical solutions using NLP for real-world problems

Brief Syllabus:
The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable.
It helps resolve ambiguity in language and adds useful numeric structure to the data for many downstream applications, such as speech recognition or text analytics.

UNIT WISE DETAILS

Unit Number: 1 **Introduction to NLP** **No. of hours: 10**

Natural Language Processing in real world, what is language, Approached to NLP,
Build NLP model: Eights Steps for building NLP Model, Web Scrapping

Unit Number: 2 **Text Representation** **No. of hours: 10**

Basic Vectorization, One-Hot Encoding, Bag of Words, Bag of N Grams, TF-IDF, Pre-trained Word Embedding, Custom Word Embeddings, Vector Representations via averaging, Doc2Vec Model.
Text Classification: Application of Text Classification, Steps for building text classification system, Text classification using Naïve Bayes Classifier, Logistic Regression, and Support Vector Machine.

Unit Number: 3 **Information Extraction** **No. of hours: 10**

Applications of Information Extraction, Processes for Information Extraction. Key phrase Extraction, Named Entity Recognition, Disambiguation, and linking of named entity
Chatbot: Real-life applications of chatbot, Chatbot Taxonomy, Dialog Systems, Process of building a dialog, Components of Dialog Systems.

Unit Number: 4 **NLP for social media** **No. of hours: 10**

Application of NLP in social media, challenges with social media, Natural Language Processing for Social Data, Understanding Twitter Sentiments, Identifying memes and Fake News
NLP for E-Commerce: E-commerce catalog, Search in E-Commerce, How to build an e-commerce catalog, Review and Sentiment Analysis.

***SELF-LEARNING COMPONENTS:**
https://onlinecourses.nptel.ac.in/noc23_cs45/preview

Please Note:
1)Students are supposed to learn the components on self-basis
2) At least 5-10 % syllabus will be asked in end-term exams from self-learning components.

Reference Books: Natural Language Processing with Python by Steven Bird, Ewan Klein, and Edward Loper
Foundations of Statistical Natural Language Processing by Christopher Manning and Hinrich Schütze

Program and Course Outcome Mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENCA303/ /NATURAL LANGUAGE PROCESSING	CO1	3	-	2	-	2	-	-	-	-	-	-	2	2	-	-	-
	CO2	3	3	2	2	2	-	-	-	-	-	-	3	-	3	-	-
	CO3	3	-	--	-	3	-	2	-	-	-	-	3	3	-	-	-
	CO4	-	-	3	-	1	-	-	-		-	2	2	-	-	2	-
	CO5	3	2	-	2	2	-	-	-	-	-	-	3	-	-	-	3

Department:	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Big Data Analytics with Spark + Scala	**ENSP309**	**4 - -**	**4**
Type of Course:	**Minor**		
Pre-requisite(s), if any: NA			

COs	Statements
CO1	Understand/Gain a comprehensive understanding of Apache Spark and its ecosystem, including Spark Core, Spark SQL, Spark Streaming, and Spark MLlib. Understand the concepts and features of distributed computing and in-memory processing offered by Spark.
CO2	Express Clearly express the objectives, requirements, and challenges of big data analysis to stakeholders. Communicate the advantages and potential impact of utilizing Apache Spark and Scala for data analysis, emphasizing the scalability, performance, and versatility of the Spark framework.
CO3	Determine and Assess the suitability of Apache Spark and Scala for the specific big data analysis task. Consider factors such as data volume, complexity, processing needs, and available computing resources to determine the appropriate Spark components and techniques to employ.
CO4	Identify relevant datasets and variables to analyze within the big data using Apache Spark. Apply Spark's data manipulation, querying, and transformation capabilities to preprocess and clean the data, ensuring data quality and consistency.
CO5	Articulate Clearly articulate the insights, findings, and outcomes derived from the big data analysis using Apache Spark and Scala. Present the results in a meaningful and actionable manner.

Brief Syllabus:
This syllabus covers the core concepts and techniques of Apache Spark and Scala in big data analytics. By the end of the course, you will have a solid foundation in using Spark and Scala to manipulate, analyze, and gain insights from large-scale datasets, enabling you to tackle real-world big data challenges.

UNIT WISE DETAILS

Unit Number: 1 **Introduction to Apache Spark** **No. of hours: 4**

Apache Spark and Installation: Introduction to Apache Spark, Features of Apache Spark, Apache Spark Stack, Introduction to RDDs, RDD's Transformation, what is good and bad in Map Reduce, Why to use Apache Spark.

Unit Number: 2 Spark: A Hadoop Replacement No. of hours: 8

Java, Scala or Python? Scala, Packages, Data Types, Classes, Calling Functions, Operations, Control Structures.

Unit Number: 3 Resilient Distributed Datasets (RDD) and SQL Data Frames No. of hours: 8

Introduction to RDD, RDD Operations, Creating RDDs, Transformations, map, flat Map, filter, union, intersection, subtract, distinct, sample, Actions, working with key/value pair RDD, Data Shuffling, Spark SQL, SQL Tables and Views, unmanaged and managed tables, create SQL database and tables, create view, reading tables into Data Frame, Data Frame Reader, Data Frame Writer, Parquet, JSON, reading JSON file into Data Frame, reading CSV file, reading Avro, ORC, Image file,

Unit Number: 4 Spark Streaming No. of hours: 8

Evolution of Apache Spark Stream Processing Engine, Micro-batch stream processing, D Streams, philosophy of structured streaming, programming model, Stream Data Source and sink, structured streaming application, streaming Data Frame Operations, joining two streaming Data Frames, working with socket Data Source, Rate Data Source, File Data Source, Kafka Data Source, Custom Data Source, Working with Data Sinks, Kafka Data Sinks, Foreach Data Sinks, Console Data Sinks, Memory Data Sinks, Output modes and Triggers.

***Self-Learning Components: mention 4-5 topics for students in bullet points**

- Big Data Analytics with Apache Spark by Data Camp https://www.datacamp.com/tutorial/apache-spark-tutorial-machine-learning
- Big Data Analysis with Scala and Spark by Coursera
- Apache Spark with Scala" by Udemy https://www.udemy.com/course/apache-spark-programming-in-scala/
- https://intellipaat.com/apache-spark-scala-training
- Apache Spark. EdX. https://www.edx.org/learn/apache-spark

Please Note:
1)Students are supposed to learn the components on self-basis
2)Mention open-source tools/ new concepts/technologies that students will be required to learn and present through presentations in class
3) At least 5-10 % syllabus will be asked in end-term exams from self-learning components
Reference Books:

- Tom White "Hadoop: The Definitive Guide" Third Edit on, O"reily Media, 2012.
- Gerard Maas and Francois Garillot, "Stream Processing with Apache Spark: Mastering Structured Streaming and Spark Streaming", O'Reilly, 2019.
- "Spark: The Definitive Guide" by Bill Chambers and Matei Zaharia.

Program and Course Outcome Mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENSP309/ Big Data Analytics	CO1	3	2	2	-	-	-	3	2	3	1	2	1	2	-	-	-
	CO2	2	2	3	1	-	1	3	2	3	2	1	1	-	3	-	-
	CO3	2	2	2	1	-	1	2	3	1	1	2	1	3	-	-	-
	CO4	3	3	2	2	-	-	2	2	2	2	1	1	-	-	2	-
	CO5	2	2	2	1	-	-	3	2	2	1	1	1	-	-	-	3

Department:	Department of Computer Science and Engineering		
Course Name: Big Data Analysis with Scala and Spark Lab	Course Code	L-T-P	Credits
	ENSP359	0-0-2	1
Type of Course:	Minor		
Pre-requisite(s), if any: Strong programming skills, particularly in Python.			

Defined Course Outcomes

COs	Statements
CO1	Understand the concepts and features of distributed computing and in-memory processing offered by Spark.
CO2	Clearly express the objectives, requirements, and challenges of big data analysis to stakeholders.
CO3	**Determine** and Assess the suitability of Apache Spark and Scala for the specific big data analysis task..
CO4	**Identify** relevant datasets and variables to analyze within the big data using Apache Spark.
CO5	**Articulate** Clearly articulate the insights, findings, and outcomes derived from the big data analysis using Apache Spark and Scala.

Ex. No	Experiment Title	Mapped CO/COs
1	Installing and configuring Apache Spark	
2	Installing and configuring the Scala IDE	
3	Installing and configuring JDK	
4	Word Count: Perform a word count on a large text dataset using Spark and Scala.	
5	Log Analysis: Analyze server logs to extract useful information such as error rates, response times, and traffic patterns using Spark and Scala.	
6	Create Spark RDD using parallelize with spark Context. Parallelize () method and using Spark shell	
7	Write a scripts in Spark to Read all text files from a directory into a single RDD	
8	Write a spark program to load a CSV file into Spark RDD using a Scala	

9	Write a Spark Streaming program for adding 1 to the stream of integers in a reliable, fault tolerant manner, and then visualize them.
10	Web Scraping: Scrape data from websites using Spark and Scala, and perform analysis on the extracted data.
11	Time Series Analysis: Analyze time series data using Spark and Scala to identify patterns and trends.
12	Anomaly Detection: Detect anomalies in large-scale datasets using Spark MLlib and Scala.
13	Network Traffic Analysis: Analyze network traffic data to detect anomalies and patterns using Spark and Scala.
14	Develop a streaming application by- Connecting to a Stream, Preparing the Data in the Stream, Performing Operations on Streaming Dataset, creating a Query, Starting the Stream Processing and Exploring the data.
15	Create a Structured streaming job by Initializing Spark, acquiring streaming data from sources, declaring the operations we want to apply to the streaming data and outputting the resulting data using Sinks.
16	Create a small but complete Internet of Things (IoT)-inspired streaming program.
17	Define the schema in Structured Streaming to handle the data at different levels.
18	Develop any Spark Streaming application and do the following : a) Create a Spark Streaming Context, b) Define one or several DStreams from data sources or other DStreams c) Define one or more output operations to materialize the results of these
19	Movie Recommendation System: Build a movie recommendation system using collaborative filtering with Spark MLlib and Scala.
20	E-commerce Recommendation System: Build a recommendation system for an e-commerce platform using collaborative filtering with Spark MLlib and Scala.

Department:	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Data Science - Tools and Techniques Lab	**SEC040**	**0-0-4**	**2**
Type of Course:	**SEC**		
Pre-requisite(s), if any: General understanding of Scala 2. Experience with Java (preferred), Python, or another object- oriented language 3. General understanding of machine learning			

COs	By the end of this lab, the learners will be able to
CO 1	Decision-centred Visualization begins with understanding the purpose, data, and context
CO 2	Use common graphs like Bar chats, tree maps, line charts, radars, bubble charts, and heatmaps.
CO 3	Design and implement data models on their own using techniques.
CO 4	Analyse and Select visualization using accuracy techniques for data shape and flow

Brief Syllabus: Learn the foundations of the language for developers and data scientists interested in using Scala for data analysis. Tackle data analysis problems involving Big Data, Scala and Spark. Get a solid understanding of the fundamentals of the language, the tooling, and the development process.
Develop a good appreciation of more advanced features.

Unit Number: 1 **No. of hours: 8**

Scala Language: Getting to know Scala programming language, Scala and Java, statically typed language, Apache Spark and Scala, Scala Performance Benefits, Installing Scala, Using Scala REPL/Shell, getting help from Scala shell, Hello World, Paste mode, retrieving history, auto-complete feature, exiting from Scala REPL

Unit Number: 2 **No. of hours: 8**

Variables, Data Types, Conditional Statements: Immutability of variables, define mutable and immutable variables, mutability and type safety, Specifying types for variables, Scala Identifier rules, naming conventions, Scala data types, Boolean types, string type, multiline strings, string operations, string concatenation, string interpolation, length of string, splitting string, extracting part of string, index of character of strings, the ANY type, type casting, Boolean expressions, conditional statement in Scala, nested IF/ELSE statement, pattern matching.

Unit Number: 3 **No. of hours: 8**

Code Blocks, Functions, Collections: Code Blocks in Scala, Why use functions in Scala, understanding functions in Scala, define and invoke a function, functions with multiple parameters, positional parameters, functions with no argument, single-line function, passing function as an argument, an anonymous function, Collections in Scala, Understanding List, list size, convert list to string, iterating over list, map function and collection, foreach, reduce operation, list equality, create set, indexing map, manipulating maps, understanding tuples, indexing tuples, mutable collections, nested collections

Unit Number: 4 **No. of hours: 8**

Loops, Packages, Classes and Exceptional Handling: For loop, While loop, Breaking Loop iteration, classes and objects in Scala, Create classes and objects, singleton objects, case classes, equality checks, classes and packages, avoid name space collusion, importing package, fundamental of exception handling, type inferences and exception handling, try, catch, finally, Scala built tool (SBT), Compile Scala applications

Self-Learning Components: mention 4-5 topics for students in bullet points
- Advanced topics on Scala from the reference books given
- Learn the concepts from https://learning.samatrix.io further
- Download different dataset from Github and practice the Scala
- Participate in Kaggle Competitions on Scala

- **Reference Books:** Programming in Scala: A comprehensive Step-by-Step Scala Programming Guide by Martin Odersky, Lex Spoon, Bill Venners
- Scala for the Impatient by Cay Hortsmann
- Scala in Depth by Joshua D Suereth

Program and Course Outcome Mapping

Course Code and Title	**PO**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO 10**	**PO1 1**	**PO1 2**	**PSO1**	**PSO2**	**PSO3**	**PSO 4**
SEC040/ Data Science - Tools and Techniques Lab	**CO1**	3	2	2	-	-	-	3	2	3	1	2	1	2	-	-	-
	CO2	2	2	3	1	-	1	3	2	3	2	1	1	-	3	-	-
	CO3	2	2	2	1	-	1	2	3	1	1	2	1	3	-	-	-
	CO4	3	3	2	2	-	-	2	2	2	2	1	1	-	-	2	-
	CO5	2	2	2	1	-	-	3	2	2	1	1	1	-	-	-	3

Department:	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
NATURAL LANGUAGE PROCESSING LAB	**ENSP352**	**0-0-2**	**1**
Type of Course:	**Minor**		
Pre-requisite(s), if any: Strong programming skills, particularly in Python.			

Defined Course Outcomes

COs	
CO 1	**Understand** the fundamental concepts and techniques of web scraping to extract data from websites efficiently and ethically.
CO 2	**Acquire** proficiency in using developer tools to inspect and analyze website elements, facilitating data extraction and understanding the underlying structure of web pages.
CO 3	**Implement** mechanisms to request permission for web scraping, ensuring compliance with legal and ethical guidelines related to data access and usage.
CO 4	**Develop** skills in inspecting specific HTML elements, such as the H1 element and table element, for targeted data extraction and analysis.

List of Experiments

Ex. No.	**Experiment Title**	**Mapped COs**
1	Write a program to scrap a website	CO1, CO3
2	Write a program to inspect a website using dev tools	CO2
3	Write a program to request permission to scrap a website	CO3
4	Write a program to inspect H1 element of a website	CO2, CO4
5	Write a program to inspect table element of a website	CO2, CO4
6	Write a program to create a column list	CO5
7	Write a program to clean a column list	CO5
8	Write a program for word tokenization	CO5
9	Write a program to implement Reg Ex for word tokenization	CO5
10	Write a program to implement stop words	CO5
11	Write a program to implement LSTM	CO5

Department:	**Department of Computer Science and Engineering**		
Course Name: Summer Internship-II	**Course Code**	**L-T-P**	**Credits**
		0-0-0	2
Type of Course:	**INT**		
Pre-requisite(s), if any:			

The duration of the internship will be two weeks. It will be after the completion of 3rd Semester and before the commencement of Semester 5th Semester.

The following options can be opted for by the students:

1. Offline internship in industry - The student is supposed to produce a joining letter and relieving letter once the internship is over in case of an Offline internship in any industry.

2. Online internships – with organizations /institutions that are approved /supported/recommended by the All-India Council of Technical Education for Internship (like SWAYAM, NPTEL, Internshala etc.)

Report Submission and Evaluation Guidelines:

• Student must prepare a detailed report and submit the report. A copy of the report can be kept in the departments for record.

• Each student must be assigned a faculty as a mentor from the university and an Industry Expert as External Guide or Industry Mentor.

• The presentation by student for Internship/ project should in the presence of all students is desirable.

• Student should produce successful completion certificate in case of summer internship in industry.

Course Outcomes:

At the end of the course, students will be able to:

1. Get exposure to the industrial environment, which cannot be simulated in the classroom and hence creating competent professionals for the industry.

2. Get possible opportunities to learn, understand and sharpen the real-time technical / managerial skills required at the job(s).

3. Gain experience in writing technical reports / projects and presentation of them.

4. Learn and gain exposure to the engineer’s responsibilities and ethics.

5. Understand the social, economic, and administrative considerations that influence the working environment of industrial organizations.

