SCHOOL OF ENGINEERING AND TECHNOLOGY

MCA (Master's in Computer Application)

Programme Code: 56

POSTGRADUATE COURSE

Scheme of Study and Syllabi

(with effect from 2025-26)

Session: 2025-27

Approved in the 38th Meeting of AcademicCouncil Held on

28 June 2025

Table of Contents

Preamble

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

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

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

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

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

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

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

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

School of Engineering & Technology

K.R Mangalam University

Categories of Courses

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

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

Natural and Physical Sciences

Mathematics, Statistics, and Computer Applications

Library, Information, and Media Sciences

Commerce and Management

Humanities and Social Sciences

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

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

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

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

Industry project/Research Project: Students choosing a 2-Year PG degree(MCA) are required to take up research projects under the guidance of a faculty member. The students are expected to complete the Research Project in the final semester. The research outcomes of their project work may be published in peer-reviewed journals or may be presented in conferences /seminars or may be patented.

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

University Vision & Mission

Vision

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

Mission

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

About School

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

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

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

The School of Engineering & Technology is committed to providing a cutting-edge curriculum by integrating the best practices from top global universitiesand leveraging the rich knowledge resources of the Open-Source Society University. The curriculum focuses on problem-solving, design, development, interdisciplinary learning, skill development, research opportunities and application of various emerging technologies with focus on innovative teaching learning methodologies.

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

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

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

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

We are dedicated to fostering a culture of innovation and entrepreneurship, recognizing these as essential pillars for the success of our students in the rapidly evolving world of technology. We inspire innovation and entrepreneurship through our dynamic Entrepreneurship and Incubation

Center, engaging contests like 'MindBenders' ,'Hack-KRMU,' participation in ‘Smart India Hackathon’, International Conference ‘MRIE’ empowering students to become forward-thinking leaders in the ever-evolving realm of technology.

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

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

School Vision & Mission

Vision

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

Mission

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

About the Program

The Master of Computer Applications (MCA) program is a comprehensive postgraduate degree designed to provide students with advanced knowledge and skills in computer science and applications. The program typically spans two to three years and is structured to encompass a blend of theoretical and practical learning. It covers a wide range of subjects, including programming languages, database management, software engineering, computer networks, artificial intelligence, and data analytics. The curriculum is meticulously crafted to keep pace with the ever-evolving technology landscape, ensuring that graduates are well-equipped to tackle contemporary challenges in the IT industry.

In addition to core computer science topics, the MCA program often includes courses on business management, mathematics, and communication skills, which are crucial for holistic professional development. Students engage in hands-on projects, internships, and practical labs that foster experiential learning and problem-solving abilities. These activities are designed to simulate real-world scenarios, preparing students for the dynamic and demanding environment of the tech industry. One of the distinguishing features of the MCA program is its emphasis on emerging technologies such as cloud computing, Internet of Things (IoT), cybersecurity, and machine learning. This forward-looking approach ensures that graduates are not only proficient in current technologies but are also capable of adapting to future advancements. Moreover, the program encourages research and innovation, often culminating in a significant capstone project or thesis, which allows students to explore and contribute to cutting-edge developments in their chosen areas of interest. Overall, the MCA program aims to produce highly skilled and adaptable professionals ready to excel in various roles within the IT sector.

Programme Outcomes (POs)

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

Programme Specific Outcomes (PSOs)

Programme Specific Outcomes define what the students should be able to do at the time of graduation and they are Programme specific. There are two to four PSOs for a Programme.

Programme Educational Objectives (PEOs)

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

Credit

Credit refers to a unit that measures the amount of academic work required for a course. It typically reflects the number of instructional hours per week.

Program Educational Objectives (PEO)

PEO1: Achieve success in industry, government, academia, research, entrepreneurship, or consulting roles related to computer applications.

PEO2: Apply advanced computer application principles and methodologies to solve real-world problems, integrating theoretical knowledge with practical skills.

PEO3: Uphold professional ethics and social responsibilities, contributing to societal well-being and sustainability through the effective use oftechnology.

PEO4: Exhibit leadership and teamwork skills to effectively manage and collaborate in diverse, multidisciplinary projects within the field of computer applications.

Program Outcome (PO)

PO1. Core Competencies in Computer Applications: Demonstrate a solid foundation in computer application principles, critical problem analysis, and solution design, with the ability to conduct thorough investigations and address complex technical challenges.

PO2. Modern Tool Usage: Create, select, and apply advanced techniques, resources, and IT tools for complex computer application tasks, with an understanding of their limitations and applications.

PO3. Societal and Environmental Responsibility: Evaluate and address societal, health, safety, legal, and cultural issues related to computing, considering the environmental impact of solutions and advocating for sustainable practices.

PO4. Ethics: Apply ethical principles and adhere to professional ethicsand norms in the practice of computer applications.

PO5. Effective Communication and Team Collaboration: Excel in both individual and team roles in diverse and multidisciplinary environments, effectively communicating complex computer application concepts through reports, presentations, and interactions.

PO6. Project Management: Utilize management principles to lead and manage projects effectively within computer applications contexts.

PO7. Life-Long Learning: Engage in continuous learning to stay updated with technological advancements and evolving practices in computer applications.

Program Specific Outcomes (PSOs)

PSO1: Understanding advanced concepts, theories, tools, techniques, and methodologies in computer applications.

PSO2: Applying advanced knowledge and techniques to address and solve real-world challenges in computer applications.

PSO3: Analysing and evaluating methodologies, problems, and issues within various contexts of computer applications.

PSO4: Assessing alternative solutions and making informed decisions to address complex problems in computer applications.

PSO5: Designing and developing innovative solutions to tackle sophisticated challenges in computer applications.

Career Avenues

A Master's in Computer Applications (MCA) opens up a wide array of career opportunities across the ever-evolving domains of software development, system administration, data management, and IT consulting. Graduates from the core MCA program are well-prepared to take on roles that involve designing software solutions, managing databases and networks, and implementing IT services across industries. The key career avenues available to MCA graduates are as follows:

1. **Software Developer / Application Programmer** Designs, develops, and maintains software applications for desktop, web, and mobile environments.
2. **Full Stack Developer** Handles both client-side and server-side development using technologies such as HTML, CSS, JavaScript, Node.js, and databases.
3. **System Analyst** Analyzes system requirements, evaluates technologies, and recommends improvements for organizational IT systems.
4. **Database Administrator (DBA)** Manages and secures organizational databases, ensuring availability, integrity, and performance.
5. **Data Engineer** Designs data pipelines and architectures for collecting, storing, and processing large volumes of data.
6. **Web Developer** Creates dynamic websites and web applications using technologies such as PHP, React, Angular, and Django.
7. **Mobile Application Developer** Develops mobile apps for Android and iOS platforms using tools such as Kotlin, Swift, or Flutter.
8. **Network Administrator** Manages and troubleshoots network infrastructure, ensuring reliable connectivity and system performance.

9. **Cybersecurity Analyst** Monitors and defends systems against cyber threats, ensuring data privacy and regulatory compliance.
10. **IT Support Engineer** Provides technical support and troubleshooting for hardware, software, and IT services.
11. **Cloud Solutions Architect** Designs cloud-based solutions and manages deployments on platforms such as AWS, Azure, or Google Cloud.
12. **DevOps Engineer** Automates software development and deployment workflows, integrating development and operations processes.
13. **Software Tester / QA Analyst** Conducts testing and quality assurance activities to ensure bug-free and efficient software performance.
14. **Data Analyst / BI Analyst** Uses data visualization and analytics tools to extract insights and support business decision-making.
15. **Project Manager / Technical Lead** Oversees the planning, execution, and delivery of software projects, ensuring alignment with business goals.
16. **IT Consultant** Advises clients on strategic use of technology, systems integration, and IT process improvement.
17. **Entrepreneur / Tech Startup Founder** Launches and manages technology-driven ventures focusing on innovative digital solutions or services.
18. **Academician / Lecturer** Engages in teaching and academic research in computer science, information technology, or related fields.
19. **Research Scholar / Ph.D. Candidate** Pursues doctoral research in computing areas such as algorithms, distributed systems, or software engineering.

Duration

2Years (4 Semesters) - Full-Time Program

Eligibility Criteria

The candidate must have a bachelor's degree in a relevant field such as Computer Science, Information Technology, or Computer Applications from a recognized university or institution, with a minimum aggregate of 50% marks. Additionally, the candidate must have studied Mathematics as a subject in 12th grade and secured at least 50% marks in it. The reservation and relaxation for SC/ST/OBC/PWD and other categories shall be as per the applicable rules of the central or state government.