EVEN SEMESTER (VI)							
Sr. No	**Course Code**	**Course Title**	**Category**	**L**	**T**	**P**	**C**
1		Department Elective I	Minor	4	-	-	4
2	ENCA302	Introduction to Computer Organization & Architecture	Major	3	1	-	4
3	ENCA304	Introduction to Computer Networks	Major	4	-	-	4
4	ENCA306	Basics of Neural Networks and Deep Learning	Major	4	-	-	4
5		Department Elective I Lab	Minor	-	-	2	1
6	ENCA352	Computer Networks Lab	Major	-	-	2	1
7	ENCA354	Neural Networks and Deep Learning Lab	Major	-	-	2	1
8	**SEC036**	Competitive Coding	SEC	-	-	4	2
9	ENSI352	Minor Project-II	Proj				2
TOTAL				15	1	10	23

Department:	**Department of Computer Science and Engineering**		
Course Name: **Introduction to Computer Organization & Architecture**	**Course Code**	**L-T-P**	**Credits**
	ENCA302	3-1-0	4
Type of Course:	Major		
Pre-requisite(s), if any: Concepts of Digital Electronics			

Course Outcomes (CO)

COs	Statements
CO1	Understand the basics of instruction sets and their impact on processor design
CO2	Demonstrate an understanding of the design of the functional units of a digital computer system
CO3	Evaluate cost performance and design trade-offs in designing and constructing a computer processor including memory.
CO4	Design a pipeline for consistent execution of instructions with minimum hazards
CO5	Manipulate representations of numbers stored in digital computers using I/O devices and store them into memory

Brief Syllabus:

Computer Organization & Architecture (COA) covers topics in computer architecture and organization focusing on multicore, graphics-processor unit (GPU), and heterogeneous SOC multiprocessor architectures and their implementation issues (architect's perspective). The objective of the course is to provide in-depth coverage of current and emerging trends in computer organization and architecture focusing on performance and the hardware/software interface. The course emphasis is on analysing fundamental issues in architecture design and their impact on application performance.

UNIT WISE DETAILS

Unit Number: 1 **Instruction** **No. of hours: 10**

Role of abstraction, basic functional units of a computer, A note on Moore’s law, Notion of IPC, and performance. Data representation and basic operations.

Unit Number: 2 **Instruction Set Architecture (RISC-V)** **No. of hours: 10**

CPU registers, instruction format and encoding, addressing modes, instruction set, instruction types, instruction decoding and execution, basic instruction cycle, Reduced Instruction Set Computer (RISC), Complex Instruction Set Computer (CISC), RISC-V instructions.

Unit Number: 3 **The Processor** **No. of hours: 10**

Revisiting clocking methodology, Amdahl's law, Building a data path and control, single cycle processor, multi-cycle processor, instruction pipelining.

Unit Number: 4 **Memory hierarchy, Storage and I/O** **No. of hours: 10**

SRAM/DRAM, locality of reference, Caching: different indexing mechanisms, Trade-offs related to block size, associativity, and cache size, concept of optimization, Average memory access time.

Introduction to magnetic disks (notion of tracks, sectors), flash memory. I/O mapped, and memory mapped I/O. I/O data transfer techniques: programmed I/O, Interrupt-driven I/O, and DMA.

***Self-Learning Components:**

1. BSim Documentation

References:

1. https://www.nand2tetris.org/
2. https://www.coursera.org/learn/computer-organization-design
3. https://www.geeksforgeeks.org/computer-organization-and-architecture-tutorials/
4. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-823-computer-system-architecture-fall-2005/

Please Note:

At least 5-10 % syllabus will be asked in end term exams from self-learning components

Text Book:

1. "Computer Organization and Design: The Hardware/Software Interface", David A. Patterson and John L. Hennessy, 5th Edition, Elsevier.

Reference Books:

1. "Computer Organization & Architecture", Smruti Ranjan Sarangi, McGraw Hill
2. "Computer System Architecture", Mano M. Morris, Pearson.
3. "Computer Organization and Embedded Systems", 6th Edition by Carl Hamacher, McGraHill Higher Education
4. "Computer Architecture and Organization", 3rd Edition by John P. Hayes, WCB/McGraw-Hill
5. "Computer Organization and Architecture: Designing for Performance", 10th Edition by William Stallings, Pearson Education.

Online References:

- https://learning.edx.org/course/course-v1:MITx+6.004.2x+3T2015/block-v1:MITx+6.004.2x+3T2015+type@sequential+block@c3s1/block-v1:MITx+6.004.2x+3T2015+type@vertical+block@c3s1v1
- RIPES: https://freesoft.dev/program/108505982
- GEM5: https://www.gem5.org/documentation/learning_gem5/introduction/

- CACTI: https://github.com/HewlettPackard/cacti
- PIN: https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-binary-instrumentation-tooldownloads.html
- TEJAS: https://www.cse.iitd.ac.in/~srsarangi/archbooksoft.html

XILINX(VHDL/Verilog tools): https://www.xilinx.com/support/university/students.html

Program and Course Outcome Mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO1 1	PO1 2	PSO1	PSO2	PSO3	PSO 4
ENCA302/ Introduction to Computer Organization & Architecture Organization &	CO1	3	-	-	-	-	-	-	-	-	-	-	-	2			3
	CO2	-	3	2	-	-	-	-	-	-	-	-	-	2	2		2
	CO3	-	-	-	3		-	-	-	-	-	-	3		2		
	CO4	-	-	3	-	-		-	-	-	-	-	-				3
	CO5	2	-	-	-	-	-	-	-	-	-	-	-		2		2

Department:	**Department of Computer Science and Engineering**		
Course Name: **Introduction to Computer Networks**	**Course Code**	**L-T-P**	**Credits**
	ENCA304	4-0-0	4
Type of Course:	Major		
Pre-requisite(s), if any:			

Define Course Outcomes (CO)

COs	Statements
CO1	Understand the fundamental concepts and principles of computer networks.
CO2	Demonstrate knowledge of network hardware and software components.
CO3	Develop skills in network administration and management.
CO4	Choose appropriate protocol for desired communication service.

Brief Syllabus:
This course provides a comprehensive study of computer networks, covering fundamental concepts, protocols, and technologies. It emphasizes hands-on learning and explores open-source tools commonly used in the field of computer networking. Through practical assignments and projects, students will gain a solid understanding of network design, implementation, security, and management.

UNIT WISE DETAILS

Unit Number: 1 **Evolution of Computer Networking** **No. of hours: 6**

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-end throughput.

Unit Number: 2 **Data Link Layer Design Issues** **No. of hours: 12**

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 **Introduction to Network Layer and Transport Services** **No. of hours: 12**

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 **Principles of Network Applications** **No. of hours: 12**

Application Layer: Domain Name Space (DNS), DDNS, TELNET, EMAIL, File Transfer Protocol (FTP), WWW, HTTP, SNMP, Bluetooth, Firewalls, Basic concepts of Cryptography.

***Self-Learning Components:**
https://gaia.cs.umass.edu/kurose_ross/videos/1/

Cisco Networking Academy: network fundamentals, routing and switching, and network security. They provide free learning materials and hands-on practice: https://www.netacad.com/

Open-Source Networking Tools and Technologies

- Open-source network monitoring tools (e.g., Nagios, Zabbix)
- Open-source network management tools (e.g., OpenNMS)
- Open-source network security tools (e.g., Snort, Suricata)

Text Book:

1. Computer Networks (Fifth Edition) – Andrew S. Tanenbaum (Prentice Hall of India)
2. Data communication and Networking (Fourth Edition)- Behrouz A Forouzan (Tata Mcgraw Hill)

Reference Books:

3. Computer Networking A Top-Down Approach (Fifth Edition)-James F. Kurose-Keith W. Ross (Pearson)

Computer Networks – Protocols, Standards and Interfaces (Second Edition) – UylessBlack (Prentice Hall of India Pvt. Ltd.)

Program and Course Outcome Mapping

Course Code and Title	**PO**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO 4**
ENCA304/ Introduction to Computer Networks	**CO1**	2	2	-	-	-	-	-	-	-	-	-	-	2	-	-	-
	CO2	-	2	2	-	-	-	-	-	-	-	-	-	-	-	-	-
	CO3	-	-	-	3	-	-	-	-	-	-	-	-	-	-	2	-
	CO4	-	-	-	3	-	-	-	-	-	-	-	-	-	-	-	-
	CO5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Department:	**Department of Computer Science and Engineering**		
Course Name: Computer Networks Lab	**Course Code**	**L-T-P**	**Credits**
	ENCA352	0-0-2	1
Type of Course:	Major		
Pre-requisite(s), if any:			

Defined Course Outcomes

COs	
CO 1	To gain hands-on experience working with network hardware, software, and tools.
CO 2	Network Configuration and Troubleshooting.
CO 3	Network Design and Implementation.
CO 4	To measure and evaluate network performance using tools and techniques.

Ex. No	Experiment Title	Mapped CO/COs
1	Create a simple network with multiple PCs, switches, and routers .	
2	Assign IP addresses to devices and configure basic connectivity.	
3	Test connectivity between PCs using ping and trace routes.	
4	Configure VLANs on switches and assign ports to specific VLANs.	
5	Enable inter-VLAN routing using a router or Layer 3 switch.	
6	Test connectivity between PCs in different VLANs.	
7	Set up a network with multiple routers.	
8	Configure static routes on routers to enable communication between networks.	
9	Verify routing tables and test connectivity between networks.	
10	Set up a network with a private IP address space.	
11	Configure NAT on a router to enable translation between private and public IP addresses.	
12	Test connectivity between devices on the private network and the Internet.	
13	Create a wireless network using access points and wireless clients.	
14	Simulate network issues such as connectivity problems, routing errors, or misconfigurations.	
15	Design and implement a network traffic monitoring.	
16	Setting up small computer networks and Hands on networking commands: Set up a small wired and wireless network of 2 to 4 computers using Hub/Switch/Access point.	
17	Write a program for error detection and correction for 7/8 bits ASCII codes using Hamming Codes.	
18	Write a program for error detection and correction for 7/8 bits ASCII codes using CRC.	

19	Write a program to simulate Go back N and Selective Repeat Modes of Sliding Window Protocol in peer-to-peer mode. Further extend it to real implementation of Flow Control over TCP protocol.	
20	Design and deploy TCP based Multithreaded HTTP client server for accessing student activity data in the institute.	
21	Design and deploy TCP based Multithreaded FTP client server to share institute level notices.	
22	Design and deploy TCP based Multithreaded Chat client server for your class.	
23	Design and deploy UDP based Multithreaded Chat client server for your class.	
24	Examining real-world network deployments.	
25	Case studies of network failures and their resolutions.	

Department:	**Department of Computer Science and Engineering**		
Course Name: **Basics of Neural Networks and Deep Learning**	**Course Code**	**L-T-P**	**Credits**
	ENCA306	4-0-0	4
Type of Course:	Major		
Pre-requisite(s), if any:			

Define Course Outcomes (CO)

COs	Statements
CO1	Understand the basic concepts of neural network and Biological Neural Network
CO2	Express proficiency in the handling of the architecture of Neural Network
CO3	Determine methods to create and manipulate Deep Neural Network
CO4	Identify commonly used operations involved in designing Deep Neural Network
CO5	Articulate Neural Networks, activation functions, Drop out, overfitting, and their use in programs.

Brief Syllabus:
The course begins with key concepts of neural networks, and feed-forward neural networks. The student gets an opportunity to learn the programming languages (TensorFlow) to design deep learning models. The student learns the concepts behind deep learning algorithms and their use cases.

UNIT WISE DETAILS

Unit Number: 1 **The Neural Network** **No. of hours: 10**

Data collection for Neural Network neural networks, information flow in the Human Brain, Architecture of the Human brain Understanding the differences in biological Neural Networks and ANN. Training a network: loss functions, activation functions.

Unit Number: 2 **Feedforward neural network** **No. of hours: 12**

Linear Models Neural network concept and its application areas Training a Neural network, how to determine hidden layers, recurrent neural, multi-layer neural network, Risk minimization, regularization, model selection, and practical optimization.

Unit Number: 3 **Deep Learning** **No. of hours: 10**

Feed Forward network, bias-variance in neural networks, Overfitting, dropouts, Gradient decent algorithm, Convolutional Neural Network, Recurrent Neural Network, RBF

Challenges in designing the best Neural Network ---**Self-learning**

Unit Number: 4 **Probabilistic Neural Network and Research** **No. of hours: 10**

Hopfield Net, Boltzmann machine, RBMs, Encoders and Auto encoders, Object recognition, computer vision, and pattern recognition.

Research areas in Probabilistic Neural Networks

***Self-Learning Components**:

Py Torch.

Reference Books:

1. Golub, G, H. and Van Loan, C, F, Matrix Computations, JHU Press,2013.
2. Satish Kumar, Neural Networks: A Classroom Approach, Tata McGraw-Hill Education, 2004.

Text Books

1 Good fellow, I., Bengio, Y, and Courville, A., Deep Learning, MIT Press, 2016.

Program and Course Outcome Mapping

Course Code and Title	**PO**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
ENCA306/ **Basics of Neural Networks and Deep Learning**	**CO1**	2	–	–	–	–	1	1	1	–	–	–	1	1	–	–	–
	CO2	2	2	-	–	-	–	–	–	–	2	1	–	2	2	3	–
	CO3	3	3	3	2	3	–	–	–	–	–	1	2	3	3	–	–
	CO4	2	2	2	–	2	–	–	–	–	–	–	–	2	2	2	2
	CO5	2	2	–	2	2	–	–	–	–	2	2	1	2	2	2	2

Department:	**Department of Computer Science and Engineering**		
Course Name: **Neural Networks and Deep Learning Lab**	**Course Code**	**L-T-P**	**Credits**
	ENCA354	0-0-2	1
Type of Course:	Major		
Pre-requisite(s), if any:			

Proposed Lab Experiments

Defined Course Outcomes

COs	Statements
CO 1	Acquire a practical understanding of neural networks and deep learning algorithms through hands-on lab experiments.
CO 2	Develop proficiency in implementing feedforward neural networks and understanding their underlying principles.
CO 3	Demonstrate the ability to create and manipulate deep neural networks for solving complex real-world problems.
CO 4	Analyze and evaluate the performance of neural network models using appropriate evaluation criteria.

Ex. No	Experiment Title	Mapped CO/COs
1	Familiarize students with the lab environment, software, and tools.	CO1
2	Compare the information flow in a simple biological neural network (such as a single neuron) with a corresponding ANN architecture. Analyze how information is processed and propagated through each system and identify similarities and differences.	CO1
3	Implement a basic neural network using a library/framework of choice.	CO2
4	Implement a program to read a dataset.	CO2
5	Train and test the neural network using a simple dataset to classify inputs.	CO2
6	Explore different activation functions and their effects on network performance.	CO1
7	Implementation of delta rule.	CO2
8	Implementation of the simple feed-forward network.	CO2
9	Explore different pre-trained models.	CO1
10	Analyze a pre-trained Neural Network	CO3
11	Common issues and errors encountered during deep learning experiments,	CO3

12	Experiment with different regularization techniques (e.g., regularization, dropout).	CO3
13	Troubleshooting strategies and debugging techniques for deep learning experiments.	CO3
14	Analyze how to design confusion metrics for a model	CO3
15	Investigate the impact of epoch while training a model in a neural network	CO3
16	Analyse and compare the performance of different loss functions for any pre-trained model	CO3
17	Implement a sequence-to-sequence model.	CO3
18	Investigate the impact of network architecture on information flow and learning capabilities.	CO4
19	Project related to the application of machine learning in healthcare.	CO4
20	Project related to the application of machine learning in business analysis.	CO4

Department:	**Department of Computer Science and Engineering**		
Course Name: **Competitive Coding**	**Course Code**	**L-T-P**	**Credits**
	SEC036	0-0-4	2
Type of Course:	SEC		
Pre-requisite(s), if any:			

Course Outcomes

CO1	Proficiency in Algorithms and Data Structures: Demonstrate proficiency in implementing and analyzing various algorithms and data structures commonly used in competitive programming.
CO2	Efficient Problem Solving: Develop the ability to analyze problem statements, design efficient algorithms, and write optimized code to solve competitive programming problems within time and memory constraints.
CO3	Algorithmic Thinking: Cultivate algorithmic thinking and problem-solving skills by identifying patterns, applying appropriate algorithms, and selecting optimal data structures for a given problem.
CO4	Code Optimization and Complexity Analysis: Apply strategies to optimize code and improve time and space complexity of solutions, considering factors such as algorithm selection, data structure usage, and efficient coding techniques.
CO5	Competitive Programming Skills: Gain familiarity with different online competitive programming platforms, participate in coding competitions, and develop strong problem-solving and critical thinking skills in a competitive programming environment.