Student's Structured Learning Experience in the Programme

University Education Objective

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

Importance of Structured Learning Experiences:

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

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

Inside the Classroom:

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

Outside the Classroom:

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

Educational Planning and Execution

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

Entry Phase

Upon entering the MCA program, students are introduced to the foundational concepts of Artificial Intelligence, Machine Learning and programming. This phase is designed to strengthen their understanding of core scientific and technical principles. Courses such as AI & ML, Problem solving and advanced programming concept, Advanced DBMS and Data Structure and Algorithm provide a strong foundation. Students also engage in hands-on laboratory sessions to complement theoretical learning, which helps them connect classroom knowledge with real-world applications.

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

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

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

Core Learning

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

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

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

Skill Development

Throughout the program, there is a significant emphasis on developing practical skills and ensuring students are industry-ready. Courses on Artificial Intelligence, Machine Learning, Cloud Computing, and Cybersecurity provide students with cutting-edge knowledge in emerging fields.

Capstone and Exit Phase

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

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

Co-Curricular and Extra-Curricular Activities

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

Ethics and Professional Values

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

Career Counselling and Entrepreneurship

The university offers comprehensive **career counselling services**, providing students with expert guidance on **job placements, internships**, and **skill development** to help them effectively navigate their career paths. In addition, the university's **incubation center** plays a pivotal role in nurturing

entrepreneurial and leadership skills, empowering students to explore innovative ideas and launch their own ventures. These initiatives are designed to equip students with the tools and resources necessary for professional success and entrepreneurial growth.

Course Registration

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

Student Support Services

Mentor-Mentee: At K.R. Mangalam University, the **Mentor-Mentee Program** plays a crucial role in fostering academic and personal growth. Each student is assigned a faculty mentor who serves as a guide throughout their academic journey. This program ensures continuous interaction, where mentors assist students with academic planning, help in resolving personal issues, and provide career guidance. The mentor-mentee relationship transcends the classroom and often involves personal development, professional growth, and overall well-being. The program aims to nurture a supportive environment that enhances the learning experience and helps students reach their full potential.

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

Evaluation of Learning:

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

I. Evaluation Scheme (Theory Courses):

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

II. Evaluation Scheme (Laboratory/Practical Courses):

Evaluation Components	Weightage
Internal Assessments–	
Conduct of Experiment	10 Marks
Lab Records	10 Marks
Lab Participation	10 Marks
Lab Project	20 Marks
External Assessments- End term Practical Exam and Viva Voce	50 Marks
Total	**100 Marks**

Feedback and Continuous Improvement Mechanisms:

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

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

Academic Integrity and Ethics:

K.R. Mangalam University upholds the highest standards of academic integrity and ethics as a core value of its educational philosophy. The university implements a zero-tolerance policy towards academic misconduct, including plagiarism and other unethical practices. To ensure transparency and honesty in academic work, plagiarism detection software like Drillbit is used to maintain the originality of student submissions and research outputs. Students and faculty are regularly sensitized on the importance of ethical behavior through workshops, seminars, and classroom discussions. The university also integrates ethics and professional values into its curriculum across various disciplines, ensuring that graduates not only excel academically but also demonstrate integrity and responsibility in their professional and personal lives.

Scheme of Studies

Program Name	Master of Computer Application
Total Credits	88
Total Semesters	4

Credit Distribution Summary

Program Name	I	II	III	IV	Total Credits
MCA	26	26	22	14	88

SEMESTER I

SN	Course_Code	Category	Course Title	L	T	P	Credit
1	ETCCPP171	Major	Problem Solving and Advanced Programming Concepts	3	0	2	4
2	ETCCDS172	Major	Data Structures and Algorithms	3	0	2	4
3	ETCCAD173	Major	Advanced Database Management Systems	3	0	2	4
4	ETMCEH174	Major	Information Security and Ethical Hacking	3	0	2	4
5	ETCCWD175	Major	Full Stack Web Development	3	0	2	4
6	SEC-1	SEC	Competitive Coding - I	2	0	0	2
7	SEC-2	SEC	Data Analysis with Power BI & KNIME	2	0	0	2
8	AEC	AEC	Verbal Ability	**2**	**0**	**0**	**2**
			TOTAL	**21**	**0**	**10**	**26**

SEMESTER II

SN	Course_Code	Category	Course Title	L	T	P	Credit
1	ETCCJP271	Major	Comprehensive Java Programming: From Basics to Advanced	3	0	2	4
2	ETCCAP272	Major	Advanced Software Engineering & Agile Practices	3	0	2	4
3	ETMCCC273	Major	Cloud Computing	3	0	2	4
4	ETMCGI274	Major	Generative AI	3	0	2	4
5	ETCCSD275	Major	Applied System Design	4	0	0	4
6	ETCCPR276	Proj	Minor Project	0	0	4	2
7	SEC-3	SEC	Competitive Coding - II	2	0	0	2
8	AEC	AEC	Communication & Personality Development	2	0	0	2
TOTAL				**20**	**0**	**12**	**26**

SEMESTER III

SN	Course_Code	Category	Course Title	L	T	P	Credit
1	**DSE**	**DSE**	**Discipline Specific Elective-I**	**4**	**0**	**0**	**4**
2	**DSE**	**DSE**	**Discipline Specific Elective-II**	**4**	**0**	**0**	**4**
3	AEC	AEC	Arithmetic and Reasoning Skills	2	0	0	2
4	AEC	AEC	Comprehensive Placement Preparation	2	0	0	2
5	ETCCIN301	INT	Summer Internship	0	0	4	2
6	MOOC	MOOC	MOOC(Swayam/ NPTEL/AICTE's ELIS)	2	0	0	2
7	ETCCPR302	Proj	Major Project -I	0	0	8	4
8	SEC	SEC	Competitive Coding – III	2	0	0	2
			TOTAL	16	0	12	22

	Discipline Specific Elective-I	L	T	P	Credits
DSE	Principles and Practices of System Design	4	0	0	4
DSE	Image Processing & Computer Vision	4	0	0	4
DSE	Game Development using Unity and C#	4	0	0	4
DSE	Natural Language Processing	4	0	0	4
DSE	Blockchain Technologies	4	0	0	4
DSE	Swarm Intelligence and Nature-Inspired Optimization	4	0	0	4

	Discipline Specific Elective-II	L	T	P	Credits
DSE	Applied Research Methodologies in Computer Science	4	0	0	4
DSE	IoT Application Development with Arduino and Raspberry Pi	4	0	0	4
DSE	Advanced Computer Networking	4	0	0	4
DSE	Quantum Computing	4	0	0	4
DSE	Augmented Reality (AR) and Virtual Reality (VR)	4	0	0	4
DSE	Real-Time Stream Data Analytics	4	0	0	4
DSE	Applied Computer Graphics and Animation	4	0	0	4

SEMESTER IV

SN	Course_Code	Category	Course Title	L	T	P	Credit
1	ETCCIN471	INT	Industry/Research Internship	0	0	16	8
2	ETCCPR472	PROJ	Major Project-II	0	0	8	4
3	MOOC	MOOC	MOOC(Swayam/ NPTEL/AICTE's ELIS)	2	0	0	2
			TOTAL	2	0	24	14

Total Credits: 88

Syllabus

Semester: 1

Problem Solving and AdvancedProgramming Concepts

Course Name:	**Course Code**	**L-T-P**	**Credits**
Problem Solving and Advanced Programming Concepts	ETCCPP171	3-0-2	4
Type of Course:	Major		
Pre-requisite(s): Intermediate-level programming in Python, with a solid understanding of data structures (lists, dictionaries, sets), control structures, functions, recursion, and basic algorithmic problem-solving.			

Course Perspective:

This course enables students to build functional, data-driven applications using Python. Learners start by mastering core Python syntax, control structures, functions, and data types. They then explore modular programming, object-oriented design, and exception handling. Midway, students begin developing scripts that interact with files, APIs, and databases. The final phase integrates libraries like Flask for web backends and Pandas/Matplotlib for data-driven visualization. Students build and deploy real applications, document their code, and collaborate using GitHub. By course end, learners are equipped to design backend systems or data-powered applications, forming the foundation for careers in backend or full-stack development. The Course Outcomes (COs).

On completion of the course the participants will be:

COs	Statements
CO 1	Writing efficient Python programs using control structures, functions, and built-in data types.
CO 2	Developing modular, reusable code with classes, files, and exception handling.
CO 3	Performing data analysis using libraries like Pandas and visualize results using Matplotlib.
CO 4	Building and testing basic web applications using Flask and integrate them with APIs or databases.
CO5	Applying software development practices including debugging, version control, and code documentation.

Course Outline:

Unit Number: 1	**Title: Python Programming Foundations**	**No. of hours: 12**
Content: • Python interpreter, script vs. interactive mode, and tooling setup (VS Code, pip, Git) • Variables, numeric and string types, formatted printing (f-strings) • Operators: arithmetic, comparison, logical, identity, membership • Control flow: if, elif, else, nested decisions • Loops: for, while, break/continue/pass • Functions: arguments, return values, default/keyword args • Working with strings: slicing, built-ins (split, join, replace, etc.) • **Hands-on focus:** weekly mini-scripts (e.g., calculator, text summarizer, login verifier)		

Unit Number: 2	**Title: Data Structures, File I/O & Exception Handling**	**No. of hours: 11**
Content: • Built-in data structures: lists, dictionaries, tuples, sets • List/dict comprehensions; nested structures • File handling: reading/writing text, CSV, JSON • Exception handling: try–except–else–finally, custom exceptions • Debugging with tracebacks and pdb, linting with flake8 • **Hands-on focus:** write programs to read from files, parse user input, and handle unexpected data		
Unit Number: 3	**Title: Object-Oriented Programming & Modular Design**	**No. of hours: 11**
Content: • OOP principles: encapsulation, abstraction, inheritance, polymorphism • Classes and objects: __init__, attributes, methods, __str__, etc. • Instance vs. class variables, method overriding • Writing and importing modules; Python packages • Project structure: entry points, main blocks, virtual environments • **Hands-on focus:** design a class-based CLI app; modularize with files and test cases		
Unit Number: 4	**Title: Data Handling, APIs & Web Backends**	**No. of hours: 11**
Content: • Introduction to Pandas: Series and DataFrame, loading CSV		

- Basic analysis: filtering, grouping, aggregation
- Visualization with Matplotlib: line plots, bar charts, scatter, histograms
- Introduction to Flask: routing, templates, form processing
- Working with JSON APIs and handling responses
- Hands-on focus: build a micro Flask app using live or local data

Learning Experiences

Inside Classroom Learning:

- Interactive Coding Exercises: Students practice writing Python programs in real-time with in-class coding challenges related to loops, data types, and control structures.
- Live Code Reviews: Students submit code for peer review, receiving constructive feedback on maintaining clean code principles.
- Hands-on File Handling Tasks: Students work on file manipulation tasks, exploring real-world scenarios using CSV, JSON, and regular expressions.
- Mini-Projects: Small projects like building a basic text-based game to apply.
- Python basics and data structures.
- Web Scraping Practice: Implement web scraping in class to extract data from websites using Python libraries.
- Error-Handling Workshops: Collaboratively debug code with exception-handling techniques, promoting troubleshooting skills.
- Function Design Practice: Group exercises in designing clean, efficient functions using lambda, map, and filter functions.

Outside Classroom Learning:

- Online Coding Competitions: Participation in coding challenges on platforms like LeetCode or HackerRank to practice clean code.
- Group Web Scraping Projects: Students collaborate on web scraping projects using real-world data to extract insights from websites
- Database Connectivity Assignments: Real-time project work with MySQL databases, creating systems to fetch and insert data
- Open-Source Contributions: Students contribute to open-source Python projects, adhering to clean coding standards.
- Hackathons: Participation in hackathons focused on building Python-based solutions for cybersecurity challenges.
- Documentation Writing: Students practice writing clear, concise documentation for their Python projects, reinforcing clean coding habits.

Textbooks:

- Eric Matthes, Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 3rd Edition, No Starch Press, 2023. ISBN: 978-1-71850-264-2
- Luciano Ramalho, Fluent Python, O'Reilly Media, 2nd Edition, 2022. ISBN: 978-1492056355
- Official Python Docs: https://docs.python.org/3/

Reference Books:

- R. Nageswara Rao, "Core Python Programming", Dreamtech Wesley J. Chun. "Core Python Programming, Second Edition", Prentice Hall5.

Lab Experiments

S. No	Task
1	**Lab 1 – "Student Attendance Tracker"** • Input daily names and timestamps • Store entries in lists and dictionaries • Generate formatted attendance summaries • Validate entries using control statements
2	**Lab 2 – "File-Based Contact Book"** • Read/write contacts to CSV/JSON • Use exception handling for missing or corrupt files • Implement search, update, and delete features • Format console output neatly with f-strings
3	**Lab 3 – "OOP Library Inventory System"** • Define Book, Member classes with methods • Persist data to files; support borrowing/returning • Refactor into modules for better reusability • Include class-level analytics (e.g., most borrowed book)
4	**Lab 4 – "Flask Weather App with Charts"**

	• Fetch weather from a JSON API • Display results in tabular HTML and graphs using Matplotlib • Use route parameters and templates for user input • Deploy locally and document setup
5	**Capstone – "Python Portfolio App"** • Design a Flask-based multi-page personal dashboard • Integrate file uploads, data visualization, and user form inputs • Apply object-oriented patterns and modular design • Document using Markdown, deploy on GitHub, and present to class

DATA STRUCTURE AND ALGORITHMS

Course Name: Data Structure and Algorithms	Course Code	L-T-P	Credits
	ETCCDS172	3-0-2	4
Type of Course:	Major		
Pre-requisite(s), if any:	Strong foundation in programming (C/C++/Python), logic design, and discrete mathematics.		

Course Perspective:

This course is designed to build essential skills in implementing and analyzing data structures and algorithms for problem-solving and system design. Emphasis is placed on time-space trade-offs, memory management, abstract data types, and algorithmic paradigms including divide and conquer, greedy strategies, and dynamic programming. The course bridges theory and application by incorporating complexity analysis and real-world data handling challenges.

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

COs	Statements
CO1	Applying fundamental data structures such as arrays, stacks, queues, linked lists, trees, and graphs in solving computational problems.
CO2	Analyzing and implement sorting, searching, and traversal algorithms using appropriate complexity metrics.
CO3	Developing optimized solutions using algorithmic strategies including divide & conquer, greedy algorithms, and dynamic programming.

CO4	Designing and evaluating the performance of abstract data types and memory-efficient structures in real-world applications.

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

Course Outline:

Unit Number: 1	**Foundations of Data Structures and Algorithm Analysis**	**No. of hours: 12**
Topics Covered: • Abstract Data Types (ADT), complexity analysis (Big O, Omega, Theta) • Arrays, dynamic arrays, strings, and matrices • Recursion, tail-recursion and iteration • Stack and Queue: linear, circular, deque • **Real-World Use Case:** Expression parsing and evaluation using stack-based algorithms		
Unit Number: 2	**Linked Structures and Trees**	**No. of hours: 11**
Topics Covered: • Singly, doubly and circular linked lists • Stack/Queue using linked lists • Trees: binary trees, binary search trees, AVL trees • Tree traversals (inorder, preorder, postorder), height-balanced trees • **Real-World Use Case:** Hierarchical file system simulation using trees		
Unit Number: 3	**Hashing, Heaps and Graphs**	**No. of hours: 11**

Topics Covered:		
• Hash tables: collision resolution (chaining, open addressing) • Heaps and priority queues: max/min heap operations • Graphs: representation, DFS, BFS, shortest path (Dijkstra's, Bellman-Ford) • Spanning trees: Prim's and Kruskal's algorithm • **Real-World Use Case:** Implementing an emergency priority system and route planner		
Unit Number: 4	**Sorting, Searching and Algorithmic Strategies**	**No. of hours: 11**
Topics Covered: • Searching: linear, binary, interpolation • Sorting: selection, insertion, merge sort, quicksort, heap sort • Divide and conquer, greedy techniques, dynamic programming • Matrix chain multiplication, 0/1 Knapsack, LIS • **Real-World Use Case:** Scheduling optimization and dynamic data ranking for ecommerce platforms		

Text Books:

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). MIT Press.
- Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2013). *Data Structures and Algorithms in Python.* Wiley.
- Ellis Horowitz, Sartaj Sahni – Fundamentals of Data Structures

- Thomas H. Cormen et al. – Introduction to Algorithms (CLRS)
- Narasimha Karumanchi – Data Structures and Algorithms Made Easy
- Mark Allen Weiss – Data Structures and Algorithm Analysis in C++
- Robert Lafore – Data Structures in C++

Learning Experiences

Classroom Learning Experience

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

Conceptual Understanding: Cover topics like arrays, linked lists, stacks, queues, trees, and graphs.