Brief Syllabus:

Introduction to Competitive Coding, Data Structures and Algorithms, Time and Space Complexity Analysis, Problem Solving Techniques, Advanced Data Structures, Coding Paradigms, Online Judges and Contest Platforms, Tips and Tricks for Competitive Coding, Mock Contests and Practice Sessions, Self-Learning Components

Table of Contents

S.N	Experiment Index	COs
1	Introduction to Competitive Coding • Overview of competitive coding and its importance in the field of computer science. • Understanding the significance of problem-solving skills and algorithmic thinking in competitive coding.	CO1
2	Data Structures and Algorithms • Review of fundamental data structures: arrays, linked lists, stacks, queues, trees, graphs, and hash tables. • Study of essential algorithms: searching, sorting, recursion, dynamic programming, greedy algorithms, and graph algorithms.	CO1

3	Time and Space Complexity Analysis • Understanding time and space complexity of algorithms. • Analysis of algorithm efficiency and choosing the most optimal solutions.	CO2
4	Problem Solving Techniques • Introduction to problem-solving techniques like brute force, divide and conquer, backtracking, and more. • Practice in applying different techniques to solve a variety of programming problems.	CO3
5	Advanced Data Structures • Study of advanced data structures: heaps, priority queues, segment trees, trie, and advanced graph structures. • Understanding the use of these data structures in solving complex programming problems.	CO4
6	Coding Paradigms • Introduction to different coding paradigms: procedural programming, object-oriented programming, and functional programming. • Understanding the benefits and drawbacks of each paradigm in competitive coding.	CO5
7	Online Judges and Contest Platforms • Familiarization with popular online judge platforms like Codeforces, Topcoder, and LeetCode. • Practice solving problems from online contests and participating in coding competitions. List of suggested links to coding platforms ▪ Codeforces: https://codeforces.com/ ▪ Topcoder: https://www.topcoder.com/ ▪ AtCoder: https://atcoder.jp/ ▪ LeetCode: https://leetcode.com/ ▪ HackerRank: https://www.hackerrank.com/ ▪ CodeChef: https://www.codechef.com/ ▪ HackerEarth: https://www.hackerearth.com/ ▪ Project Euler: https://projecteuler.net/ ▪ UVa Online Judge: https://onlinejudge.org/ ▪ SPOJ (Sphere Online Judge): https://www.spoj.com/ ▪ Google Code Jam: https://codingcompetitions.withgoogle.com/codejam ▪ Kick Start by Google: https://codingcompetitions.withgoogle.com/kickstart ▪ ACM ICPC Live Archive: https://icpcarchive.ecs.baylor.edu/ ▪ A2 Online Judge: https://a2oj.com/ ▪ CodeSignal: https://codesignal.com/	CO5
8	Tips and Tricks for Competitive Coding • Learning effective coding techniques, shortcut methods, and best practices for competitive coding. • Developing strategies to optimize code, manage time, and improve problem-solving speed.	CO5

9	Mock Contests and Practice Sessions • Conducting mock contests and practice sessions to simulate real coding competitions. • Solving a wide range of problems to enhance coding skills and adaptability to different problem types.	CO5
	Self-Learning Component:	CO5
10	List of Suggested Competitive programming Courses: ▪ Competitive Programmer's Core Skills" by Coursera: This course covers fundamental algorithms and data structures used in competitive programming. Link: https://www.coursera.org/learn/competitive-programming-core-skills ▪ "Algorithms and Data Structures" by MIT OpenCourseWare: This course teaches essential algorithms and data structures for competitive programming. Link: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/ ▪ "Data Structures and Algorithms" by GeeksforGeeks: This course covers various data structures and algorithms commonly used in competitive programming. Link: https://practice.geeksforgeeks.org/courses/dsa-self-paced ▪ "Introduction to Competitive Programming" by NPTEL: This course introduces the basics of competitive programming and covers algorithms and problem-solving techniques. Link: https://onlinecourses.nptel.ac.in/noc21_cs07/ ▪ "Competitive Programming" by HackerRank: This course provides in-depth coverage of algorithms and data structures with hands-on coding exercises. Link: https://www.hackerrank.com/domains/tutorials/10-days-of-statistics ▪ "Advanced Data Structures and Algorithms" by Udemy: This course dives deeper into advanced data structures and algorithms for competitive programming. Link: https://www.udemy.com/course/advanced-data-structures-and-algorithms-in-java/ ▪ "Mastering Data Structures and Algorithms using C and C++" by Udemy: This course covers data structures and algorithms with a focus on problem-solving for coding interviews and competitive programming. Link: https://www.udemy.com/course/datastructurescncpp/ ▪ "Competitive Programming" by Coding Ninjas: This course provides comprehensive training in competitive programming, covering algorithms, data structures, and problem-solving techniques. Link: https://www.codingninjas.com/courses/online-competitive-programming-course ▪ "Algorithmic Toolbox" by Coursera: This course from the University of California San Diego covers algorithmic techniques and data structures for competitive programming. Link: https://www.coursera.org/learn/algorithmic-toolbox ▪ "Competitive Programming - From Beginner to Expert" by Udemy: This course offers a complete guide to competitive programming, starting from the basics and progressing to advanced topics. Link:	

https://www.udemy.com/course/competitive-programming-from-beginner-to-expert/
- Competitive Programming Essentials, Master Algorithms 2022 (Udemy)
  https://www.udemy.com/course/competitive-programming-algorithms-coding-minutes/
- The Bible of Competitive Programming & Coding Interviews

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

List of popular Competitive Programming Competitions:

1. ACM International Collegiate Programming Contest (ICPC): This is one of the most prestigious programming competitions for college students. Teams compete in solving a set of challenging algorithmic problems within a time limit. Website
2. Google Code Jam: Organized by Google, this annual coding competition challenges participants to solve algorithmic problems. It consists of multiple online rounds leading to a final onsite competition. Website
3. Facebook Hacker Cup: This annual coding competition by Facebook features multiple online rounds and an onsite final round. Participants solve algorithmic problems for a chance to win prizes. Website
4. Topcoder Open: Topcoder hosts this annual programming competition featuring algorithmic and design challenges. Participants compete for cash prizes and a chance to be recognized by industry experts. Website
5. International Olympiad in Informatics (IOI): IOI is an annual international programming competition for high school students. Participants solve algorithmic problems in a contest format. Website
6. AtCoder Grand Contest: AtCoder hosts this regular contest series featuring algorithmic programming challenges. Participants can compete individually or as a team. Website
7. Codeforces: Codeforces is a popular competitive programming platform that hosts regular contests. Participants compete in solving algorithmic problems and earn ratings based on their performance. Website
8. LeetCode Weekly Contests: LeetCode organizes weekly contests where participants can solve algorithmic problems and compete for rankings. Website
9. HackerRank Contests: HackerRank hosts various contests and challenges covering a wide range of programming topics. Participants can compete individually or as part of a team. Website
10. Kaggle Competitions: Kaggle is a platform for data science competitions, where participants solve real-world problems using machine learning and data analysis techniques. Website

****All students must participate in some competitions***

Suggested Books

1. "Competitive Programming 3" by Steven Halim and Felix Halim: This book is a comprehensive guide to competitive programming, covering algorithms, data

structures, problem-solving techniques, and contest strategies. It includes numerous examples, explanations, and practice problems. Book Link

2. "Algorithms" by Robert Sedgewick and Kevin Wayne: This book provides a thorough introduction to algorithms, including sorting, searching, graph algorithms, and dynamic programming. It includes detailed explanations, visualizations, and implementation examples. Book Link
3. "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: Known as "CLRS," this book is a classic reference for algorithms. It covers a wide range of algorithms, data structures, and algorithm design techniques. Book Link
4. "Programming Challenges" by Steven S. Skiena and Miguel A. Revilla: This book presents a collection of programming problems from various competitions and online judges. It provides problem-solving techniques, algorithmic approaches, and example solutions. Book Link
5. "The Art of Computer Programming" by Donald E. Knuth: This multi-volume series is considered a classic in computer science. It covers various algorithms, data structures, and mathematical techniques in great detail. Book Link
6. "Cracking the Coding Interview" by Gayle Laakmann McDowell: Although not specifically focused on competitive programming, this book is a popular resource for coding interview preparation. It covers essential data structures, algorithms, and problem-solving techniques. Book Link
7. "Programming Pearls" by Jon Bentley: This book presents a collection of programming challenges and discusses techniques for solving them efficiently. It emphasizes problem-solving skills and algorithmic thinking. Book Link

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://www.udemy.com/course/competitive-programming
- https://github.com/smv1999/CompetitiveProgrammingQuestionBank
- https://github.com/parikshit223933/Coding-Ninjas-Competitive-Programming
- https://www.hackerearth.com/getstarted-competitive-programming/

https://www.csestack.org/competitive-coding-questions/

List of Suggested Experiments in Lab Sessions

Questions on Arrays

1. Maximum Subarray Sum: Given an array of integers, find the contiguous subarray with the largest sum.
2. Two Sum: Given an array of integers and a target value, find two numbers in the array that add up to the target.
3. Rotate Array: Rotate an array of n elements to the right by k steps.
4. Merge Intervals: Given a collection of intervals, merge overlapping intervals.
5. Majority Element: Find the majority element in an array. The majority element appears more than n/2 times, where n is the size of the array.

6. Trapping Rain Water: Given an array representing the heights of bars, calculate the amount of water that can be trapped between the bars.
7. Next Permutation: Implement the next permutation algorithm to find the lexicographically next greater permutation of an array of integers.
8. Subarray with Given Sum: Given an unsorted array of non-negative integers and a target sum, find a subarray that adds up to the target sum.
9. Product of Array Except Self: Given an array of n integers, return an array output such that each element at index i of the output array is the product of all the elements in the original array except the one at i.
10. Minimum Size Subarray Sum: Given an array of positive integers and a target sum, find the minimum length of a contiguous subarray whose sum is greater than or equal to the target sum.

Questions on Recursion

1. Factorial: Write a recursive function to calculate the factorial of a given number.
2. Fibonacci Series: Write a recursive function to generate the nth term of the Fibonacci series.
3. Power of a Number: Write a recursive function to calculate the power of a given number.
4. Sum of Digits: Write a recursive function to find the sum of digits of a given number.
5. Palindrome Check: Write a recursive function to check whether a given string is a palindrome or not.
6. Tower of Hanoi: Solve the Tower of Hanoi problem using recursion.
7. Binary Search: Implement a recursive binary search algorithm to find an element in a sorted array.
8. Permutations: Write a recursive function to generate all permutations of a given string.
9. Subset Sum: Given an array of integers and a target sum, write a recursive function to check if there exists a subset that sums up to the target.
10. Combination Sum: Given an array of integers and a target sum, write a recursive function to find all possible combinations that sum up to the target.

Questions on Stacks & Queues:

1. Balanced Parentheses: Given a string of parentheses, write a function to determine if the parentheses are balanced using a stack.
2. Reverse a String: Write a function to reverse a string using a stack.
3. Evaluate Postfix Expression: Given a postfix expression, write a function to evaluate it using a stack.
4. Next Greater Element: Given an array, find the next greater element for each element in the array using a stack.
5. Largest Rectangle in Histogram: Given a histogram represented by an array of bar heights, find the largest rectangle that can be formed in the histogram using a stack.
6. Implement Stack using Queues: Implement a stack data structure using queues.
7. Implement Queue using Stacks: Implement a queue data structure using stacks.
8. Sliding Window Maximum: Given an array and an integer k, find the maximum element in each sliding window of size k using a queue.

9. Print Binary Tree in Level Order: Given a binary tree, print its elements in level order using a queue.
10. Implement Recent Counter: Design a data structure that counts the number of recent requests within a certain time range using a queue.

Questions on Linked Lists

1. Reverse a Linked List: Write a function to reverse a singly linked list.
2. Detect Cycle in a Linked List: Write a function to detect if a linked list contains a cycle.
3. Find the Middle of a Linked List: Write a function to find the middle node of a linked list.
4. Merge Two Sorted Lists: Given two sorted linked lists, write a function to merge them into a single sorted linked list.
5. Remove Nth Node from End of List: Given a linked list, remove the nth node from the end of the list and return its head.
6. Intersection of Two Linked Lists: Given two linked lists, write a function to find the intersection point if it exists.
7. Palindrome Linked List: Given a singly linked list, determine if it is a palindrome.
8. Remove Duplicates from Sorted List: Given a sorted linked list, remove duplicates from it.
9. Add Two Numbers as Linked Lists: Given two linked lists representing two numbers, write a function to add them and return the resulting linked list.
10. Flatten a Multilevel Linked List: Given a linked list with a special structure, flatten it into a single-level linked list.

Questions on Trees

1. Binary Tree Traversals: Implement different tree traversal algorithms such as in-order, pre-order, and post-order traversal.
2. Maximum Depth of Binary Tree: Find the maximum depth or height of a binary tree.
3. Validate Binary Search Tree: Given a binary tree, check if it is a valid binary search tree.
4. Lowest Common Ancestor of Two Nodes: Find the lowest common ancestor of two nodes in a binary tree.
5. Diameter of Binary Tree: Find the diameter of a binary tree, which is the longest path between any two nodes.
6. Binary Tree Level Order Traversal: Traverse a binary tree in level order and return the nodes in each level.
7. Symmetric Tree: Check if a binary tree is symmetric, meaning it is a mirror image of itself.
8. Serialize and Deserialize Binary Tree: Design algorithms to serialize and deserialize a binary tree.
9. Count Complete Tree Nodes: Count the number of nodes in a complete binary tree.
10. Construct Binary Tree from Preorder and Inorder Traversal: Given the preorder and inorder traversal of a binary tree, construct the tree.

Questions on Graphs

- Shortest path: Find the shortest path between two vertices in a graph. This can be solved using Dijkstra's algorithm or Bellman-Ford's algorithm.
- Maximum flow: Find the maximum flow from one vertex to another in a graph. This can be solved using the Ford-Fulkerson algorithm or the Dinic algorithm.
- Minimum spanning tree: Find the minimum spanning tree of a graph. This can be solved using Prim's algorithm or Kruskal's algorithm.
- Topological sorting: Find a topological ordering of a graph. This can be solved using Kahn's algorithm.
- Strongly connected components: Find the strongly connected components of a graph. This can be solved using Tarjan's algorithm.
- Bipartite matching: Find a maximum bipartite matching in a graph. This can be solved using the Hungarian algorithm.
- Traveling salesman problem: Find the shortest tour that visits all the vertices in a graph. This is an NP-hard problem, but there are approximation algorithms that can be used to find a good solution.

Time & Space Complexity

1. Time Complexity Analysis: Analyze the time complexity of a given algorithm or piece of code.
2. Space Complexity Analysis: Analyze the space complexity of a given algorithm or piece of code.
3. Big O Notation: Given a function or algorithm, determine its big O notation in terms of time or space complexity.
4. Best/Worst/Average Case Complexity: Analyze the best, worst, and average-case time or space complexity of an algorithm.
5. Sorting Algorithms: Implement and analyze the time complexity of various sorting algorithms such as Bubble Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort.
6. Searching Algorithms: Implement and analyze the time complexity of various searching algorithms such as Linear Search, Binary Search, and Hashing.
7. Dynamic Programming: Solve dynamic programming problems and analyze their time and space complexity.
8. Recursion vs. Iteration: Compare and analyze the time and space complexity of recursive and iterative solutions for a given problem.
9. Complexity Trade-offs: Analyze and compare the time and space complexity trade-offs of different algorithms for the same problem.
10. Space-Optimized Data Structures: Implement and analyze space-optimized data structures such as Bit Arrays, Bloom Filters, or Space-Efficient Hash Tables.

Questions on Divide & Conquer Strategy

1. Binary Search: Implement a recursive binary search algorithm to find an element in a sorted array.
2. Merge Sort: Implement the Merge Sort algorithm to sort an array of integers.

3. Quick Sort: Implement the Quick Sort algorithm to sort an array of integers.
4. Count Inversions: Given an array of integers, find the number of inversions present using the Divide and Conquer approach.
5. Closest Pair of Points: Given a set of points in a 2D plane, find the pair of points with the smallest distance between them using the Divide and Conquer technique.
6. Maximum Subarray Sum: Given an array of integers, find the maximum sum of a subarray using the Divide and Conquer approach.
7. Matrix Multiplication: Implement a Divide and Conquer algorithm to multiply two matrices efficiently.
8. Finding Majority Element: Given an array of integers, find the majority element (appearing more than n/2 times) using the Divide and Conquer technique.
9. Finding Kth Smallest Element: Given an array of integers, find the kth smallest element using the Divide and Conquer approach.
10. Closest Pair Sum: Given two sorted arrays and a target value, find the pair of elements (one from each array) with the closest sum to the target using the Divide and Conquer technique.

Questions on Dynamic Programming

1. Fibonacci Series: Implement the Fibonacci series using dynamic programming to efficiently calculate the nth term.
2. Longest Common Subsequence: Given two strings, find the length of the longest common subsequence using dynamic programming.
3. Knapsack Problem: Given a set of items with weights and values, determine the maximum value that can be obtained by selecting a subset of items within a weight limit using dynamic programming.
4. Coin Change Problem: Given a set of coin denominations and a target value, find the minimum number of coins needed to make the target value using dynamic programming.
5. Rod Cutting Problem: Given a rod of a certain length and a price list for different rod lengths, find the maximum value that can be obtained by cutting and selling the rod using dynamic programming.
6. Edit Distance: Given two strings, find the minimum number of operations (insertion, deletion, and substitution) required to convert one string into another using dynamic programming.
7. Maximum Subarray Sum: Given an array of integers, find the maximum sum of a subarray using dynamic programming.
8. Longest Increasing Subsequence: Given an array of integers, find the length of the longest increasing subsequence using dynamic programming.
9. Matrix Chain Multiplication: Given a sequence of matrices, find the minimum number of scalar multiplications needed to multiply them using dynamic programming.
10. Subset Sum Problem: Given a set of integers and a target sum, determine if there exists a subset that sums up to the target using dynamic programming.