Problem-Solving Sessions: Conduct in-class exercises focused on implementing and using various data structures.

Theory Assignments: Assign theoretical problems that reinforce data structure concepts, discussed in class.

Group Work: Collaborate on projects that require designing and optimizing data structures.

Case Studies: Analyze real-world applications of data structures in software development.

Continuous Feedback: Implement quizzes and peer reviews to assess understanding and coding practices.

Outside Classroom Learning Experience

Theory Assignments: Assign take-home projects that apply data structure concepts to practical problems.

Lab Projects: Facilitate hands-on programming tasks using data structures in real-world scenarios.

Question Bank: Provide practice problems and resources for self-assessment on data structures.

Online Forums: Create platforms for discussing data structure challenges and solutions.

Lab Experiments

S. No	Task
1	**LAB TASK 1:** Stack and Queue Simulation – Railway Management System Objective: To simulate train arrivals and platform allocation using stack and queue data structures. Real-World Scenario: A railway station system handles trains arriving and departing using FIFO and LIFO operations for scheduling. Activities: • Implement stack and queue using both arrays and linked lists. • Simulate train arrival and departure with real-time platform assignment. • Track turnaround time and occupancy using queue operations. • Handle queue overflow and underflow conditions gracefully. Learning Focus: Stack/queue implementation, operational flow, dynamic memory handling. Tools: C++/Python, CLI, VS Code
2	**LAB TASK 2:** Tree-Based File System Simulator Objective: To simulate a file system hierarchy using binary tree and BST concepts. Real-World Scenario: A simplified operating system file manager is designed using tree-based structure for efficient access. Activities: • Create binary search tree for directories and files with add/delete/search operations.

	• Simulate traversal operations to display hierarchy (inorder, preorder, postorder). • Visualize memory layout and calculate file structure depth. • Implement dynamic memory management for insertion and deletion. Learning Focus: Tree traversal, BST operations, recursion, hierarchical simulation. Tools: C++/Python, CLI tools
3	**LAB TASK 3:** Graph Navigation Engine – Campus Map Routing Objective: To build a routing engine based on graph algorithms for navigating a university campus. Real-World Scenario: Students use the system to find shortest routes between buildings using weighted and unweighted paths. Activities: • Represent campus map using adjacency matrix and list. • Implement BFS and DFS for route search. • Use Dijkstra's algorithm to compute shortest paths and travel cost. • Display multiple alternate paths and analyze graph density. Learning Focus: Graph traversal, pathfinding, Dijkstra, real-world modeling. Tools: Python (NetworkX), C++ STL, console I/O

4	**LAB TASK 4: Dynamic Programming & Greedy – Task Scheduler** • **Objective:** To solve resource allocation and optimization problems using dynamic programming and greedy techniques. **Real-World Scenario:** A tech company needs to assign minimal employees to projects while maximizing total project value. **Activities:** • Implement 0/1 Knapsack, Activity Selection, and Coin Change problems. • Use recursion and memoization to optimize performance. • Visualize subproblem overlap and computation reuse. • Compare greedy vs dynamic strategies using sample datasets. **Learning Focus:** Dynamic programming, greedy choice, recursion tracing. **Tools:** C++/Python, CLI or Jupyter Notebooks
5	**CAPSTONE PROJECT:** Inventory Management System with Real-Time Reordering Objective: To develop a real-time inventory system that tracks, reorders, and reports stock dynamically using data structures. Real-World Scenario: A warehouse system that monitors product levels and triggers automated reordering based on threshold levels. Activities: • Create BST to maintain inventory catalog (insert/delete/search). • Use queues for order processing and stacks for return management. • Track low-stock items using min-heap and trigger reordering. • Generate daily reports sorted by product category or stock level using merge/quick sort.

	Learning Focus: DS integration, sorting algorithms, real-world system design. Tools: C++/Python, CLI, SQLite

Advanced Database Management Systems

Course Name: Advanced Database Management Systems	**Course Code**	**L-T-P**	**Credits**
	ETCCAD173	3-0-2	4
Type of Course:	Major		
Pre-requisite(s), if any: Prior knowledge of relational databases, SQL, ER modeling, normalization, and basic transactions.			

Course Perspective. This course equips students with advanced database concepts required for modern data-driven applications. Emphasis is placed on distributed and NoSQL systems, transaction processing, query optimization, and database security. It also explores current technologies including time-series and graph databases, preparing students for research, development, and high-performance database design.

The Course Outcomes (COs): On completion of the course theparticipants will be:

COs	**Statements**
CO 1	Designing and implement distributed and parallel database architectures for scalable and reliable systems.
CO 2	Applying advanced transaction processing, concurrency control, and recovery techniques in distributed environments.
CO 3	Evaluating modern database models including NoSQL, time-series, and graph databases for diverse use cases.
CO4	Assessing and implementing query optimization, access control, and data security mechanisms in large-scale databases.

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

Course Outline:

Unit Number: 1	**Distributed and Parallel Databases**	**No. of hours: 12**
Topics Covered: • Distributed DBMS architecture: Homogeneous vs. Heterogeneous • Data fragmentation, replication, and allocation • Distributed query processing and optimization • Parallel query execution and intra-query parallelism • CAP Theorem, consistency models (strong, eventual, causal) • **Real-World Use Case:** Scalable multi-location inventory management for a global supply chain		
Unit Number: 2	**Advanced Transaction and Concurrency Control**	**No. of hours: 11**
Topics Covered: • Advanced concurrency control: Two-phase locking, timestamp ordering, MVCC • Deadlock detection, avoidance, prevention • Advanced transaction models: Nested, Long-duration, Compensating transactions • Distributed transactions: Two-phase commit, three-phase commit • ARIES recovery algorithm, logging, checkpoints • **Real-World Use Case:** Ensuring atomic e-commerce order placement across services in a microservices environment.		
Unit Number: 3	**Emerging Database Models**	**No. of hours: 11**

Topics Covered: • NoSQL databases: Document (MongoDB), Column (Cassandra), Key-Value (Redis), Graph (Neo4j) • Time-series databases (InfluxDB, TimescaleDB): Architecture and applications • Spatial and Temporal databases • Semantic data models: RDF, OWL, SPARQL basics • Data lakes and federated systems • **Real-World Use Case:** Time-series data management in industrial IoT or financial markets		
Unit Number: 4	**Query Optimization and Database Security**	**No. of hours:1 1**
Topics Covered: • Cost-based query optimization, join reordering, bushy trees • Materialized views and query rewriting • Access control models: DAC, MAC, RBAC, ABAC • SQL injection and inference attacks • Encryption (TDE, column-level, row-level), auditing and compliance (GDPR, HIPAA) • **Real-World Use Case:** Building a secure healthcare data system with encrypted medical records and role-based access		

Text and Reference Books

- Hector Garcia-Molina, Jeffrey Ullman, Jennifer Widom – *Database Systems: The Complete Book*
- Raghu Ramakrishnan & Johannes Gehrke – *Database Management Systems*, McGraw-Hill
- Silberschatz, Korth & Sudarshan – *Database System Concepts*, 7th Edition, McGraw-Hill

- Sadalage & Fowler – *NoSQL Distilled*, Addison-Wesley

Learning Experience

Inside Classroom Learning Experience

- Advanced SQL Querying and Optimization
- Master the use of complex SQL queries including subqueries, joins, and set operations.
- Analyze and optimize query performance using tools like query planners and execution plans.
- Understand and apply indexing strategies to enhance performance.
- Transaction Management and Concurrency Control
- Explain the principles of ACID (Atomicity, Consistency, Isolation, Durability) in database transactions.
- Implement transaction management techniques using locking, timestamp ordering, and multi-version concurrency control (MVCC).
- Handle deadlocks and conflicts in concurrent transactions.
- Use Entity-Relationship (ER) models and UML diagrams to design complex databases.
- Balance trade-offs between normalization and denormalization based on real-world needs.
- Database Security and Access Control
- Implement access control techniques, including role-based access control (RBAC) and mandatory access control (MAC).
- Understand database vulnerabilities and apply best practices in database security.
- Secure sensitive data using encryption, authentication mechanisms, and auditing.

Outside Classroom Learning Experience

- Practical Database Application Development

- Develop full-stack applications that integrate with a database backend.
- Implement data storage and retrieval operations within web and mobile applications.
- Explore real-world databases in industries like e-commerce, healthcare, and finance.
- Hands-on Experience with Cloud Databases
- Deploy and manage cloud-based databases using platforms like AWS RDS, Google Cloud SQL, or Azure SQL Database.
- Understand the cost, scalability, and performance considerations of cloud-based database solutions.
- Explore serverless databases and database as a service (DBaaS) options for scaling.

Lab Experiments

S. No	Task
1	**LAB TASK 1: Distributed Inventory Management – Bookstore Chain** **Objective:** To design and simulate a distributed database for managing inventory across multiple bookstores. **Real-World Scenario:** A national bookstore chain with branches in different cities needs a system that manages books, stock, and orders in a distributed way. **Activities:** • Design horizontal fragmentation of inventory data across city-based nodes. • Use PostgreSQL Foreign Data Wrappers (FDW) or MongoDB sharding to distribute and replicate data. • Execute distributed queries and analyze performance with and without parallel

	execution. • Evaluate consistency models: strong, eventual, and causal. **Learning Focus:** Distributed architecture, query optimization, data fragmentation, CAP theorem. **Tools:** PostgreSQL, MongoDB, PgAdmin, DBeaver
2	**LAB TASK 2: Transaction Integrity in E-Commerce System** **Objective:** To ensure transactional consistency and recovery in a multi-module order placement system. **Real-World Scenario:** An e-commerce platform managing orders, payments, and stock updates across multiple services. **Activities:** • Implement transactions using ACID principles in PostgreSQL. • Simulate concurrent transactions with locking (2PL) and MVCC. • Detect and resolve deadlocks using logging and ARIES-based recovery. • Apply two-phase commit (2PC) between inventory and payment services. **Learning Focus:** Transaction isolation, concurrency control, ARIES logging, distributed commit. **Tools:** PostgreSQL, PgAdmin, SQL Workbench, Wireshark (for analysis)
3	**LAB TASK 3: Real-Time IoT Dashboard using Time-Series DB** **Objective:** To store and analyze time-stamped sensor data for industrial applications. **Real-World Scenario:** A manufacturing plant monitoring temperature, humidity, and vibration using IoT sensors.

	Activities: • Design schema and ingest data into InfluxDB or TimescaleDB. • Create continuous queries for real-time metrics and anomaly detection. • Connect to a dashboard tool (Grafana/Flask) to visualize trends. • Compare time-series vs. relational performance for the same workload. **Learning Focus:** Time-series modeling, continuous queries, real-time analytics, dashboard integration. **Tools:** InfluxDB, TimescaleDB, Grafana, Python Flask
4	**LAB TASK 4: Graph-Based Fraud Detection System** **Objective:** To identify suspicious transaction patterns using graph databases. Real-World Scenario: A bank monitors peer-to-peer transactions to detect fraudulent loops and hidden connections. **Activities:** • Model users and transactions as a graph in Neo4j. • Use Cypher queries to find high-frequency transactions, short cycles, and central actors. • Visualize fraud paths using Neo4j Bloom or Gephi. • Benchmark performance vs. equivalent SQL queries in a relational model. **Learning Focus:** Graph modeling, pattern detection, Cypher querying, fraud analytics. **Tools:** Neo4j, Neo4j Bloom, Gephi, Python

5	**CAPSTONE PROJECT: Secure Patient Data System** **Objective:** To build a healthcare database that protects patient privacy and enforces access control. **Real-World Scenario:** A hospital database managing patient records, diagnostics, and role-based access for staff. **Activities:** • Design schema with optimization using views and materialized views. • Implement RBAC for admin, doctor, nurse, and patient roles. • Apply field-level encryption to sensitive fields (e.g., diagnosis). • Simulate attacks like SQL injection and implement countermeasures. **Learning Focus:** Access control models, encryption, security threats, privacy compliance. **Tools:** PostgreSQL, pgcrypto, DBeaver, Python, SQLMap

INFORMATION SECURITY & ETHICAL HACKING

Course Name: Information Security & Ethical Hacking	Course Code	L-T-P	Credits
	ETMCEH174	3-0-2	4
Type of Course:	Major		
Pre-requisite(s), if any:	Fundamentals of computer networks, operating systems, and basic programming knowledge.		

Course Perspective:

This course offers a comprehensive foundation in information security and ethical hacking. It covers key concepts such as network security, cryptographic methods, system vulnerabilities, penetration testing, and cyber laws. Students will learn both defensive and offensive security techniques and gain hands-on experience with tools used by ethical hackers and cybersecurity professionals.

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

Cos	Statements
CO1	Explaining key principles of cybersecurity, threat modeling, and common types of attacks.
CO2	Applying cryptographic techniques and security protocols to secure information systems.
CO3	Performing ethical hacking and penetration testing on networks and applications using standard tools.
CO4	Analyzing and mitigate vulnerabilities using real-world case studies and comply with legal and ethical standards.

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

Course Outline:

Unit Number: 1	**Foundations of Information Security**	**No. of hours: 12**
Topics Covered: • Goals of security: Confidentiality, Integrity, Availability • Threats and vulnerabilities: malware, social engineering, zero-day attacks • Security models: Bell-LaPadula, Biba, Clark-Wilson • Network models and OSI security architecture • **Real-World Use Case:** Risk assessment in a financial institution's internal network		
Unit Number: 2	**Cryptography and Secure Communication**	**No. of hours: 11**
Topics Covered: • Symmetric and asymmetric encryption (AES, DES, RSA) • Hashing algorithms (MD5, SHA family) • Digital signatures and certificates • Public Key Infrastructure (PKI) and SSL/TLS • **Real-World Use Case:** Securing communication between clients and servers in e-commerce		
Unit Number: 3	**Ethical Hacking and Penetration Testing**	**No. of hours: 11**
Topics Covered: • Footprinting and reconnaissance techniques • Scanning and enumeration (Nmap, Nessus) • Exploitation of vulnerabilities using Metasploit • SQL injection, cross-site scripting, password attacks		

• **Real-World Use Case:** Conducting a black-box penetration test on a web-based CRM system		
Unit Number: 4	**Security Countermeasures, Laws, and Compliance**	**No. of hours: 11**
Topics Covered: • Firewalls, IDS/IPS, honeypots, endpoint protection • Secure coding practices and patch management • Indian IT Act, GDPR, HIPAA, PCI DSS • Cyber forensics and incident response • **Real-World Use Case:** Post-breach forensic investigation for a compromised retail website		

Text Books:

- William Stallings – *Cryptography and Network Security*
- Michael T. Goodrich – *Introduction to Computer Security*
- Kevin Mitnick – *The Art of Intrusion*
- Georgia Weidman – *Penetration Testing: A Hands-On Introduction to Hacking*
- EC-Council – *Certified Ethical Hacker (CEH) Official Study Guide*

Learning Experiences

Inside the classroom

Students will be able to:

1. **Understand Core Concepts of Information Security**
   - Learn about confidentiality, integrity, and availability (CIA triad), encryption, hashing, digital signatures, and authentication techniques.

2. **Analyze Cybersecurity Threats and Vulnerabilities**
    - Identify different types of cyberattacks such as phishing, DDoS, SQL injection, and malware.
3. **Apply Ethical Hacking Techniques**
    - Perform penetration testing and vulnerability assessments using industry-standard tools like Kali Linux, Metasploit, Nmap, and Wireshark.
4. **Study Security Protocols and Architectures**
    - Explore security in web applications, wireless networks, firewalls, IDS/IPS, and VPNs.
5. **Understand Legal, Ethical, and Professional Issues**
    - Discuss laws related to cybercrime, ethical responsibilities of ethical hackers, and the impact of hacking on society and organizations.
6. **Hands-on Lab Experience**
    - Execute simulated attacks and defenses in controlled lab environments, enhancing practical skills.

Outside the Classroom

Students will be able to:

1. **Conduct Independent Security Audits**
    - Evaluate the security posture of systems and networks outside the lab through mini-projects or assignments.
2. **Participate in Bug Bounty Programs and CTFs (Capture The Flag)**
    - Apply skills in real-world competitions and platforms like Hack The Box, TryHackMe, or HackerOne.