Questions on Greedy Programming

1. Fractional Knapsack Problem: Given a set of items with weights and values, determine the maximum value that can be obtained by selecting fractions of items within a weight limit using a greedy algorithm.
2. Activity Selection Problem: Given a set of activities with start and finish times, select the maximum number of activities that can be performed without overlapping using a greedy algorithm.
3. Minimum Spanning Tree: Given a weighted graph, find the minimum spanning tree using Kruskal's or Prim's algorithm, which are both based on greedy approaches.
4. Huffman Coding: Given a set of characters and their frequencies, construct a binary code that minimizes the total encoded length using a greedy algorithm.
5. Coin Change Problem: Given a set of coin denominations and a target value, find the minimum number of coins needed to make the target value using a greedy algorithm.
6. Job Scheduling Problem: Given a set of jobs with their deadlines and profits, schedule the jobs to maximize the total profit using a greedy algorithm.
7. Interval Scheduling Problem: Given a set of intervals, select the maximum number of non-overlapping intervals using a greedy algorithm.
8. Dijkstra's Algorithm: Given a weighted graph, find the shortest path from a source vertex to all other vertices using Dijkstra's algorithm, which is based on a greedy approach.
9. Egyptian Fraction: Given a fraction, represent it as a sum of unique unit fractions using a greedy algorithm.
10. Car Fueling Problem: Given the total distance to be covered, the capacity of the fuel tank, and a list of distances between fuel stations, determine the minimum number of refueling needed to reach the destination using a greedy algorithm.

Questions on String Matching

1. Naive String Matching: Implement the naive string-matching algorithm to find all occurrences of a pattern in a text.
2. Knuth-Morris-Pratt (KMP) Algorithm: Implement the KMP algorithm to efficiently find all occurrences of a pattern in a text.
3. Rabin-Karp Algorithm: Implement the Rabin-Karp algorithm to efficiently find all occurrences of a pattern in a text using hashing.
4. Longest Common Substring: Given two strings, find the longest common substring using dynamic programming or other efficient algorithms.
5. Longest Common Prefix: Given an array of strings, find the longest common prefix using a suitable algorithm.
6. Regular Expression Matching: Implement a regular expression matching algorithm to determine if a string matches a given pattern.
7. Anagrams: Given a list of strings, find all pairs of strings that are anagrams of each other.
8. Palindromic Substrings: Given a string, find all palindromic substrings using a suitable algorithm.
9. Boyer-Moore Algorithm: Implement the Boyer-Moore algorithm to efficiently find all occurrences of a pattern in a text.
10. Subsequence Matching: Given two strings, determine if one string is a subsequence of the other.

Questions on Advanced Data Structures

1. Trie: Implement a Trie data structure and solve problems such as word search, autocomplete, or finding the longest common prefix.
2. Segment Tree: Implement a Segment Tree data structure and solve problems such as range sum queries, range minimum/maximum queries, or range updates.
3. Fenwick Tree (Binary Indexed Tree): Implement a Fenwick Tree data structure and solve problems such as prefix sum queries or range updates.
4. Disjoint Set Union (DSU) / Union-Find: Implement a DSU data structure and solve problems such as connected components, cycle detection, or Kruskal's algorithm for finding the minimum spanning tree.
5. Treap: Implement a Treap (a balanced binary search tree with randomized priorities) and solve problems such as maintaining the median of a dynamic set of numbers or solving range queries on a set of intervals.
6. Suffix Array: Implement a Suffix Array data structure and solve problems such as finding the longest common substring, finding the lexicographically smallest substring, or pattern matching.
7. LCA (Lowest Common Ancestor): Implement an LCA data structure and solve problems such as finding the lowest common ancestor of two nodes in a tree or solving distance-related queries on a tree.
8. K-D Tree: Implement a K-D Tree data structure and solve problems such as nearest neighbor search or range search in a multi-dimensional space.
9. AVL Tree or Red-Black Tree: Implement a balanced binary search tree (either AVL Tree or Red-Black Tree) and solve problems such as maintaining a sorted dynamic set or solving range queries.
10. B+ Tree: Implement a B+ Tree data structure and solve problems such as indexing or range queries on a large dataset.

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/
- https://unstop.com/blog/competitive-coding-questions-with-solutions
- https://unstop.com/blog/competitive-coding-questions-with-solutions

Department	**Department of Computer Science and Engineering**		
Course Name- Competitive Programming Lab	**Course Code**	**L-T-P**	**Credits**
	SEC036	0-0-4	2
Type of Course	Skill Enhancement Course (SEC)		
Pre-requisite(s), if any: None			

Brief Syllabus:
Introduction to Competitive Coding, Data Structures, and Algorithms, Time and Space Complexity Analysis, Problem Solving Techniques, Advanced Data Structures, Coding Paradigms, Online Judges and Contest Platforms, Tips and Tricks for Competitive Coding, Mock Contests and Practice Sessions, Self-Learning Components

CO1	Proficiency in Algorithms and Data Structures: Demonstrate proficiency in implementing and analyzing various algorithms and data structures commonly used in competitive programming.
CO2	Efficient Problem Solving: Develop the ability to analyze problem statements, design efficient algorithms, and write optimized code to solve competitive programming problems within time and memory constraints.
CO3	Algorithmic Thinking: Cultivate algorithmic thinking and problem-solving skills by identifying patterns, applying appropriate algorithms, and selecting optimal data structures for a given problem.
CO4	Code Optimization and Complexity Analysis: Apply strategies to optimize code and improve time and space complexity of solutions, considering factors such as algorithm selection, data structure usage, and efficient coding techniques.
CO5	Competitive Programming Skills: Gain familiarity with different online competitive programming platforms, participate in coding competitions, and develop strong problem-solving and critical thinking skills in a competitive programming environment.

S.N	Experiment Index	COs
1	Introduction to Competitive Coding • Overview of competitive coding and its importance in the field of computer science. • Understanding the significance of problem-solving skills and algorithmic thinking in competitive coding.	CO1
2	Data Structures and Algorithms • Review of fundamental data structures: arrays, linked lists, stacks, queues, trees, graphs, and hash tables. • Study of essential algorithms: searching, sorting, recursion, dynamic programming, greedy algorithms, and graph algorithms.	CO1
	Time and Space Complexity Analysis	

3	• Understanding the time and space complexity of algorithms. • Analysis of algorithm efficiency and choosing the most optimal solutions.	CO2
4	Problem Solving Techniques • Introduction to problem-solving techniques like brute force, divide and conquer, backtracking, and more. • Practice in applying different techniques to solve a variety of programming problems.	CO3
5	Advanced-Data Structures • Study of advanced data structures: heaps, priority queues, segment trees, trie, and advanced graph structures. • Understanding the use of these data structures in solving complex programming problems.	CO4
6	Coding Paradigms • Introduction to different coding paradigms: procedural programming, object-oriented programming, and functional programming. • Understanding the benefits and drawbacks of each paradigm in competitive coding.	CO5
7	Online Judges and Contest Platforms • Familiarization with popular online judge platforms like Codeforces, Topcoder, and LeetCode. • Practice solving problems from online contests and participating in coding competitions. List of suggested links to coding platforms ▪ Codeforces: https://codeforces.com/ ▪ Topcoder: https://www.topcoder.com/ ▪ AtCoder: https://atcoder.jp/ ▪ LeetCode: https://leetcode.com/ ▪ HackerRank: https://www.hackerrank.com/ ▪ CodeChef: https://www.codechef.com/ ▪ HackerEarth: https://www.hackerearth.com/ ▪ Project Euler: https://projecteuler.net/ ▪ UVa Online Judge: https://onlinejudge.org/ ▪ SPOJ (Sphere Online Judge): https://www.spoj.com/ ▪ Google Code Jam: https://codingcompetitions.withgoogle.com/codejam ▪ Kick Start by Google: https://codingcompetitions.withgoogle.com/kickstart ▪ ACM ICPC Live Archive: https://icpcarchive.ecs.baylor.edu/ ▪ A2 Online Judge: https://a2oj.com/ ▪ CodeSignal: https://codesignal.com/	CO5
8	Tips and Tricks for Competitive Coding • Learning effective coding techniques, shortcut methods, and best practices for competitive coding. • Developing strategies to optimize code, manage time, and improve problem-solving speed.	CO5
	Mock Contests and Practice Sessions	CO5

9	• Conduct mock contests and practice sessions to simulate real coding competitions. • Solving a wide range of problems to enhance coding skills and adaptability to different problem types.	
10	**Self-Learning Component:** List of Suggested Competitive Programming Courses: ▪ Competitive Programmer's Core Skills" by Coursera: This course covers fundamental algorithms and data structures used in competitive programming. Link: https://www.coursera.org/learn/competitive-programming-core-skills ▪ "Algorithms and Data Structures" by MIT OpenCourseWare: This course teaches essential algorithms and data structures for competitive programming. Link: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/ ▪ "Data Structures and Algorithms" by GeeksforGeeks: This course covers various data structures and algorithms commonly used in competitive programming. Link: https://practice.geeksforgeeks.org/courses/dsa-self-paced ▪ "Introduction to Competitive Programming" by NPTEL: This course introduces the basics of competitive programming and covers algorithms and problem-solving techniques. Link: https://onlinecourses.nptel.ac.in/noc21_cs07/ ▪ "Competitive Programming" by HackerRank: This course provides in-depth coverage of algorithms and data structures with hands-on coding exercises. Link: https://www.hackerrank.com/domains/tutorials/10-days-of-statistics ▪ "Advanced Data Structures and Algorithms" by Udemy: This course dives deeper into advanced data structures and algorithms for competitive programming. Link: https://www.udemy.com/course/advanced-data-structures-and-algorithms-in-java/ ▪ "Mastering Data Structures and Algorithms using C and C++" by Udemy: This course covers data structures and algorithms with a focus on problem-solving for coding interviews and competitive programming. Link: https://www.udemy.com/course/datastructurescncpp/ ▪ "Competitive Programming" by Coding Ninjas: This course provides comprehensive training in competitive programming, covering algorithms, data structures, and problem-solving techniques. Link: https://www.codingninjas.com/courses/online-competitive-programming-course ▪ "Algorithmic Toolbox" by Coursera: This course from the University of California San Diego covers algorithmic techniques and data structures for competitive programming. Link: https://www.coursera.org/learn/algorithmic-toolbox ▪ "Competitive Programming - From Beginner to Expert" by Udemy: This course offers a complete guide to competitive programming, starting from the basics and progressing to advanced topics. Link: https://www.udemy.com/course/competitive-programming-from-beginner-to-expert/ ▪ Competitive Programming Essentials, Master Algorithms 2022 (Udemy)	CO5

	https://www.udemy.com/course/competitive-programming-algorithms-coding-minutes/ ▪ The Bible of Competitive Programming & Coding Interviews **All students must complete one online course from the suggested programs*	

List of popular Competitive Programming Competitions:

11. ACM International Collegiate Programming Contest (ICPC): This is one of the most prestigious programming competitions for college students. Teams compete in solving a set of challenging algorithmic problems within a time limit. Website
12. Google Code Jam: Organized by Google, this annual coding competition challenges participants to solve algorithmic problems. It consists of multiple online rounds leading to a final onsite competition. Website
13. Facebook Hacker Cup: This annual coding competition by Facebook features multiple online rounds and an onsite final round. Participants solve algorithmic problems for a chance to win prizes. Website
14. Topcoder Open: Topcoder hosts this annual programming competition featuring algorithmic and design challenges. Participants compete for cash prizes and a chance to be recognized by industry experts. Website
15. International Olympiad in Informatics (IOI): IOI is an annual international programming competition for high school students. Participants solve algorithmic problems in a contest format. Website
16. AtCoder Grand Contest: AtCoder hosts this regular contest series featuring algorithmic programming challenges. Participants can compete individually or as a team. Website
17. Codeforces: Codeforces is a popular competitive programming platform that hosts regular contests. Participants compete in solving algorithmic problems and earn ratings based on their performance. Website
18. LeetCode Weekly Contests: LeetCode organizes weekly contests where participants can solve algorithmic problems and compete for rankings. Website
19. HackerRank Contests: HackerRank hosts various contests and challenges covering a wide range of programming topics. Participants can compete individually or as part of a team. Website
20. Kaggle Competitions: Kaggle is a platform for data science competitions, where participants solve real-world problems using machine learning and data analysis techniques. Website

****All students must participate in some competitions***

Suggested Books

8. "Competitive Programming 3" by Steven Halim and Felix Halim: This book is a comprehensive guide to competitive programming, covering algorithms, data structures, problem-solving techniques, and contest strategies. It includes numerous examples, explanations, and practice problems. Book Link
9. "Algorithms" by Robert Sedgewick and Kevin Wayne: This book provides a thorough introduction to algorithms, including sorting, searching, graph algorithms, and dynamic programming. It includes detailed explanations, visualizations, and implementation examples. Book Link

10. "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: Known as "CLRS," this book is a classic reference for algorithms. It covers a wide range of algorithms, data structures, and algorithm design techniques. Book Link
11. "Programming Challenges" by Steven S. Skiena and Miguel A. Revilla: This book presents a collection of programming problems from various competitions and online judges. It provides problem-solving techniques, algorithmic approaches, and example solutions. Book Link
12. "The Art of Computer Programming" by Donald E. Knuth: This multi-volume series is considered a classic in computer science. It covers various algorithms, data structures, and mathematical techniques in great detail. Book Link
13. "Cracking the Coding Interview" by Gayle Laakmann McDowell: Although not specifically focused on competitive programming, this book is a popular resource for coding interview preparation. It covers essential data structures, algorithms, and problem-solving techniques. Book Link
14. "Programming Pearls" by Jon Bentley: This book presents a collection of programming challenges and discusses techniques for solving them efficiently. It emphasizes problem-solving skills and algorithmic thinking. Book Link

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://www.udemy.com/course/competitive-programming
- https://github.com/smv1999/CompetitiveProgrammingQuestionBank
- https://github.com/parikshit223933/Coding-Ninjas-Competitive-Programming
- https://www.hackerearth.com/getstarted-competitive-programming/
- https://www.csestack.org/competitive-coding-questions/

List of Suggested Experiments in Lab Sessions

Questions on Arrays

11. Maximum Subarray Sum: Given an array of integers, find the contiguous subarray with the largest sum.
12. Two Sum: Given an array of integers and a target value, find two numbers in the array that add up to the target.
13. Rotate Array: Rotate an array of n elements to the right by k steps.
14. Merge Intervals: Given a collection of intervals, merge overlapping intervals.
15. Majority Element: Find the majority element in an array. The majority element appears more than n/2 times, where n is the size of the array.
16. Trapping Rain Water: Given an array representing the heights of bars, calculate the amount of water that can be trapped between the bars.
17. Next Permutation: Implement the next permutation algorithm to find the lexicographically next greater permutation of an array of integers.
18. Subarray with Given Sum: Given an unsorted array of non-negative integers and a target sum, find a subarray that adds up to the target sum.
19. Product of Array Except Self: Given an array of n integers, return an array output such that each element at index i of the output array is the product of all the elements in the original array except the one at i.

20. Minimum Size Subarray Sum: Given an array of positive integers and a target sum, find the minimum length of a contiguous subarray whose sum is greater than or equal to the target sum.

Questions on Recursion

11. Factorial: Write a recursive function to calculate the factorial of a given number.
12. Fibonacci Series: Write a recursive function to generate the nth term of the Fibonacci series.
13. Power of a Number: Write a recursive function to calculate the power of a given number.
14. Sum of Digits: Write a recursive function to find the sum of digits of a given number.
15. Palindrome Check: Write a recursive function to check whether a given string is a palindrome or not.
16. Tower of Hanoi: Solve the Tower of Hanoi problem using recursion.
17. Binary Search: Implement a recursive binary search algorithm to find an element in a sorted array.
18. Permutations: Write a recursive function to generate all permutations of a given string.
19. Subset Sum: Given an array of integers and a target sum, write a recursive function to check if there exists a subset that sums up to the target.
20. Combination Sum: Given an array of integers and a target sum, write a recursive function to find all possible combinations that sum up to the target.

Questions on Stacks & Queues:

11. Balanced Parentheses: Given a string of parentheses, write a function to determine if the parentheses are balanced using a stack.
12. Reverse a String: Write a function to reverse a string using a stack.
13. Evaluate Postfix Expression: Given a postfix expression, write a function to evaluate it using a stack.
14. Next Greater Element: Given an array, find the next greater element for each element in the array using a stack.
15. Largest Rectangle in Histogram: Given a histogram represented by an array of bar heights, find the largest rectangle that can be formed in the histogram using a stack.
16. Implement Stack using Queues: Implement a stack data structure using queues.
17. Implement Queue using Stacks: Implement a queue data structure using stacks.
18. Sliding Window Maximum: Given an array and an integer k, find the maximum element in each sliding window of size k using a queue.
19. Print Binary Tree in Level Order: Given a binary tree, print its elements in level order using a queue.
20. Implement Recent Counter: Design a data structure that counts the number of recent requests within a certain time range using a queue.

Questions on Linked Lists

11. Reverse a Linked List: Write a function to reverse a singly linked list.
12. Detect Cycle in a Linked List: Write a function to detect if a linked list contains a cycle.
13. Find the Middle of a Linked List: Write a function to find the middle node of a linked list.
14. Merge Two Sorted Lists: Given two sorted linked lists, write a function to merge them into a single sorted linked list.

15. Remove Nth Node from End of List: Given a linked list, remove the nth node from the end of the list and return its head.
16. Intersection of Two Linked Lists: Given two linked lists, write a function to find the intersection point if it exists.
17. Palindrome Linked List: Given a singly linked list, determine if it is a palindrome.
18. Remove Duplicates from Sorted List: Given a sorted linked list, remove duplicates from it.
19. Add Two Numbers as Linked Lists: Given two linked lists representing two numbers, write a function to add them and return the resulting linked list.
20. Flatten a Multilevel Linked List: Given a linked list with a special structure, flatten it into a single-level linked list.

Questions on Trees

11. Binary Tree Traversals: Implement different tree traversal algorithms such as in-order, pre-order, and post-order traversal.
12. Maximum Depth of Binary Tree: Find the maximum depth or height of a binary tree.
13. Validate Binary Search Tree: Given a binary tree, check if it is a valid binary search tree.
14. Lowest Common Ancestor of Two Nodes: Find the lowest common ancestor of two nodes in a binary tree.
15. Diameter of Binary Tree: Find the diameter of a binary tree, which is the longest path between any two nodes.
16. Binary Tree Level Order Traversal: Traverse a binary tree in level order and return the nodes in each level.
17. Symmetric Tree: Check if a binary tree is symmetric, meaning it is a mirror image of itself.
18. Serialize and Deserialize Binary Tree: Design algorithms to serialize and deserialize a binary tree.
19. Count Complete Tree Nodes: Count the number of nodes in a complete binary tree.
20. Construct Binary Tree from Preorder and Inorder Traversal: Given the preorder and inorder traversal of a binary tree, construct the tree.

Questions on Graphs

- Shortest path: Find the shortest path between two vertices in a graph. This can be solved using Dijkstra's algorithm or Bellman-Ford's algorithm.
- Maximum flow: Find the maximum flow from one vertex to another in a graph. This can be solved using the Ford-Fulkerson algorithm or the Dinic algorithm.
- Minimum spanning tree: Find the minimum spanning tree of a graph. This can be solved using Prim's algorithm or Kruskal's algorithm.
- Topological sorting: Find a topological ordering of a graph. This can be solved using Kahn's algorithm.
- Strongly connected components: Find the strongly connected components of a graph. This can be solved using Tarjan's algorithm.
- Bipartite matching: Find a maximum bipartite matching in a graph. This can be solved using the Hungarian algorithm.

- Traveling salesman problem: Find the shortest tour that visits all the vertices in a graph. This is an NP-hard problem, but there are approximation algorithms that can be used to find a good solution.

Time & Space Complexity

11. Time Complexity Analysis: Analyze the time complexity of a given algorithm or piece of code.
12. Space Complexity Analysis: Analyze the space complexity of a given algorithm or piece of code.
13. Big O Notation: Given a function or algorithm, determine its big O notation in terms of time or space complexity.
14. Best/Worst/Average Case Complexity: Analyze the best, worst, and average-case time or space complexity of an algorithm.
15. Sorting Algorithms: Implement and analyze the time complexity of various sorting algorithms such as Bubble Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort.
16. Searching Algorithms: Implement and analyze the time complexity of various searching algorithms such as Linear Search, Binary Search, and Hashing.
17. Dynamic Programming: Solve dynamic programming problems and analyze their time and space complexity.
18. Recursion vs. Iteration: Compare and analyze the time and space complexity of recursive and iterative solutions for a given problem.
19. Complexity Trade-offs: Analyze and compare the time and space complexity trade-offs of different algorithms for the same problem.
20. Space-Optimized Data Structures: Implement and analyze space-optimized data structures such as Bit Arrays, Bloom Filters, or Space-Efficient Hash Tables.

Questions on Divide & Conquer Strategy

11. Binary Search: Implement a recursive binary search algorithm to find an element in a sorted array.
12. Merge Sort: Implement the Merge Sort algorithm to sort an array of integers.
13. Quick Sort: Implement the Quick Sort algorithm to sort an array of integers.
14. Count Inversions: Given an array of integers, find the number of inversions present using the Divide and Conquer approach.
15. Closest Pair of Points: Given a set of points in a 2D plane, find the pair of points with the smallest distance between them using the Divide and Conquer technique.
16. Maximum Subarray Sum: Given an array of integers, find the maximum sum of a subarray using the Divide and Conquer approach.
17. Matrix Multiplication: Implement a Divide and Conquer algorithm to multiply two matrices efficiently.
18. Finding Majority Element: Given an array of integers, find the majority element (appearing more than n/2 times) using the Divide and Conquer technique.
19. Finding Kth Smallest Element: Given an array of integers, find the kth smallest element using the Divide and Conquer approach.
20. Closest Pair Sum: Given two sorted arrays and a target value, find the pair of elements (one from each array) with the closest sum to the target using the Divide and Conquer technique.

Questions on Dynamic Programming

11. Fibonacci Series: Implement the Fibonacci series using dynamic programming to efficiently calculate the nth term.
12. Longest Common Subsequence: Given two strings, find the length of the longest common subsequence using dynamic programming.
13. Knapsack Problem: Given a set of items with weights and values, determine the maximum value that can be obtained by selecting a subset of items within a weight limit using dynamic programming.
14. Coin Change Problem: Given a set of coin denominations and a target value, find the minimum number of coins needed to make the target value using dynamic programming.
15. Rod Cutting Problem: Given a rod of a certain length and a price list for different rod lengths, find the maximum value that can be obtained by cutting and selling the rod using dynamic programming.
16. Edit Distance: Given two strings, find the minimum number of operations (insertion, deletion, and substitution) required to convert one string into another using dynamic programming.
17. Maximum Subarray Sum: Given an array of integers, find the maximum sum of a subarray using dynamic programming.
18. Longest Increasing Subsequence: Given an array of integers, find the length of the longest increasing subsequence using dynamic programming.
19. Matrix Chain Multiplication: Given a sequence of matrices, find the minimum number of scalar multiplications needed to multiply them using dynamic programming.
20. Subset Sum Problem: Given a set of integers and a target sum, determine if there exists a subset that sums up to the target using dynamic programming.

Questions on Greedy Programming

11. Fractional Knapsack Problem: Given a set of items with weights and values, determine the maximum value that can be obtained by selecting fractions of items within a weight limit using a greedy algorithm.
12. Activity Selection Problem: Given a set of activities with start and finish times, select the maximum number of activities that can be performed without overlapping using a greedy algorithm.
13. Minimum Spanning Tree: Given a weighted graph, find the minimum spanning tree using Kruskal's or Prim's algorithm, which are both based on greedy approaches.
14. Huffman Coding: Given a set of characters and their frequencies, construct a binary code that minimizes the total encoded length using a greedy algorithm.
15. Coin Change Problem: Given a set of coin denominations and a target value, find the minimum number of coins needed to make the target value using a greedy algorithm.
16. Job Scheduling Problem: Given a set of jobs with their deadlines and profits, schedule the jobs to maximize the total profit using a greedy algorithm.
17. Interval Scheduling Problem: Given a set of intervals, select the maximum number of non-overlapping intervals using a greedy algorithm.
18. Dijkstra's Algorithm: Given a weighted graph, find the shortest path from a source vertex to all other vertices using Dijkstra's algorithm, which is based on a greedy approach.
19. Egyptian Fraction: Given a fraction, represent it as a sum of unique unit fractions using a greedy algorithm.
20. Car Fueling Problem: Given the total distance to be covered, the capacity of the fuel tank, and a list of distances between fuel stations, determine the minimum number of refueling needed to reach the destination using a greedy algorithm.

Questions on String Matching

11. Naive String Matching: Implement the naive string-matching algorithm to find all occurrences of a pattern in a text.
12. Knuth-Morris-Pratt (KMP) Algorithm: Implement the KMP algorithm to efficiently find all occurrences of a pattern in a text.
13. Rabin-Karp Algorithm: Implement the Rabin-Karp algorithm to efficiently find all occurrences of a pattern in a text using hashing.
14. Longest Common Substring: Given two strings, find the longest common substring using dynamic programming or other efficient algorithms.
15. Longest Common Prefix: Given an array of strings, find the longest common prefix using a suitable algorithm.
16. Regular Expression Matching: Implement a regular expression matching algorithm to determine if a string matches a given pattern.
17. Anagrams: Given a list of strings, find all pairs of strings that are anagrams of each other.
18. Palindromic Substrings: Given a string, find all palindromic substrings using a suitable algorithm.
19. Boyer-Moore Algorithm: Implement the Boyer-Moore algorithm to efficiently find all occurrences of a pattern in a text.
20. Subsequence Matching: Given two strings, determine if one string is a subsequence of the other.

Questions on Advanced Data Structures

11. Trie: Implement a Trie data structure and solve problems such as word search, autocomplete, or finding the longest common prefix.
12. Segment Tree: Implement a Segment Tree data structure and solve problems such as range sum queries, range minimum/maximum queries, or range updates.
13. Fenwick Tree (Binary Indexed Tree): Implement a Fenwick Tree data structure and solve problems such as prefix sum queries or range updates.
14. Disjoint Set Union (DSU) / Union-Find: Implement a DSU data structure and solve problems such as connected components, cycle detection, or Kruskal's algorithm for finding the minimum spanning tree.
15. Treap: Implement a Treap (a balanced binary search tree with randomized priorities) and solve problems such as maintaining the median of a dynamic set of numbers or solving range queries on a set of intervals.
16. Suffix Array: Implement a Suffix Array data structure and solve problems such as finding the longest common substring, finding the lexicographically smallest substring, or pattern matching.
17. LCA (Lowest Common Ancestor): Implement an LCA data structure and solve problems such as finding the lowest common ancestor of two nodes in a tree or solving distance-related queries on a tree.
18. K-D Tree: Implement a K-D Tree data structure and solve problems such as nearest neighbor search or range search in a multi-dimensional space.
19. AVL Tree or Red-Black Tree: Implement a balanced binary search tree (either AVL Tree or Red-Black Tree) and solve problems such as maintaining a sorted dynamic set or solving range queries.
20. B+ Tree: Implement a B+ Tree data structure and solve problems such as indexing or range queries on a large dataset.

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/
- https://unstop.com/blog/competitive-coding-questions-with-solutions
- https://unstop.com/blog/competitive-coding-questions-with-solutions

Department:	**Department of Computer Science and Engineering**		
Course Name: Minor Project-II	**Course Code**	**L-T-P**	**Credits**
	ENSI352	---	2
Type of Course:	Project		
Pre-requisite(s), if any: NA			

- Students expected to develop a basic project that demonstrates the application of learnings from studied subjects.
- Students are required to submit a hard copy of project file as per the template. File needs to be submitted in spiral bind.
- Project will be evaluated on the scale of 100 with following evaluation criteria.
    - Project idea & features (10)
    - Literature review (10)
    - Tools & Techniques employed (10)
    - Methodology (10)
    - Presentation of Results and their usefulness (20)
    - Implementation and its understandability (10)
    - Meetings & comments by guide (20)
    - Research paper (10)

File format for Minor project

1.	Abstract	Page No.
2.	Introduction (description of broad topic)	
3.	Motivation	
4.	Literature Review	
5.	Gap Analysis	
6.	Problem Statement	
7.	Objectives	
8.	Tools/platform used	
9.	Methodology	
10.	Experimental Setup	
11.	Evaluation Metrics	
12.	Results and Discussion	

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

Department:	Department of Computer Science and Engineering		
Course Name:	Course Code	L-T-P	Credits
Image Processing & Computer Vision	ENSP304	4-0-0	4
Type of Course:	Minor (Department Elective I)		
Pre-requisite(s), if any: (1) Linear Algebra and (2) programming in Python			

COs	Statements
CO1	Understand the fundamental concepts and techniques of image processing.
CO2	Apply image enhancement techniques for improving image quality.
CO3	Analyze the impact of different image enhancement techniques on image quality and visual perception.
CO4	Evaluate the strengths and limitations of computer vision techniques in various applications
CO5	Develop innovative image fusion techniques for combining multiple images to enhance visual perception

Brief Syllabus:
The syllabus for the subject "Image Processing and Computer Vision using Python" covers the following topics: introduction to image processing and computer vision, Python programming basics for image processing, image acquisition and manipulation using Python libraries, image enhancement techniques, image filtering and convolution, feature extraction, and object detection, image segmentation and boundary detection, image registration and alignment, camera calibration and 3D reconstruction, deep learning for image classification and object recognition, and applications of computer vision in fields like robotics, healthcare, and autonomous systems. The syllabus emphasizes hands-on programming exercises and projects to develop practical skills in implementing image processing and computer vision algorithms using Python.

UNIT WISE DETAILS

Unit Number: 1 **Introduction to Basic Concepts of Image Formation** **No. of hours: 10**

Fundamentals and Applications of image processing, Image processing system components, Image sensing and acquisition, Sampling and quantization, Neighbors of pixel adjacency connectivity, regions and boundaries, Distance measures.

Image Enhancement: Frequency and Spatial Domain, Contrast Stretching, Histogram Equalization, Low pass and High pass filtering.

Unit Number: 2 **Image Restoration and coloring** **No. of hours: 10**

Model of The Image Degradation Restoration Process, Noise Models, Restoration in the presence of Noise Only Spatial Filtering, Periodic Noise Reduction by Frequency Domain Filtering, Linear Position Invariant Degradations, Estimation of Degradation Function, Inverse filtering, Wiener filtering, Constrained Least Square Filtering, Geometric Mean Filter, Geometric Transformations.
Colour Image Processing, Image Segmentation, Texture Descriptors, Colour Features, Edges/Boundaries, Object Boundary and Shape Representations, Interest or Corner Point Detectors, Speeded up Robust Features, and Saliency.

Unit Number: 3	**Image Compression and Segmentation**	**No. of hours: 10**

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, Threshold, Region-based Segmentation Introduction to Morphology, Dilation, Erosion, Some basic Morphological Algorithms Object

Unit Number: 4	**Object Representation and Computer Vision Techniques**	**No. of hours: 10**

Representation and Description and Computer Vision Techniques: Introduction to Morphology, Some Basic Morphological Algorithms, Representation, Boundary Descriptors, Regional Descriptors, Chain Code, and Structural Methods. Review of Computer Vision applications; Artificial Neural Networks for Pattern Classification, Convolutional Neural Networks, Machine Learning Algorithms and their Applications in Image Segmentation, Motion Estimation and Object Tracking, Gesture Recognition, Face and Facial Expression Recognition, Image Fusion

***Self-Learning Components:**
Please Note:

1. **Concepts of Huffman coding, arithmetic coding, and other compression algorithms.**
2. **Presenting an overview of image compression standards (e.g., JPEG, JPEG2000) and their performance characteristics.**
3. **Presentation on a specific computer vision application (e.g., gesture recognition, facial expression recognition) and the underlying algorithms used.**

Note:
1)Students are supposed to learn the components on self-basis
2) At least 5-10 % syllabus will be asked in end term exams from self-learning components.

Reference Books:

1. **Gonzalez Rafael C. and Woods Richard E., Digital Image Processing, New Delhi: Prentice–Hall of India.**
2. **M.K. Bhuyan , " Computer Vision and Image Processing: Fundamentals and Applications", CRC Press, USA, ISBN 9780815370840 - CAT# K338147**
3. **MOOCs course by Prof. M. K. Bhuyan, "Computer Vision and Image Processing – Fundamentals and Applications"https://onlinecourses.nptel.ac.in/noc21_ee23/course**

Program and course outcome mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO1 1	PO1 2	PSO1	PSO2	PSO3	PSO 4
ENSP304/Image Processing & Computer Vision	CO1	3	1	-	-	1	-	-	-	1	1	-	2	3	1	-	-
	CO2	1	1	3	-	2	-	-	-	1	-	-	2	2	3	-	-
	CO3	1	2	1	3	2	-	-	-	1	-	-	2	2	2	1	1
	CO4	-	2	3	-	3	-	-	-	1	-	-	2	2	3	2	2
	CO5	-	-	3	1	1	1	-	-	2	1	1	2	2	3	2	2

Department:	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Image Processing & Computer Vision Lab	**ENSP354**	0-0-2	1
Type of Course:	Minor (Department Elective I)		
Pre-requisite(s), if any: (1) Linear Algebra and (2) programming in python			

COs	Statements
CO 1	Apply image processing techniques using Python libraries.
CO 2	Analyze and evaluate the effectiveness of different image enhancement algorithms
CO 3	Implement image restoration algorithms and evaluate their performance in the presence of noise.
CO 4	Develop image compression algorithms and analyze their impact on image quality.
CO 5	Formulate computer vision techniques such as object detection and tracking, gesture recognition, and facial expression recognition using Python.