3. **Raise Awareness about Cyber Hygiene**
    - Educate peers and communities about safe browsing habits, strong passwords, and the importance of data privacy.
4. **Develop Security Policies and Best Practices**
    - Create documentation and security frameworks for small-scale organizations or college systems.
5. **Internship and Industry Engagement**
    - Apply classroom knowledge in professional settings via internships or security consultancy roles.
6. **Explore Ethical Hacking as a Career Path**
    - Build a portfolio of work and certifications (e.g., CEH, OSCP) and prepare for cybersecurity roles.

Lab Experiments

S. No	Task
1	**LAB TASK 1: System Scanning and Vulnerability Assessment** **Objective:** To detect and assess system vulnerabilities using automated tools. **Real-World Scenario:** A network administrator evaluates security posture before an audit. **Activities:** • Scan hosts and ports using Nmap • Identify OS and service versions • Generate vulnerability reports using Nessus • Classify risks using CVSS **Learning Focus:** Vulnerability scanning, threat exposure, reconnaissance **Tools:** Nmap, Nessus, OpenVAS
2	**LAB TASK 2: Exploiting Web Application Vulnerabilities** **Objective:** To identify and exploit common web app vulnerabilities in a test environment. **Real-World Scenario:** A cybersecurity consultant tests an e-commerce platform for OWASP top 10 issues. **Activities:** • Simulate SQL injection and XSS on DVWA or bWAPP • Analyze request/response headers using Burp Suite • Exploit login flaws and session hijacking

	• Propose mitigation strategies **Learning Focus:** Web hacking, request analysis, OWASP Top 10 **Tools:** DVWA, Burp Suite, OWASP ZAP
3	**LAB TASK 3: Password Cracking and Privilege Escalation** **Objective:** To understand how attackers gain unauthorized access and elevate privileges. **Real-World Scenario:** Red team performs internal network audit for password hygiene. **Activities:** • Brute-force attacks using Hydra • Crack password hashes using John the Ripper • Identify privilege escalation vulnerabilities • Hardening strategies and policy design **Learning Focus:** Cracking techniques, hash analysis, privilege abuse **Tools:** Hydra, John the Ripper, Linux enum scripts

4	**LAB TASK 4: Network Sniffing and Secure Protocol Analysis** **Objective:** To analyze live traffic for sensitive data leaks and protocol behavior. **Real-World Scenario:** Blue team validates network encryption policies in a hybrid cloud deployment. **Activities:** • Capture packets using Wireshark • Analyze plain-text vs encrypted protocols (HTTP vs HTTPS, FTP vs SFTP) • Filter and extract credentials from live sessions • Recommend security upgrades **Learning Focus:** Packet analysis, protocol security, encryption validation **Tools:** Wireshark, tcpdump, Netcat
5	**Capstone Project: Simulated Cyberattack & Defense Report** **Objective:** To simulate a complete cyberattack lifecycle including reconnaissance, exploitation, reporting, and patch recommendation in a controlled lab environment. **Project Tasks:** • Perform end-to-end penetration test on a vulnerable system (e.g., Metasploitable2) • Document vulnerabilities found with CVE references • Simulate exploitation and privilege escalation • Develop mitigation and hardening recommendations

	• Present a detailed security audit report **Learning Focus:** Ethical hacking lifecycle, vulnerability reporting, defense planning **Tools:** Metasploit, Nmap, Wireshark, Nessus, Burp Suite, documentation tools

Full Stack Web Development

Course Name: Full Stack Web Development	Course Code	L-T-P	Credits
	ETCCWD175	3-0-2	4
Type of Course:	Major		
Pre-requisite(s), if any: Basic knowledge of programming (preferably in JavaScript/Python), HTML, and CSS.			

Course Perspective. This course enables students to design, develop, and deploy robust and responsive full-stack web applications. The curriculum covers both frontend and backend technologies, RESTful APIs, database integration, authentication systems, and modern development workflows using tools like Git, Docker, and cloud deployment platforms. Emphasis is placed on real-world applications, teamwork, and version control.

The Course Outcomes (COs): On completion of the course theparticipants will be:

COs	Statements
CO1	Designing interactive and responsive user interfaces using HTML, CSS, and JavaScript frameworks.
CO2	Developing scalable backend services and APIs using server-side programming and databases.
CO3	Integrating frontend and backend systems using RESTful APIs, and implement secure authentication and authorization.

CO4	Deploying and managing full-stack applications using cloud platforms, version control, and CI/CD practices.

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

Course Outline:

Unit Number:1	**Title: Frontend Development with HTML, CSS, JavaScript**	**No. of hours:12**
Topics Covered: • HTML5 semantic structure, form validation • CSS3: Flexbox, Grid, Responsive design • JavaScript ES6+: DOM manipulation, events, fetch API • Framework basics: React or Vue.js • **Real-World Use Case:** Designing a responsive online bookstore frontend		
Unit Number:2	**Title: Backend Development with Node.js / Express / Django**	**No. of hours:11**
Topics Covered: • Introduction to server-side frameworks (Node.js/Express or Django) • Routing, middleware, templating engines (EJS, Jinja) • RESTful API creation and error handling • File uploads, sessions, cookies • **Real-World Use Case:** Building a REST API for user registration and data access in a		

blogging platform		
Unit Number:3	**Title: Database Integration and Authentication**	**No. of hours:11**
Topics Covered: • Relational (MySQL/PostgreSQL) vs. NoSQL (MongoDB) databases • Schema design and normalization • CRUD operations with ORM/ODM (Mongoose/Sequelize) • JWT-based authentication, OAuth, RBAC • **Real-World Use Case:** Creating a secure multi-user dashboard with login and database-driven analytics		
Unit Number:4	**Title: DevOps, Testing, and Deployment**	**No. of hours:11**
Topics Covered: • Git workflows, GitHub collaboration • Docker and containerization basics • Deployment using Render/Netlify/Vercel/Heroku • Introduction to CI/CD and unit testing (Jest/PyTest) • **Real-World Use Case:** Team-based CI/CD setup for an e-commerce platform		

Text and Reference Books

- Jon Duckett – *HTML and CSS: Design and Build Websites*
- Brad Traversy – *Modern Full Stack Development*
- Robin Nixon – *Learning PHP, MySQL & JavaScript*
- Ethan Brown – *Web Development with Node and Express*

- Andrew Mead – *The Complete Node.js Developer Course*
- Mozilla Developer Network (MDN) – Web Docs (Free Online Resource)

Lab Experiments

S. No	Task
1	**LAB TASK 1: Responsive Portfolio Website** **Objective:** To design a personal website showcasing academic and project credentials. **Real-World Scenario:** Students create online resumes hosted on GitHub Pages or Netlify. **Activities:** • Design using HTML/CSS Flexbox and Grid • Add interactivity using JavaScript • Make the site mobile-responsive and optimize load speed • Deploy on GitHub Pages **Learning Focus:** Responsive design, accessibility, deployment **Tools:** HTML5, CSS3, JavaScript, GitHub Pages
2	**LAB TASK 2: REST API with Node.js / Django** **Objective:** To implement a CRUD-based API for managing a product catalog. **Real-World Scenario:** An admin dashboard requires secure data entry and retrieval for items and categories. **Activities:** • Create RESTful routes for CRUD operations

	• Add middleware for input validation and error handling • Connect to a local database (MongoDB/MySQL) • Test endpoints using Postman **Learning Focus:** API design, REST principles, backend logic **Tools:** Node.js + Express / Django, MongoDB/PostgreSQL, Postman
3	**LAB TASK 3: User Authentication System** **Objective:** To implement a secure login/register system with hashed passwords and session control. **Real-World Scenario:** Users log in to view personalized dashboards with protected content. **Activities:** • Implement login, register, logout routes • Hash passwords using bcrypt • Manage sessions with JWT and cookies • Role-based access control (e.g., admin/user) **Learning Focus:** Security, sessions, authentication workflows **Tools:** Express-session/JWT, bcrypt, MongoDB

4	**LAB TASK 4: Full Stack Mini Project – ToDo / Blog App** **Objective:** To integrate frontend and backend systems into a complete single-page app. Real-World Scenario: A personal task tracker or blog platform that stores and displays content dynamically. **Activities:** • React frontend with routing and state management • Express/Django backend serving REST API • Connect to database and handle authentication • Deploy full-stack app on Render/Vercel **Learning Focus:** Full-stack integration, project structuring, deployment **Tools:** React, Express/Django, MongoDB/PostgreSQL, Vercel/Render
5	**Capstone Project: Team-Based Web Application with End-to-End Deployment** **Objective:** To collaboratively develop, test, and deploy a scalable full-stack web app (e.g., e-commerce, student portal, helpdesk system). **Project Tasks:** • Collaboratively design frontend and backend in teams • Integrate authentication, database, and secure REST APIs • Deploy live app using GitHub Actions and CI/CD pipeline • Document project with API references and user manual **Learning Focus:** Team coding practices, scalable app design, Git workflows, deployment pipelines **Tools:** MERN/MEVN Stack or Django + React, Docker, GitHub Actions, Netlify/Vercel/Render

Competitive Coding-1

Course Name: Competitive Coding-1	Course Code	L-T-P	Credits
	SEC	2-0-0	2
Type of Course:	Skill Enhancement Course (SEC)		
Pre-requisite(s), if any:	Basic knowledge of programming (Python/C++/Java)		

Course Perspective. To enhance students' problem-solving abilities in competitive coding by providing in-depth knowledge of core data structures, algorithms, and efficient coding techniques. This course aims to prepare students for technical assessments and coding interviews, building a strong foundation for tackling real-world coding challenges.

The Course Outcomes (COs): On completion of the course theparticipants will be:

COs	Statements
CO 1	Applying fundamental and advanced coding techniques to solve problems involving arrays, strings, recursion, matrices, and linked lists.
CO 2	Analyzing and implementing efficient data structure operations, including stacks, queues, and their real-world applications in competitive programming.
CO 3	Evaluating and optimizing problem-solving approaches through comprehensive understanding and revision of key concepts from previous sessions.

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

Course Outline:

Session: 1	**Introduction to competitive programming**	**No. of hours: 2**
Content Summary: Introduction to LeetCode and Codechef coding platforms, Overview of competitive programming, setting up environment, approach to problem solving		
Session: 2	**Array I**	**No. of hours: 2**
Reversing the array, finding maximum and minimum elements, Running sum of 1d Array, count elements with maximum frequency , left/right rotate an array by k positions.		
Session: 3	**Array II**	**No. of hours: 2**
Content Summary: find element in an array, Remove duplicate elements from an sorted array, find repeating element an array, find equilibrium element in an array.		
Session: 4	**Array's Sorting and Time and space complexity Analysis**	**No. of hours: 2**
Content Summary: Bubble sort, selection sort, Insertion Sort and complexity Analysis		
Session: 5	**Array III**	**No. of hours: 2**
Content Summary: union and intersection of sorted arrays, maximum subarray sum (Kadane's Algorithm), maximum product subarray(based on Kandane's) , majority Element (moore's voting algorithm)		
Session: 6	**Strings I**	**No. of hours: 2**
Content Summary: check given string is palindrome or not, count number of vowel and consonant, remove character except alphabet.		
Session: 7	**String II**	**No. of hours: 2**

Content Summary: Calculate frequency of a character, print maximum occurring character in a string, Remove duplicate character from a string, count number of word in a string		
Session: 8	**Recursion I**	**No. of hours: 2**
Content Summary: find factorial, find power of a number, (printing increasing, decreasing and Decreasing Increasing), count digit, sum of array using recursion		
Session: 9	**Recursion II**	**No. of hours: 2**
Content Summary: find pivot index, remove duplicates, fibonacci number, tower of hanoi with recursion tree presentation,		
Session: 11	**Matrix Problems I**	**No. of hours: 2**
Content Summary: Spiral traversal, searching elements in a matrix, Printing elements in sorted order.		
Session: 12	**Matrix Problems II**	**No. of hours: 2**
Content Summary: Finding median in row-wise sorted matrix, identifying rows with maximum 1s , rotating matrices by 90 degrees.		
Session: 13	**LinkedList Introduction.**	**No. of hours: 2**
Content Summary: add Node on any position, delete Node from given position, search Node in a linked List, Count Node in linked List		
Session: 14	**LinkedList I**	**No. of hours: 2**
Content Summary: reverse LinkedList, find mid of the linkedList, Merge Two sorted LinkedList.		
Session: 15	**LinkedList II**	**No. of hours: 2**
Content Summary: add two number, rotate list, remove duplicates from sorted list		
Session: 16	**Stack Implementation**	**No. of hours: 2**

Content Summary: Stack Implementation using Array, Next Greater Element		
Session: 17	**Stack I**	**No. of hours: 2**
Content Summary: Smaller element on left, valid parentheses, Evaluate postfix expression		
Session: 18	**Stack II**	**No. of hours: 2**
Content Summary: min stack, asteroid collision, stock span problem		
Session : 19	**Queue Introduction.**	**No. of hours: 2**
Content Summary: Queue implementation using array, Implement circular queue, queue using stack		
Session :20	**Summary**	
Content Summary: Revising the completed topics and company specific problems on given topics.		
Reference Books: ***Programming Challenges*** – Steven Skiena & Miguel Revilla A gentle introduction to algorithmic problem solving with problems and detailed solutions. ***Competitive Programming (3rd Edition)*** – Steven Halim & Felix Halim Widely recommended for ICPC preparation. Covers data structures, algorithms, and contest strategies.		

DATA ANALYSIS WITH POWER BI & KNIME

Course Name: Data Analysis with Power BI & KNIME	**Course Code**	**L-T-P**	**Credits**
	SEC	2-0-0	2
Type of Course:	Skill Enhancement Course (SEC)		
Pre-requisite(s), if any:	Basic understanding of statistics, spreadsheets, and introductory data analysis concepts.		

Course Perspective: This course provides a hands-on introduction to data analysis and visualization using Power BI and KNIME. Students will learn how to extract, transform, and visualize data, apply analytical models, and generate business insights using both graphical and low-code platforms. The course emphasizes real-world data preparation, dashboard creation, and predictive analysis in a user-friendly environment.

Course Outcomes (COs):

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

CO's	**Statements**
CO 1	Importing, cleaning, and preparing data using Power BI and KNIME workflows.
CO 2	Building dynamic dashboards and visualizations to summarize insights.
CO 3	Applying statistical and machine learning techniques using low-code tools.
CO 4	Evaluating data-driven decisions through reports, KPIs, and business scenarios.

Course Outline:

Unit Number:1	Title: Introduction to Data Analysis and BI Tools	No. of hours:8
Topic Covered: • Types of data and common preprocessing steps • Overview of Business Intelligence platforms • Introduction to Power BI and KNIME interface • **Real-World Use Case:** Evaluating sales trends using imported Excel/CSV data		
Unit Number:2	**Title: Power BI for Interactive Dashboards**	**No. of hours:6**
Topics Covered: • Data loading, shaping with Power Query • Visuals: charts, maps, cards, slicers • Creating calculated fields and KPIs with DAX • Publishing reports to Power BI Service • **Real-World Use Case:** Designing a regional performance dashboard for a retail chain		
Unit Number:3	**Title: KNIME for Visual Workflow-based Analysis**	**No. of hours:8**
Topics Covered: • Introduction to nodes, workflows, and data manipulation • Filtering, grouping, joining datasets • Exploratory Data Analysis (EDA) using built-in nodes • **Real-World Use Case:** Cleaning customer survey data for sentiment analysis		

Unit Number:4	**Title: Basic Predictive Analytics with Power BI & KNIME**	**No. of hours:8**
Topics Covered: • Trend analysis and forecasting in Power BI • Classification and clustering using KNIME (Decision Trees, k-Means) • Model evaluation and reporting • **Real-World Use Case:** Predicting student dropout rates from academic datasets		

Textbooks & References:

1. Alberto Ferrari & Marco Russo, "The Definitive Guide to DAX," Microsoft Press, 2nd Edition, 2020.
2. C. Ott & F. Villwock, "KNIME Beginner’s Luck: A Guide to KNIME Data Analytics Platform," KNIME Press, 2021.
3. Adam Aspin, "Pro Power BI Desktop," Apress, 2nd Edition, 2019.

Learning Outcomes

Inside the Classroom

1. **Understand Core Concepts of Data Analytics Tools:**
   - Grasp foundational concepts in data visualization, ETL (Extract, Transform, Load), and analytics workflows using Power BI and KNIME.
2. **Data Preparation and Transformation:**
   - Learn techniques to clean, integrate, and transform data using Power Query in Power BI

and node-based workflows in KNIME.