Ex. No	**Experiment Title**	**Mapped CO/COs**
1	Image acquisition and display using the Open CV library	CO 1
2	Image enhancement techniques: contrast stretching, histogram equalization	CO 2
3	Low-pass and high-pass filtering for image smoothing and sharpening	CO 2
4	Image degradation and restoration: modeling degradation process, noise reduction	CO 3
5	Inverse filtering and Wiener filtering for image restoration	CO 3
6	Geometric mean filter for image denoising	CO 3
7	Geometric transformations: translation, rotation, scaling	CO 1
8	Color image processing: color space conversion, histogram-based operations	CO 2
9	Image segmentation using thresholding techniques	CO 1
10	Texture analysis and feature extraction	CO 2
11	Edge detection and boundary extraction	CO 2
12	Interest point detection using Harris corner detector	CO 2
13	Speeded up robust features (SURF) for feature extraction	CO 2
14	Saliency detection in images	CO 2
15	Lossless and lossy image compression using Huffman coding	CO 4
16	Shanon-Fano coding and arithmetic coding for image compression	CO 4
17	Golomb coding and LZW coding for data compression	CO 4
18	Run-length coding for image compression	CO 4
19	Lossless predictive coding for image compression	CO 4
20	Bit plane coding for image compression	CO 4
21	Image segmentation based on discontinuity and similarity	CO 1
22	Edge linking and boundary detection in images	CO 1
23	Morphological operations: dilation and erosion	CO 1
24	Object representation and description using morphological algorithms	CO 1
25	Introduction to computer vision applications	CO 1

26	Pattern classification using artificial neural networks	CO 5
27	Convolutional neural networks for image classification	CO 5
28	Machine learning algorithms for image segmentation	CO 5
29	Motion estimation and object tracking	CO 5
30	Gesture recognition and face/facial expression recognition	CO 5

Detailed syllabus

Session 1: Image acquisition and display using the OpenCV library

- Session: Introduction to image acquisition and display using OpenCV library
- Exercise: Write a Python code to capture and display images using OpenCV
- Project: Build a simple application to capture images from a webcam and display them in real-time

Session 2: Image enhancement techniques: contrast stretching, histogram equalization

- Session: Introduction to image enhancement techniques
- Exercise: Implement contrast stretching and histogram equalization algorithms in Python
- Project: Apply image enhancement techniques on a set of images and compare the results

Session 3: Low-pass and high-pass filtering for image smoothing and sharpening

- Session: Understanding low-pass and high-pass filters for image processing
- Exercise: Implement low-pass and high-pass filters in Python for image smoothing and sharpening
- Project: Apply filters on a set of images and analyze the effects of smoothing and sharpening

Session 4: Image degradation and restoration: modeling degradation process, noise reduction

- Session: Introduction to image degradation and restoration
- Exercise: Model image degradation process and implement noise reduction techniques
- Project: Restore a set of degraded images using various restoration methods

Session 5: Inverse filtering and Wiener filtering for image restoration

- Session: Understanding inverse filtering and Wiener filtering for image restoration
- Exercise: Implement inverse filtering and Wiener filtering algorithms in Python
- Project: Apply restoration techniques on a set of images and evaluate the performance

Session 6: Geometric mean filter for image denoising

- Session: Introduction to geometric mean filter for image denoising
- Exercise: Implement geometric mean filter in Python for denoising images
- Project: Apply the filter on noisy images and compare the results with other denoising techniques

Session 7: Geometric transformations: translation, rotation, scaling

- Session: Introduction to geometric transformations in image processing
- Exercise: Implement translation, rotation, and scaling operations on images using OpenCV
- Project: Apply geometric transformations on a set of images and analyze the transformations' effects

Session 8: Color image processing: color space conversion, histogram-based operations

- Session: Understanding color image processing techniques
- Exercise: Perform color space conversion and histogram-based operations on images
- Project: Apply color image processing techniques on a set of images and analyze the results

Session 9: Image segmentation using thresholding techniques

- Session: Introduction to image segmentation using thresholding techniques
- Exercise: Implement thresholding algorithms for image segmentation in Python
- Project: Segment images using various thresholding methods and evaluate the segmentation results

Session 10: Texture analysis and feature extraction

- Session: Understanding texture analysis and feature extraction methods
- Exercise: Extract texture features from images using texture analysis algorithms
- Project: Apply texture analysis and feature extraction techniques on images and analyze the extracted features

Session 11: Edge detection and boundary extraction

- Session: Introduction to edge detection and boundary extraction
- Exercise: Implement edge detection algorithms in Python
- Project: Detect edges and extract boundaries from a set of images using different edge detection methods

Session 12: Interest point detection using Harris corner detector

- Session: Understanding interest point detection using Harris corner detector
- Exercise: Implement Harris corner detection algorithm in Python
- Project: Detect interest points in images and analyze their properties using the Harris corner detector

Session 13: Speeded up robust features (SURF) for feature extraction

- Session: Introduction to SURF (Speeded Up Robust Features) algorithm
- Exercise: Implement SURF algorithm for feature extraction in Python
- Project: Extract features from images using SURF and evaluate their robustness and speed

Session 14: Saliency detection in images

- Session: Understanding saliency detection in images
- Exercise: Implement saliency detection algorithm in Python
- Project: Detect salient regions in images and analyze their significance using the implemented algorithm

Session 15: Lossless and lossy image compression using Huffman coding

- Session: Introduction to image compression using Huffman coding
- Exercise: Implement Huffman coding for lossless image compression in Python
- Project: Compress a set of images using Huffman coding and evaluate the compression ratio and quality

Session 16: Shanon-Fano coding and arithmetic coding for image compression

- Session: Understanding Shanon-Fano coding and arithmetic coding for image compression
- Exercise: Implement Shanon-Fano coding and arithmetic coding algorithms in Python
- Project: Compare the performance of Shanon-Fano coding and arithmetic coding for image compression

Session 17: Golomb coding and LZW coding for data compression

- Session: Introduction to Golomb coding and LZW (Lempel-Ziv-Welch) coding for data compression
- Exercise: Implement Golomb coding and LZW coding algorithms in Python
- Project: Apply Golomb coding and LZW coding on data and analyze the compression efficiency

Session 18: Run-length coding for image compression

- Session: Understanding run-length coding for image compression
- Exercise: Implement run-length coding algorithm in Python
- Project: Compress images using run-length coding and analyze the compression performance

Session 19: Lossless predictive coding for image compression

- Session: Introduction to lossless predictive coding for image compression
- Exercise: Implement lossless predictive coding algorithm in Python
- Project: Apply predictive coding on images and evaluate the compression results

Session 20: Bit plane coding for image compression

- Session: Understanding bit plane coding for image compression

- Exercise: Implement bit plane coding algorithm in Python
- Project: Apply bit plane coding on images and analyze the compression efficiency

Session 21: Image segmentation based on discontinuity and similarity

- Session: Introduction to image segmentation based on discontinuity and similarity
- Exercise: Implement image segmentation algorithms using discontinuity and similarity measures
- Project: Segment images based on different segmentation criteria and evaluate the results

Session 22: Edge linking and boundary detection in images

- Session: Understanding edge linking and boundary detection in images
- Exercise: Implement edge linking algorithms for boundary detection in Python
- Project: Detect and link edges to extract boundaries from images using various edge linking methods

Session 23: Morphological operations: dilation and erosion

- Session: Introduction to morphological operations in image processing
- Exercise: Implement dilation and erosion operations using morphological algorithms
- Project: Apply morphological operations on images to analyze their effects on different objects

Session 24: Object representation and description using morphological algorithms

- Session: Understanding object representation and description using morphological algorithms
- Exercise: Implement object representation and description techniques using morphological operations
- Project: Represent and describe objects in images using morphological algorithms and analyze the results

Session 25: Introduction to computer vision applications

- Session: Overview of computer vision applications and use cases
- Exercise: Explore different computer vision applications and their functionalities
- Project: Choose a specific computer vision application, implement it, and demonstrate its capabilities

Session 26: Pattern classification using artificial neural networks

- Session: Introduction to pattern classification using artificial neural networks
- Exercise: Implement an artificial neural network for pattern classification in Python
- Project: Train a neural network model to classify patterns and evaluate its performance

Session 27: Convolutional neural networks for image classification

- Session: Understanding convolutional neural networks (CNNs) for image classification
- Exercise: Implement a CNN architecture in Python for image classification
- Project: Train a CNN model on a dataset for image classification and evaluate its accuracy

Session 28: Machine learning algorithms for image segmentation

- Session: Introduction to machine learning algorithms for image segmentation
- Exercise: Implement machine learning algorithms for image segmentation in Python
- Project: Apply machine learning techniques for image segmentation and analyze the segmentation results

Session 29: Motion estimation and object tracking

- Session: Understanding motion estimation and object tracking techniques
- Exercise: Implement motion estimation and object tracking algorithms in Python
- Project: Track objects in video sequences using motion estimation and analyze the tracking performance

Session 30: Gesture recognition and face/facial expression recognition

- Session: Introduction to gesture recognition and face/facial expression recognition
- Exercise: Implement gesture recognition and face/facial expression recognition algorithms in Python
- Project: Develop a system that can recognize gestures and facial expressions from video input

Department:	**Department of Computer Science and Engineering**		
Course Name: **Introduction to Generative AI**	**Course Code**	**L-T-P**	**Credits**
	ENSP306	4-0-0	4
Type of Course:	Minor (Department Elective I)		
Pre-requisite(s), if any:			

COs	Statements
CO1	Understand the foundational concepts of Generative AI
CO2	Apply probability distributions and random variables in generative models
CO3	Employ various generative models, such as VAEs, GANs, and flow-based models, to generate new data samples in different domains.
CO4	Implement and Analyze generative models
CO5	Evaluate emerging trends and future directions in the field of Generative AI
CO6	Develop sequence generation models using recurrent neural networks (RNNs) and LSTM

Brief Syllabus:
This course introduces students to the fundamental concepts and techniques of Generative Artificial Intelligence (AI). Generative AI is an emerging field that focuses on developing algorithms and models capable of generating new content, such as images, music, and text. The course will cover the theoretical foundations of generative models and provide hands-on experience with open-source tools for creating and exploring generative AI applications.

UNIT WISE DETAILS

Unit Number: 1 Foundations of Generative AI No. of hours: 10

Introduction to Generative AI: Definition, working and applications of generative AI, Historical overview and recent advancements, Ethical considerations and societal impact.
Probability and Statistics for Generative AI: Probability distributions and random variables, Maximum likelihood estimation, Bayesian inference and generative models.
Fundamentals of Deep Learning: Neural networks and their architectures, Backpropagation and optimization algorithms, Transfer learning and pre-trained models.

Unit Number: 2 Generative Models No. of hours: 10

Overview of generative models: Gaussian Mixture Models, Hidden Markov Models; Representation learning and latent variables; Autoencoders: Basics of autoencoders and their applications, Encoder and decoder architectures, Reconstruction loss and latent space representation; Variational autoencoders (VAEs): Introduction to VAEs, reparameterization;

Unit Number: 3 **Generative Adversarial Networks and Flow-based Models** **No. of hours: 10**

Generative Adversarial Networks (GANs)**:** Introduction, Architecture of GANs, Training GANs and understanding the loss functions; Autoregressive Models (including information-theoretic foundations)
Flow-based generative models and their advantages, Normalizing flows and invertible transformations, Training and sampling from flow-based models, Evaluation of Generative Models: Metrics for evaluating generative models (log-likelihood, Inception Score)

Unit Number: 4 **Applications and Future Directions** **No. of hours: 10**

Real-World Applications of Generative AI: Image synthesis and editing, Data augmentation and data generation, Generative AI in healthcare, gaming, and art; **Ethical Considerations and Challenges:** Bias and fairness in generative models, Deepfakes and misinformation, Responsible AI practices; Emerging **Trends and Future Directions:** Reinforcement learning and generative models, Meta-learning and few-shot generation, Open AI's DALL-E.

***Self-Learning Components:**

- Students are encouraged to explore and familiarize themselves with the tools of Python programming language for machine learning (NumPy, Pandas, PyTorch)
- Experiment with popular open-source tools: TensorFlow and Keras
- Presentation on current research areas like: style transfer, multimodal generation, and unsupervised learning.
- Open source tools for image: CycleGAN for image translation, StyleGAN and StyleGAN2 for high-quality image synthesis, OpenAI's CLIP for cross-model understanding
- Course on "Introduction to Generative AI" with Google Cloud

Please Note:
1) Students are supposed to learn the components on self-basis
2) At least 5-10 % syllabus will be asked in end term exams from self-learning components.

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
4. Natural Language Processing with Python" by Steven Bird, Ewan Klein, and Edward Loper

ference Links:

- Deep Learning Specialization on Coursera (includes a course on generative models):https://www.coursera.org/specializations/deep-learning
- TensorFlow Tutorials on Generative Models: https://www.tensorflow.org/tutorials/generative
- OpenAI's Generative Models page: https://openai.com/research/generative-models/

Program and course outcome mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENSP306/ Introduction to Generative AI	CO1	3	1	-	-	1	-	-	-	1	1	-	2	3	-	-	-
	CO2	1	1	3	-	2	-	-	-	1	-	-	2	3	3	-	-
	CO3	1	2	1	3	2	-	-	-	1	-	-	2	3	3	3	-
	CO4	-	2	3	-	3	-	-	-	1	-	-	2	-	3	-	-
	CO5	-	-	3	1	1	1	-	-	2	1	1	2	3	3	-	-

Department:	**Department of 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:	Minor (Department Elective II)		
Pre-requisite(s), if any: NA			

COs	Statements
CO 1	Utilize Python programming to generate random samples from various probability distributions
CO 2	Apply knowledge of generative AI models and frameworks
CO 3	Develop proficiency in building and training feedforward neural networks and deep learning frameworks
CO 4	Implement basic autoencoder models and train them on datasets
CO5	Evaluate the performance metrics of trained models, such as accuracy and loss

Ex. No	**Experiment Title**	**Mapped CO/COs**
1	Generate random samples from various probability distributions (e.g., normal distribution, uniform distribution) using Python	CO1
2	Implement maximum likelihood estimation (MLE) for a given dataset and estimate the parameters of a selected probability distribution.	CO1
3	Explore and experiment with existing generative AI models and frameworks (e.g., TensorFlow, PyTorch).	CO2
4	Implement a basic generative AI model (e.g., a simple image generator) using a chosen framework.	CO2
5	Implement a feedforward neural network using a deep learning framework (e.g., TensorFlow, PyTorch).	CO3
6	Train the neural network on a benchmark dataset (e.g., MNIST, CIFAR-10) using backpropagation and a chosen optimization algorithm (e.g., stochastic gradient descent).	CO3
7	Evaluate the trained model's performance metrics (e.g., accuracy, loss) on a separate validation set.	CO5
8	Compare and analyze the performance of the trained model with and without transfer learning.	CO5
9	Train an autoencoder on a dataset of images.	CO4
10	Encode a set of images using the trained encoder and visualize their corresponding latent space representations	CO4
11	Build an encoder and a decoder architecture for a VAE using a deep learning framework.	CO4
12	Train the VAE on a dataset of images (e.g., MNIST, CIFAR-10) using a chosen loss function	CO4
13	Implement a basic autoencoder model and train it on a dataset.	CO4

14	Implement an autoregressive model, such as PixelCNN or PixelRNN, using a deep learning framework.	CO3
15	Implement a GAN architecture using a deep learning framework.	CO3
16	Train the GAN on a dataset of images (e.g., MNIST, CIFAR-10) and monitor the generator and discriminator losses.	CO5
17	Analyze the loss functions used in GAN training (e.g., adversarial loss, feature matching loss)	CO5
18	Train an RNN-based model to generate sequences (e.g., text or music)	CO3
19	Train the RNN on a dataset of sequences (e.g., text corpus, MIDI data) using backpropagation through time (BPTT)	CO3
20	Implement a flow-based generative model using a deep learning framework	CO3
21	Fine-tune a pre-trained deep learning model on a new task or dataset.	CO3
22	Implement the Tacotron model using a deep learning framework.	CO3
23	Implement the CycleGAN model using a deep learning framework.	CO3
24	Implement the evaluation metrics using appropriate libraries or frameworks.	CO5
25	Evaluate the performance of different generative models using the implemented metrics.	CO5

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

Department:	**Department of Computer Science and Engineering**		
Course Name: **Mobile Application Development using iOS**	**Course Code**	**L-T-P**	**Credits**
	ENSP409	4-0-0	4
Type of Course:	Minor (Department Elective III)		
Pre-requisite(s), if any: Basics of Android			

COs	Statements
CO1	Create iPhone apps using Objective-C and Apple's new programming language, use industry tools and frameworks such as Cocoa, Xcode, UIKit, Git.
CO2	Understand and know how to use properly UIKit, asynchronous code, Core Image, NSURL Session and JSON Map Kit and Core Location, Auto Layout, Source Control, Core Data, Animation, and the app submission process.
CO3	Read and write programs based on Objective-C, also have a strong grasp of Objective-C objects
CO4	Organize their code professionally using objects and blocks, prototype several entries- level apps and try to publish on App store.

Brief Syllabus:
The objective of the course is to provide skills to develop applications for OS X and iOS. It includes introduction to development framework Xcode. Objective-C is used as programming language to develop the 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.

UNIT WISE DETAILS

Unit Number: 1 **Title: Introduction to IDE and SDK of iOS App Development** **No. of hours: 10**

Xcode-The SDK environment, Supporting tools, Advance settings. Development Technique, Fundamental of Object-Oriented Programming, The MVC architecture.

Unit Number: 2 **Title: Objective-C** **No. of hours: 10**

Introduction to Objective C, Primitive Data Types, Conditions, Loops, Functions, Arrays, Pointers, Structures, Classes, Objects, Foundation, Memory Management, Inheritance, Categories, Protocols, Predicates, Blocks, Multi-Threading.

Objects Send and Receive Messages concept, Use of Pointers to Keep Track of Objects, Methods - Return Values.

Unit Number: 3 Title: Encapsulating Data No. of hours:10

Content Summary:

Properties of Encapsulation of an Object's Values, Declare Public Properties for Exposed Data, Use Accessor Methods to Get or Set Property Values, Concept of Dot Syntax, Properties Are Backed by Instance Variables.

Dealing with Errors: Use NSError for Most Errors, Some Delegate Methods Alert You to Errors, Some Methods Pass Errors by Reference

Unit Number: 4 Title: Developing iOS Applications No. of hours: 10

Content Summary:
iOS App Anatomy, Design Principles, creating a Basic Hello World App with interface elements, UI View & Controller, UI Elements, Trigger Actions, Storyboard, Device Orientations, Using Gestures, Popovers and Modal Dialogs, Creating Universal Apps, Status Bar, Navigation Bar, Tab Bar, Content Views (e.g., Image view, Map View etc.), UI Table View and Table View Controller, Core Data, test your App, Publishing your App.

***Self-Learning Components:**

2. XCode Documentation

References:

5. https://www.tutorialspoint.com/objective_c/objective_c_quick_guide.htm
6. https://www.coursera.org/learn/introduction-to-ios-mobile-application-development
7. https://www.geeksforgeeks.org/classes-objects-in-objective-c/

Please Note:
1)Students are supposed to learn the components on self-basis
2) At least 5-10 % syllabus will be asked in end term exams from self-learning components.
Textbook:
1. Effective objective C 2.0, Matt Galloway, Effective software development series, Scott Meyers.
Reference Books:

2. Programming in Objective-C (5th Edition) (Developer's Library) by Stephen G. Kochan.

3. iOS 6 Development Unleashed: Developing Mobile Applications for Apple iPhone, iPad, and iPod Touch by Robert McGovern

Online References:

- https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html
- https://www.digitalocean.com/community/tutorials/objective-c-hello-world-tutorial

Program and course outcome mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO1 1	PO1 2	PSO1	PSO2	PSO3	PSO 4
ENSP409/ Mobile Application Development using iOS	**CO1**	3	1	2	-	-	-	-	-	-	-	-	3	2		-	3
	CO2	-	3	2	-	2	-	-	-	-	-	-	3	2	2	-	2
	CO3	-	2	2	3	1	-	-	-	-	-	-	3	-	2	-	-
	CO4	-	2	3	-	-		-	-	-	-	-	2	-	-	-	3
	CO5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Department:	**Department of 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:	Minor (Department Elective III)		
Pre-requisite(s), if any: Basics of Android			

Proposed Lab Experiments

Defined Course Outcomes

COs	Statements
CO 1	**Create** iPhone apps using Objective-C and Apple's new programming language, use industry tools and frameworks such as Cocoa, Xcode, UIKit, Git.
CO 2	**Understand** and know how to use properly UIKit, asynchronous code, Core Image, NSURL Session and JSON Map Kit and Core Location, Auto Layout, Source Control, Core Data, Animation, and the app submission process.
CO 3	**Read** and **write** programs based on Objective-C, also have a strong grasp of Objective-C objects
CO 4	**Organize** their code professionally using objects and blocks, prototype several entries- level apps and try to publish on App store.

Ex. No.	**Experiment Title**	**Mapped CO/COs**
1	Case Study of Objective-C language.	CO2
2	Case study of Windows and MAC systems	CO2
3	Case Study of XCode based on MAC Systems	CO2
4	Design an App for UISwitch based on Objective-C language	CO1
5	Design an App for UISlider based on Objective-C language	CO1
6	Design an App for UIStepper based on Objective-C language	CO1
7	Write a program for creating Story Boards	CO1
8	Design an App for UIAnimation based on Objective-C language	CO1

9	Create a Simple Calculator using Objective-C Language	CO1
10	Design an App for UIProgress Bar based on Objective-C language	CO1
11	Design an App for UIDatePicker Bar based on Objective-C language	CO1
12	Write an Objective-C program to print factorial of a given number	CO3
13	Write an Objective-C program to print Fibonacci series	CO3
14	Write an Objective-C program that displays the Phrase “Hello World”	CO3
15	Write an Objective-C program for displaying the value of variables	CO3
16	Write an Objective-C program for displaying the sum and subtraction of two variables	CO3
17	Write an Objective-C program for displaying the multiplication and division of the two variables	CO3
18	Write an Objective-C program that demonstrate control structure of Objective-C language	CO3
19	Create a Button using Objective-C	CO3
20	Write an Objective-C program to print the value of a variable inside a text, place it in parentheses, and insert a backslash just prior to the opening parenthesis.	CO3
21	Write an Objective-C program to print Floyd’s Triangle.	CO3
22	Write an Objective-C program to print palindrome of a number.	CO3
23	Write an Objective-C program to print pyramid.	CO3
24	Write an Objective-C program to find greatest number in between three numbers	CO3
25	Write an Objective-C program to check whether a number is even or odd.	CO3
	Mini Project 1: Make an interactive project based on iOS App using Objective-C Language	CO4
	Mini Project 2: Upload your iOS App in Apple AppStore and Publish it	CO4

Department:	**Department of Computer Science and Engineering**		
Course Name: **DevOps & Automation**	**Course Code**	**L-T-P**	**Credits**
	ENSP411	4-0-0	4
Type of Course:	Minor (Department Elective III)		
Pre-requisite(s), if any: Nil			

COs	Statements
CO1	Understand the principles and benefits of DevOps, and its role in enhancing collaboration and efficiency between development and operations teams.
CO2	Acquire 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.
CO3	Demonstrate proficiency in containerization and orchestration techniques using Docker and Kubernetes for efficient and scalable application deployment and management.
CO4	Implement configuration management and Infrastructure as Code (IaC) using Ansible and Terraform to automate the provisioning and management of infrastructure resources.
CO5	Develop skills in monitoring, logging, and security practices in the context of DevOps, ensuring application performance, resilience, and adherence to security best practices.

Brief Syllabus:

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.

UNIT WISE DETAILS

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

Content Summary:

Overview of DevOps: Definition, objectives, and benefits.

DevOps Principles: Collaboration, automation, continuous integration, continuous delivery, and continuous deployment.

DevOps Tools: Introduction to popular DevOps tools like Git, Jenkins, Docker, Kubernetes, and Ansible.

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.

Continuous Delivery and Deployment: Implementing CD pipelines for deploying applications to various environments.

Unit Number: 2 **Title: Containerization and Orchestration** **No. of hours: 10**

Content Summary:

Introduction to Containers: Docker and containerization concepts, Container Management: Working with Docker containers, images, and registries, Docker Compose: Managing multi-container applications.

Introduction to Kubernetes: Container orchestration and Kubernetes architecture, Deploying Applications with Kubernetes: Deploying, scaling, and managing applications on Kubernetes.

Unit Number: 3 **Title: Configuration Management and Infrastructure as Code (IaC)** **No. of hours: 10**

Content Summary:

Introduction to Configuration Management: Need for configuration management tools.

Managing Infrastructure with Ansible: Ansible architecture and playbooks for automated configuration management.

Infrastructure as Code (IaC) Concepts: Managing infrastructure using code, benefits of IaC.

IaC with Terraform: Infrastructure provisioning using Terraform and cloud service providers (e.g., AWS, Azure).

Unit Number: 4 **Title: Monitoring, Logging, and Security in DevOps** **No. of hours:10**

Content Summary:
Application Monitoring: Monitoring tools and techniques for tracking application performance and health.
Log Management: Centralized log collection, analysis, and visualization.
Security in DevOps: Implementing security best practices in CI/CD pipelines and containerized environments.
DevOps Culture and Collaboration: Encouraging collaboration between development and operations teams.

***SELF-LEARNING COMPONENTS:**
https://elearn.nptel.ac.in/shop/iit-workshops/completed/cicd-devops-automation-and-devsecops-automation/

Please Note:
1)Students are supposed to learn the components on self-basis
2) At least 5-10 % syllabus will be asked in end term exams from self-learning components.

Reference Books:

1. Jez Humble and David Farley, "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation," Pearson Education, Inc., 2011.
2. Nigel Poulton, "The Kubernetes Book," Independently published, 2018.
3. Sam Newman, "Building Microservices: Designing Fine-Grained Systems," O'Reilly Media, Inc., 2015.
4. Eberhard Wolff, "Microservices Patterns: With examples in Java," Manning Publications, 2018.
5. Yevgeniy Brikman, "Terraform: Up & Running: Writing Infrastructure as Code," O'Reilly Media, Inc., 2017.

Program and course outcome mapping

Course Code and Title	**PO**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO 10**	**PO1 1**	**PO1 2**	**PSO1**	**PSO2**	**PSO3**	**PSO 4**
ENSP411/ DevOps & Automation	**CO1**	3	3	2	2	-	3	-	3	-	-	-	3	3	-	1	-
	CO2	3	3	-	3	3	-	-	3	-	-	-	3	2	1	-	1
	CO3	3	3	2	2	3	2	2	3	-	-	-	3	3	-	-	-
	CO4	-	3	2	3	3	-	2	3	2	-	-	3	1	1	1	1
	CO5	-	3	2	3	-	-	3	3	-	-	-	3	-	2	-	-

Department:	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
DevOps & Automation Lab	**ENSP461**	0-0-2	1
Type of Course:	Minor (Department Elective III)		
Pre-requisite(s), if any:			

Proposed Lab Experiments

Defined Course Outcomes

COs	**Course Outcomes (COs)**
CO 1	**Gain** hands-on experience in setting up version control using Git and performing collaborative software development with branching and merging techniques.
CO 2	**Acquire** practical knowledge in implementing continuous integration and continuous deployment (CI/CD) pipelines using Jenkins, automating the build, test, and deployment processes.
CO 3	**Develop** proficiency in containerization with Docker, including managing Docker containers and images, and deploying applications on Kubernetes for efficient and scalable orchestration.
CO 4	**Demonstrate** skills in infrastructure automation and configuration management using Ansible and Terraform to provision and manage cloud resources and application configurations.
CO 5	**Understand** and **apply** monitoring, logging, and security practices in DevOps, ensuring application performance, resilience, and adherence to security best practices throughout the software development lifecycle.

Ex. No.	**Experiment Title**	**Mapped CO(s)**
1	Setting up version control with Git	CO1
2	Implementing a basic Jenkins CI/CD pipeline	CO2
3	Automating application deployment with Jenkins	CO2
4	Containerizing an application using Docker	CO3
5	Managing Docker containers and images	CO3
6	Deploying applications with Kubernetes	CO3
7	Implementing Kubernetes deployment strategies	CO3
8	Continuous deployment with Kubernetes	CO3

9	Configuring infrastructure with Ansible	CO4
10	Automating application configuration with Ansible	CO4
11	Implementing Infrastructure as Code (IaC) with Terraform	CO4
12	Creating scalable and resilient infrastructure with Terraform	CO4
13	Monitoring application performance with Prometheus	CO5
14	Logging and centralized log management	CO5
15	Implementing security measures in CI/CD pipelines	CO5
16	Implementing feature flags for controlled feature rollout	CO5
17	Load testing and performance optimization	CO5
18	Automating application tests with Selenium	CO2, CO5
19	Integrating automated testing in CI/CD pipelines	CO2, CO5
20	Blue-green deployment for zero-downtime updates	CO3, CO5
21	Canary deployment for testing new features	CO3, CO5
22	Implementing GitOps for application deployments	CO3, CO5
23	Managing secrets and sensitive data securely	CO5
24	Disaster recovery planning and testing	CO5
25	Creating a DevOps project integrating multiple tools	CO1, CO2, CO3, CO4, CO5

1. **Setting up version control with Git**: Exercise: Initialize a Git repository, create branches, perform commits, and push changes to a remote repository. Project: Collaboratively work on a project using branching and merging techniques in Git.

2. **Implementing a basic Jenkins CI/CD pipeline**: Exercise: Set up a simple Jenkins pipeline to build and test a sample application from version control. Project: Develop a complete CI/CD pipeline that includes code building, automated testing, and deployment to a staging environment.

3. **Automating application deployment with Jenkins**: Exercise: Configure Jenkins to automatically deploy the application to a test server upon successful build. Project: Implement a full-fledged CD pipeline with Jenkins, including deployment to production after successful testing.

4. **Containerizing an application using Docker**: Exercise: Dockerize a basic application and run it in a container. Project: Containerize a multi-service application with Docker Compose for easier deployment.

5. **Managing Docker containers and images**: Exercise: Explore Docker commands to manage containers and images, such as starting, stopping, and cleaning up. Project: Implement a container registry and manage images for different application versions.

6. **Deploying applications with Kubernetes**: Exercise: Set up a Kubernetes cluster and deploy a basic application using YAML manifests. Project: Deploy a microservices-based application with Kubernetes, configuring services and network policies.

7. **Implementing Kubernetes deployment strategies**: Exercise: Implement rolling updates and rollbacks in Kubernetes. Project: Use Kubernetes deployment strategies like blue-green and canary deployments for a real-world application.

8. **Continuous deployment with Kubernetes**: Exercise: Set up a Jenkins pipeline for continuous deployment to Kubernetes. Project: Create an end-to-end automated CD pipeline with Jenkins and Kubernetes.

9. **Configuring infrastructure with Ansible**: Exercise: Use Ansible to provision and configure virtual machines. Project: Create a playbook to configure a complete development environment for an application.

10. **Automating application configuration with Ansible**: Exercise: Create Ansible playbooks to automate application-specific configurations. Project: Implement dynamic inventory and use Ansible roles for better code organization.

11. **Implementing Infrastructure as Code (IaC) with Terraform**: Exercise: Set up a basic Terraform configuration to create cloud resources. Project: Use Terraform to define infrastructure for a scalable and fault-tolerant application.

12. **Creating scalable and resilient infrastructure with Terraform**: Exercise: Implement auto-scaling and load balancing in Terraform. Project: Design a Terraform template for a highly available architecture using multiple availability zones.

13. **Monitoring application performance with Prometheus**: Exercise: Set up Prometheus for monitoring application metrics. Project: Create custom Prometheus metrics and use Grafana for visualization and alerting.

14. **Logging and centralized log management**: Exercise: Configure centralized log collection using tools like Fluentd or Logstash. Project: Set up ELK (Elasticsearch, Logstash, and Kibana) stack for efficient log analysis.

15. **Implementing security measures in CI/CD pipelines**: Exercise: Use Jenkins plugins to implement security checks in CI/CD pipelines. Project: Implement security scanning tools like SonarQube and integrate them into the pipeline.

16. **Implementing feature flags for controlled feature rollout**: Exercise: Add feature flags to a sample application to enable/disable specific features. Project: Implement a feature flag service for a real-world application and manage feature rollout.

17. **Load testing and performance optimization**: Exercise: Use load testing tools to evaluate application performance under heavy traffic. Project: Analyse performance bottlenecks and optimize the application for scalability.

18. **Automating application tests with Selenium**: Exercise: Use Selenium WebDriver for automating browser-based tests. Project: Develop an automated testing suite covering multiple application features.

19. **Integrating automated testing in CI/CD pipelines**: Exercise: Integrate automated tests into the Jenkins CI/CD pipeline. Project: Implement a complete testing strategy, including unit, integration, and end-to-end tests.

20. **Blue-green deployment for zero-downtime updates**: Exercise: Perform blue-green deployment for a sample application update. Project: Set up a blue-green deployment strategy for a production application.

21. **Canary deployment for testing new features**: Exercise: Implement canary deployment for a specific application feature. Project: Use canary deployment to gradually release new features to a subset of users.

22. **Implementing GitOps for application deployments**: Exercise: Use GitOps principles to manage Kubernetes manifests with Git. Project: Implement a GitOps workflow for application deployment and configuration management.

23. **Managing secrets and sensitive data securely**: Exercise: Utilize Kubernetes secrets or HashiCorp Vault to manage sensitive data. Project: Set up a secure secret management system for a production environment.

24. **Disaster recovery planning and testing**: Exercise: Design a disaster recovery plan for a sample application. Project: Test the disaster recovery plan and validate its effectiveness.

25. **Creating a DevOps project integrating multiple tools**: Exercise: Choose and integrate various DevOps tools into a sample project. Project: Create an end-to-end DevOps project showcasing the integration of tools and best practices.

Department:	Department of Computer Science and Engineering		
Course Name:	Course Code	L-T-P	Credits
.NET Framework	ENSP413	4-0-0	4
Type of Course:	Minor (Department Elective III)		
Pre-requisite(s), if any:			

COs	Statements
CO1	Understanding the fundamental concepts and components of the .NET Framework.
CO 2	Applying knowledge to design and develop applications using Windows Forms, WPF, and ASP.NET.
CO 3	Analysing performance considerations and troubleshooting errors in the .NET Framework.
CO 4	Integrating advanced topics like .NET Core, Entity Framework, and WCF for cross-platform development and service creation.
CO 5	Assessing security, reliability, scalability, and performance of applications developed using the .NET Framework.

Brief Syllabus:
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.

UNIT WISE DETAILS

Unit Number: 1 Title: Introduction to .NET Framework No. of hours: 8

Content Summary:
Overview of .NET Framework ,Introduction to the .NET platform, Evolution and history of .NET Framework, Key components and architecture of .NET Framework, Common Language Runtime (CLR) and Just-In-Time (JIT) compilation, Common Intermediate Language (CIL) and Intermediate Language (IL), Programming Languages in .NET (C# as the primary language for .NET development & Visual Basic .NET) ,Introduction to Visual Studio IDE, Installation and configuration of .NET Framework and Visual Studio, NuGet package manager and third-party libraries

Unit Number: 2 Title: .NET Framework Fundamentals No. of hours: 8

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 concepts and algorithms, Finalizers and the Dispose pattern, Performance considerations and best practices

Unit Number: 3 **Title: Building Applications with .NET Framework** **No. of hours: 12**

Windows Forms and WPF Applications, Introduction to Windows Forms and Windows Presentation Foundation (WPF), Designing user interfaces using WinForms/WPF controls, Event-driven programming and event handling, Data binding and data access in WinForms/WPF applications, ASP.NET Web Development, Data access and validation in ASP.NET applications, Web Services and RESTful APIs, Creating and consuming web services in .NET, Authentication and security considerations in web services.

Unit Number: 4 **Title: Advanced Topics in .NET Framework** **No. of hours: 12**

Content Summary:

.NET Core and Cross-Platform Development, Introduction to .NET Core and its advantages, Building cross-platform applications with .NET Core, Deploying and hosting .NET Core applications, Entity Framework and Database Connectivity, Overview of Entity Framework and Object-Relational Mapping (ORM), Creating and manipulating databases with Entity Framework, Querying data using LINQ (Language Integrated Query), Handling database migrations and versioning, Windows Communication Foundation (WCF), Introduction to WCF and service-oriented architecture (SOA), Creating and consuming WCF services, Message exchange patterns and bindings in WCF, Security and reliability in WCF applications

*Self-Learning Components:

1. Online Tutorials and Documentation: Direct students to the official Microsoft documentation for .NET Framework, which provides comprehensive guides and resources. Microsoft .NET Documentation
2. Hands-on Coding Exercises: Assign coding exercises from platforms like LeetCode or HackerRank that focus on implementing concepts of .NET Framework. LeetCode HackerRank
3. Project-Based Learning: Encourage students to work on small projects using different aspects of the .NET Framework. Provide examples of project ideas and resources like GitHub repositories for inspiration. GitHub

Please Note:

1)Students are supposed to learn the components on self-basis

2) At least 5-10 % syllabus will be asked in end term exams from self-learning components.

Reference/Text Books:

1. "Mastering C# and .NET Framework" by Jayantha Dhanapala
2. "Pro C# and .NET Framework" by Andrew Troelsen
3. ".NET Framework Programming with C#" by G. Shankar
4. ".NET Programming: Concepts and Practice" by Atul Kumar

Program and course outcome mapping

Course Code and Title	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
ENSP413/ .NET Framework	CO1	3	3	-	-	2	-	-	-	-	1	-	3	3	3	-	-
	CO2	1	3	3		1	-	-	-	-	2	-	3	1	3	1	
	CO3	1		3	2	1	-	-	-	-	1	-	3	2	1	-	1
	CO4	-	-	3	1	2	-	-	-	-	-	-	3	-	1	-	1
	CO5	-	2	3	-	2	-	-	-	-	1	-	3	1	2	-	-

Department:	**Department of Computer Science and Engineering**		
Course Name:	**Course Code**	**L-T-P**	**Credits**
.NET Framework Lab	**ENSP463**	0-0-2	1
Type of Course:	Minor (Department Elective III)		
Pre-requisite(s), if any: Nil			

Proposed Lab Experiments

Defined Course Outcomes

COs	Statements
CO 1	Gain a thorough understanding of the core concepts and components of the .NET Framework.
CO 2	Apply .NET Framework knowledge to design and develop applications, solving programming problems effectively.
CO 3	Analyze and troubleshoot .NET applications, using debugging techniques and optimizing performance.
CO 4	Integrate advanced .NET topics like .NET Core, Entity Framework, and WCF to create cross-platform applications, work with databases, and build services.

Ex. No	**Experiment Title**	**Mapped CO/COs**
1	Installing and setting up the .NET Framework, Visual Studio IDE, and NuGet package manager	CO1
2	Creating a basic console application in C# or Visual Basic.NET and running it in Visual Studio.	CO1
3	Write a program to display "Hello World" using C#.	CO2
4	Create a Windows Forms application to design a simple calculator.	CO2
5	Develop a console application to perform basic arithmetic operations	CO2
6	Create a class hierarchy to represent different types of vehicles.	CO2
7	Implement inheritance and polymorphism concepts in a C# program.	CO2
8	Design a Windows Forms application to manage student records.	CO3
9	Create a WPF application to build a simple photo gallery.	CO3
10	Develop a web application to display and manage a list of books using ASP.NET..	CO3

11	Implement form validation and data access in an ASP.NET application.	CO3
12	Build a RESTful API using ASP.NET Web API to perform CRUD operations on a database.	CO3
13	Create a client application to consume a web service and display the retrieved data.	CO2
14	Implement a cross-platform application using .NET Core.	CO3
15	Develop a database-driven application using Entity Framework for data manipulation.	CO3
16	Design and implement a WCF service to provide secure communication between client and server.	CO4
17	Connect a .NET application to a database using ADO.NET and retrieve data.	CO3
18	Use LINQ (Language Integrated Query) to perform data querying and manipulation operations.	CO3
19	Deploy a .NET application to a web server or a cloud platform.	CO4
20	Configure and manage the hosting environment for a .NET application.	CO4
21	Use debugging techniques and tools in Visual Studio to identify and fix bugs in a program.	CO2
22	Create a program to demonstrate the automatic memory management feature in .NET.	CO4
23	Implement a program to analyze and optimize memory usage in a .NET application.	CO2
24	Develop a WCF service to perform CRUD operations on a database.	CO4
25	Design a client application to consume the WCF service and display the retrieved data.	CO4

Department:	**Department of Computer Science and Engineering**		
Course Name: New-Age programming languages	**Course Code**	**L-T-P**	**Credits**
	ENSP415	4-0-0	4
Type of Course:	Minor (Department Elective III)		

COs	Statements
CO1	Understand the fundamental principles and paradigms of modern programming languages, including functional programming, object-oriented programming, and concurrent programming.
CO2	Develop proficiency in using the syntax, data structures, and control flow constructs of each language (GO, F#, Clojure, and Kotlin) to solve programming problems.
CO3	Explore 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	Apply the languages' respective development tools, such as Go's gofmt and go vet, F#'s F# Interactive (FSI), Clojure's Leiningen or Boot, and Kotlin's integrated development environment (IDE) support, to improve code quality and productivity.
CO5	Design and implement projects that integrate multiple programming languages, using appropriate inter-language communication mechanisms and libraries (e.g., Go and Kotlin interacting via REST APIs, F# and Clojure communicating via message queue

Brief Syllabus:
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.

UNIT WISE DETAILS

Unit Number: 1 **Title: GO programming Language** **No. of hours: 10**

Content Summary:
Overview of GO, F#, Clojure, and Kotlin, Comparison with traditional programming languages, Installation and setup of development environment, Introduction to GO syntax and data types, Control structures, Functions and packages in GO, Arrays, slices, and maps in GO, Structs and custom data types, Pointers and memory management, Concurrency and parallelism in GO, Error Handling, Concurrent Programming in GO, Advanced GO Concepts- Function closures and anonymous functions, Reflection and type introspection, Testing and benchmarking in GO, Writing concurrent and parallel programs.

Unit Number: 2 **Title: F# Programming Language** **No. of hours: 10**

Content Summary:
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:
Overview of Clojure and its features, Setting up the development environment, Basic syntax and data structures in Clojure, Functional Programming in Clojure, Immutable data and pure functions, Higher-order functions and recursion, Collections and sequence operations in Clojure, De-structuring and pattern matching, Macros and metaprogramming in Clojure, Concurrency models in Clojure, Asynchronous programming with core. async, Parallel programming with reducers and p-map, Interacting with Java libraries and APIs, Java interoperability in Clojure, Working with Java collections and objects, Web Development with Clojure, Building web applications using Clojure and Ring, Database access and persistence in Clojure, Error Handling and Testing: Exception handling and error management in Clojure, Testing strategies and frameworks in Clojure, 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.

***Self-Learning Components:**

3. Web programming with GO
4. F# for Data Science and Machine Learning:
5. Metaprogramming and DSLs in Clojure:
6. Android App Development with Kotlin:

References:

1. Building Modern Web Applications with Go (Golang) by Udemy
2. https://www.jetbrains.com/academy/
3. https://www.classcentral.com/subject/f-sharp
4. https://www.classcentral.com/subject/clojure

Please Note:
1)Students are supposed to learn the components on self-basis
2) At least 5-10 % syllabus will be asked in end term exams from self-learning components.

Reference Books:

6. The Go Programming Language, Alan A. A. Donovan and Brian W. Kernighan, Addison-Wesley Professional.
7. An Introduction to Programming in Go, Caleb Doxsey, CreateSpace Independent Publishing.
8. Real-World Functional Programming: With Examples in F# and C#, Tomas Petricek and Jon Skeet, Manning.
9. Programming F# 3.0: A Comprehensive Guide for Writing Simple Code to Solve Complex Problems, Chris Smith, O'Reilly Media.
10. Getting Clojure: Build Your Functional Skills One Idea at a Time, Russ Olsen, O′Reilly.
11. The Joy of Clojure, Michael Fogus and Chris Houser, Manning Publication.
12. Atomic Kotlin, Bruce Eckel and Svetlana Isakova, Mindview LLC.
13. Kotlin in Action, Dmitry Jemerov and Svetlana Isakova, Manning Publication.

Online References:

1. https://gobyexample.com/ [
2. https://golang.org/doc/
3. https://www.youtube.com/playlist?list=PLlxmoA0rQ-LwgK1JsnMsakYNACYGa1cjR
4. https://kotlinlang.org/docs/home.html
5. https://docs.microsoft.com/en-us/dotnet/fsharp/
6. https://www.udemy.com/course/learning-functional-programming-with-f/
7. https://clojure.org/guides/getting_started

Program and course outcome mapping

Course Code and Title	**PO**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
ENSP415/ New-Age programming languages	**CO1**	2	2	-	-	2	-	-	-	-	2	1	3	3	-	-	3
	CO2	2	2	-	-	2	-	-	-	-	2	2	3	3	2	-	-
	CO3	2	2	-	3	-	-	-	-	-		-	3	-	2	-	3
	CO4	-	-	-	-	3	-	-	-	-	3	1	2	-	2	-	3
	CO5	-	-	-	-		2	-	-	-	2	-	3	-	2	2	2

Department:	**Department of 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:	Minor (Department Elective III)		
Pre-requisite(s), if any: Nil			

Course Outcomes (CO)

COs	Statements
CO1	Understand the fundamental principles and paradigms of modern programming languages.
CO2	Develop proficiency in using the syntax, data structures, and control flow constructs of each language.
CO3	Explore 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	Apply the languages' respective development tools and best practices.
CO5	Design and implement projects that utilize the strengths of each language to tackle complex problems or tasks.

Proposed Lab Experiments

Ex. No	**Experiment Title**	**Mapped CO/COs**
	Practical on GO Programming Language	
1	Write a program that takes user input and performs basic calculations (e.g., addition, subtraction, multiplication) using different data types like integers and floats. Use control structures like if statements and loops to handle different scenarios and validate user input.	CO2
2	Create a package that contains multiple functions to perform common tasks, such as string manipulation or mathematical operations. Use these functions in a separate program to demonstrate their functionality and reusability.	CO1
3	Implement a program that stores a collection of elements using arrays. Perform operations like adding, removing, or updating elements	CO2

4	Define a struct Person with the following members: name, age, job and salary. Create methods associated with the struct to read data in structure and print data.	CO4
5	Develop a program that utilizes pointers to modify and manipulate data in memory. Explore concepts like referencing, dereferencing, and memory allocation/deallocation.	CO2
6	Write a program that demonstrates the use of Go routines and channels to achieve concurrent execution of tasks.	CO3
7	Create a program that handles various error scenarios and provides appropriate error messages or responses. Write unit tests for critical functions and verify their correctness using Go's testing package.	CO5
8	**Mini Project**: Task Manager Application in Go Create a task manager application using the Go programming language. The application should allow users to manage their tasks by adding, updating, and deleting tasks. The tasks should have attributes such as title, description, due date, and status (e.g., "in progress", "completed").	CO5
	Practicals on F# Programming Language	
9	AP to read marks of 4subjects and calculate the Percentage of student and find the result according to given conditions a. 60>=1st Division 60<&& 50>= 2nd Division 50<&& 40>=3rd Division 40<=fail. b. AP to accept an integer and check whether it is prime or not.	CO2
10	a. Write a function that takes a string as input and returns the reverse of the string. Also check if a given string is a palindrome b. Create a function that takes a string as input and performs the following transformations: i. If the string contains only alphabetic characters, convert it to uppercase. ii. If the string contains only numeric characters, convert it to an integer and double its value. iii. If the string contains a mix of alphabetic and numeric characters, return it as is. c. Design a function that validates an email address based on specific rules, such as the presence of an '@' symbol and a valid domain name. Use pattern matching to check if the input string matches the expected email format.	CO2
12	Implement a program that performs various operations on lists using higher-order functions (define a list of integers or strings). Write pure functions that demonstrate the map, filter, reduce/fold operations.	CO1
13	Implement a program that performs multiple I/O-bound or computationally intensive tasks concurrently using F#'s asynchronous workflows and parallel programming constructs.	CO3

14	Create a program that demonstrates the object-oriented programming (OOP) capabilities of F#. Define classes, objects, and inheritance hierarchies using F#'s OOP syntax.	CO3
15	Create a program that demonstrates the following tasks: i. Establish a connection to both the relational and NoSQL databases using appropriate database drivers or libraries. ii. Perform basic CRUD operations (Create, Read, Update, Delete) on the databases.	CO4
16	**Mini Project**: Employee Management System Create an Employee Management System using the F# programming language and a relational database. The system should allow users to perform CRUD (Create, Read, Update, Delete) operations on employee records stored in the database. It should provide functionality to add new employees, retrieve employee information, update employee details, and delete employee records.	CO5
	Practicals on Clojure Programming Language	
17	Write a program that demonstrates the basic syntax and data structures in Clojure, such as lists, vectors, maps, and sets.	CO1
18	Write functions that manipulate and transform sequences using operations such as map, filter, reduce, and take.	CO2
19	Implement a program that showcases asynchronous programming using the core.async library.	CO3
20	Write code that calls Java methods, creates Java objects, and works with Java collections and objects from Clojure.	CO4
21	Develop a web application using Clojure and the Ring library. Set up routes, handle HTTP requests and responses, and render dynamic content.	CO5
22	Write functions that interact with the database, perform CRUD operations, and handle transactions.	CO5
23	Implement error handling mechanisms, such as exception handling and error management, in Clojure.	CO4
24	**Mini Project**: Blogging Platform with Clojure Create a Blogging Platform using the Clojure programming language. The platform should allow users to create and publish blog posts, manage user accounts, and provide functionality for reading and commenting on blog posts. It should utilize a relational database for data storage and retrieval.	CO5
	Practicals on Kotlin Programming Language	
25	19 WAP for print following o/p Hello Kotlin!!!	CO2
	20 WAP to take employee's basic salary, dept_code and experience. Calculate bonus according to following criteria i. dept_code = 101 && exp <= 2 bonus = 3% ii. dept_code = 102 && exp <= 4 bonus = 5% iii. dept_code = 103 && exp <= 7 bonus = 8%	

21	WAP to accept an integer and display average of digit.	
26	Write a program in Kotlin that demonstrates various aspects of function declarations, parameters, and higher-order functions. a. Implement a function that takes two integer parameters and returns their sum. b. Create a function that has default parameter values for an optional third parameter, which is a string representing a greeting. If no greeting is provided, the function should use a default greeting. c. Explore named parameters by creating a function that takes multiple parameters and demonstrate how to call the function by specifying the parameter names explicitly. d. Implement a variable-length argument function that takes a variable number of integers and calculates their average. e. Utilize a higher-order function by creating a function that accepts a lambda expression as a parameter. The lambda should take an integer parameter and return the square of that integer.	CO2
27	WAP to create a class Student with data members' rollno, student name, course and percentage and member functions to accept and display the details of student. a. Implement properties, methods, and constructors in classes. b. Explore access modifiers and visibility scopes in Kotlin.	CO1
28	Implement a program that demonstrates the declaration and usage of nullable and non-nullable variables. Utilize safe calls (?.) and the Elvis operator (?:) to handle nullable values and provide alternative values or perform fallback actions.	CO3
29	WAP to implement various collections like lists, sets, and maps in Kotlin and perform common operations on them. Use collection functions and transformations such as map, filter, and reduce to manipulate data.	CO2
30	Implement a DSL for a domain-specific problem, showcasing Kotlin's expressive syntax and extension functions.	CO5
31	Implement a program that demonstrates the creation and usage of extension functions in Kotlin (Choose a specific class or data type, such as String). For example, you can create an extension function that counts the number of vowels in a string or reverses the string.	CO3
32	**Mini Project**: Quiz App Build a quiz application that presents users with multiple-choice questions on various topics. Users can select their answers, and the app provides instant feedback on correctness. Keep track of the user's score and display the result at the end of the quiz. Include features like a timer, score calculation, and a database of questions.	CO5

ANY OTHER INSTRUCTIONS:

NOTICES: All notices for the course will be displayed on A-Block, 2nd Floor Notice Board.

GLOSSARY AND NOTES

Programme Outcomes: POs are statements that describe what the students graduating from any of the educational Programmes of the institution should be able to do on completion.

Programme Specific Outcomes: PSOs are statements that describe what the graduates of a specific educational Programme should be able to do on completion.

Course Outcomes: COs are statements that describe what students should be able to do on completion of the course.

Program Articulation Matrix: Program articulation matrix gives the correlation among CO & PO and CO & PSO. The strength of correlation is interpreted in three levels: weakly mapped (1), moderately mapped (2), strongly mapped (3).

***Teaching –Learning Methods**: Teaching –Learning Methods may include Lecture/Group Discussion/Presentation/Case-study/Demonstration using simulation or a tool/ Interview/ Quiz/Debate/Project/Field Project/Experiment etc.

****Mode of Evaluation**: Mode of Evaluation may include Assignment/Quiz/Test/Interview/Peer Review/Report/Presentation/Open Book Test/Evaluated Discussion Forum etc.