3. **Interactive Visualizations and Dashboards:**
   - Develop and publish interactive reports and dashboards in Power BI.
   - Explore storytelling with data using effective visual design.
4. **Automation and Workflow Management:**
   - Create automated workflows in KNIME for repetitive data processing tasks.
   - Understand integration with external data sources (e.g., Excel, databases, APIs).
5. **Hands-on Practice and Case Studies:**
   - Apply techniques through guided labs, real-world case studies, and mini-projects.
   - Perform data modeling, KPI design, and DAX expression writing in Power BI.
6. **Comparison of Tools:**
   - Evaluate strengths and use-cases of Power BI (BI-focused) vs KNIME (data science-focused).

Outside the Classroom

1. **Real-World Data Projects:**
   - Design and execute individual or group mini-projects using publicly available datasets.
   - Build full-cycle analytics projects—from data collection to actionable insights.
2. **Industry Exposure and Certification Preparation:**
   - Engage with online resources (e.g., Microsoft Learn, KNIME Hub).
   - Prepare for certifications such as **Microsoft Power BI Data Analyst Associate** and **KNIME L1/L2 certifications**.
3. **Collaborative Learning and Peer Review:**

- Participate in peer code review sessions and group discussions on workflow design.
- Collaborate on dashboard projects simulating industry environments.

4. **Self-Learning and Exploration:**
   - Encourage use of real-time data from open data platforms.
   - Explore integration of advanced analytics (e.g., predictive modeling in KNIME, AI visuals in Power BI).
5. **Professional Documentation and Reporting:**
   - Practice technical documentation of data analysis workflows and dashboard presentations.
   - Learn to communicate insights effectively to both technical and non-technical audiences.

Verbal Ability

Course Name:	**Course Code**	**L-T-P**	**Credits**
Verbal Ability	**AEC**	2-0-0	2
Type of Course:	Ability Enhancement Course (AEC)		

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

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

COs	Statements
CO1	Understanding the grammar rules and word meaning (Vocabulary).
CO 2	Applying grammar rules and vocabulary in different context & purpose
CO3	Analyzing situations/ context of communication and selecting appropriate grammar and

	words.
CO4	Developing sentences and paragraphs to describe and narrate a situation

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

Course Outline:

Unit Number: 1	**Title: Vocabulary Development and Application**	**No. of hours:10**
Content: Understanding the concept of root words, Prefix and suffix, Ways to enhance Vocabulary, Crosswords and word quizzes, Confusing words, One word substitution, Odd one out, Synonyms and Antonyms, Commonly misspelt words, Idioms and Phrases		
Unit Number: 2	**Title: Fundamentals of Grammar and Sentence Structure**	**No. of hours:8**
Content: Content Summary: Introduction to Parts of Speech, Tenses and its ‘rules, Sentences (Simple, Compound and Complex), Subject Verb Agreement, Pronoun Antecedent agreement, Phrases and Clauses		
Unit Number: 3	**Title: Mastering Sentence Accuracy and Completion Skills**	**No. of hours:7**

Content: Content Summary: Spot the error (grammatical errors in a sentence), Sentence Correction (Improvement of sentences based on Grammar rules), Sentence Completion, Cloze Tests		
Unit Number: 4	**Title: Enhancing Sentence Structure and Reading Comprehension**	**No. of hours:6**
Content: Logical Arrangement of Sentences, Comprehending passages, Contextual questions, Anagrams, Analogies		

Additional Readings:

https://www.indiabix.com/online-test/aptitude-test/

https://www.geeksforgeeks.org/aptitude-questions-and-answers/

https://www.hitbullseye.com/

R1. Norman Lewis – Word Power Made Easy

R2. Wren & Martin – High School English Grammar & Composition

R3. R.S. Agarwal & Vikas Agarwal – Quick Learning Objective General English

R4. S.P. Bakshi - Objective General English

R 5. Praxis Groups -Campus Recruitment Complete Reference

SEMESTER-II

COMPREHENSIVE JAVA PROGRAMMING: FROM BASIC TO ADVANCED

Course Name: Comprehensive Java Programming: From Basic to Advanced	**Course Code**	**L-T-P**	**Credits**
	ETCCJP271	3-0-2	4
Type of Course:	Major		
Pre-requisite(s):	Prior experience with programming logic, object-oriented concepts, and data structures (in any language like C++ or Python).		

Course Perspective:

This course covers the fundamentals to advanced features of Java, ensuring students can develop industry-standard Java applications. It includes topics such as multithreading, Java Collections, JDBC, GUI development, RESTful APIs, security, microservices, and deployment. The hands-on approach emphasizes real-world problem-solving and equips students to build and deploy scalable and secure Java applications.

Defined Course Outcomes

CO1	Implementing core Java concepts, OOP principles, and Java Collections in real-world applications.

CO 2	Developing and optimizimg Java applications using advanced features like multithreading, exception handling, and functional programming.
CO 3	Designing industry-standard applications integrating GUI (JavaFX/Swing), databases (JDBC, Hibernate), and file handling.
CO 4	Building and deploying modern Java applications using Spring Boot, RESTful APIs, microservices, security, and cloud deployment.

Course Outline:

Unit Number:1	**Core Java & Object-Oriented Programming for Real-World Applications**	**No. of hours:12**
Topics Covered: • Introduction to Java, JVM, JDK, and JIT Compilation • Object-Oriented Programming (OOP) in Java: Classes, Objects, Inheritance, Polymorphism, Abstraction, and Encapsulation • Java Collections Framework (List, Set, Map, Queue) for Data Management • Functional Programming in Java: Lambda Expressions, Streams API, Optional Class • Java 8+ Features: Streams, Date & Time API, Completable Future • **Real-World Applications:** • Data processing using Java Collections • Implementing microservices using Java Streams and functional programming		

Unit Number:2	Advanced Java Concepts and Optimized Performance	No. of hours:11

Topics Covered:

- Multithreading & Concurrency: Thread creation, Executors, Callable & Future
- Synchronization, Locks, Atomic Variables, Fork-Join Framework
- Exception Handling: Best practices, logging, custom exceptions
- Reflection API: Dynamic method invocation, annotations processing
- Design Patterns in Java: Singleton, Factory, Observer, Dependency Injection

Real-World Applications:

- Stock market price analyzer using multithreading
- E-commerce order processing system using Factory Design Pattern

Unit Number:3	GUI Development, Database Connectivity & File Handling	No. of hours:11

Topics Covered:

- GUI Programming: JavaFX & Swing, Event Handling, FXML, Styling with CSS
- Database Connectivity (JDBC & Hibernate): CRUD Operations, Connection Pooling, ORM Mapping
- File Handling & Serialization: Handling JSON, XML, CSV, and Object Serialization
- Email & SMS Notification Integration: Java Mail API, Twilio API

Real-World Applications:

- Bank Management System: A GUI-based system with database integration
- Automated Report Generator: Generates PDF/Excel reports for an enterprise

Unit Number:4	RESTful APIs, Microservices & Secure Application Deployment	No. of hours:11
Topics Covered: • RESTful Web Services (Spring Boot): Creating, Consuming APIs, JSON Parsing, Swagger Documentation • Security in Java: JWT Authentication, OAuth2, Secure REST APIs • Role-Based Access Control (RBAC) using Spring Security • Microservices Architecture: Building Microservices with Spring Boot, API Gateway (Zuul/Spring Cloud Gateway) • Deployment & Cloud Integration: Dockerizing a Java Application, Deploying on AWS/GCP using Jenkins **Real-World Applications:** • Real-time order tracking system using REST APIs & WebSockets • Secure online payment system implementing JWT authentication		

Text and Reference Books

- Herbert Schildt – *Java: The Complete Reference*
- Cay Horstmann – *Core Java Volume I – Fundamentals*
- Kathy Sierra & Bert Bates – *Head First Java*
- Y. Daniel Liang – *Introduction to Java Programming*
- Oracle Java SE Documentation (Official)

Learning Outcomes

Inside the Classroom

1. Mastery of Core Java and OOP Principles:
    - Apply fundamental object-oriented concepts like inheritance, polymorphism, abstraction, and encapsulation in real-world coding scenarios.
2. Proficiency in Collections and Functional Programming:
    - Utilize Java Collections Framework and Java 8+ features such as Streams and Lambda expressions for effective data manipulation.
3. Multithreading and Concurrency:
    - Implement multi-threaded applications using Java concurrency tools and understand synchronization mechanisms.
4. Robust Error and Exception Handling:
    - Apply best practices for exception handling, logging, and debugging in scalable applications.
5. Design Patterns and Reflection:
    - Use common design patterns and Java Reflection API to build extensible and dynamic applications.
6. Building GUIs and Managing Databases:
    - Design user interfaces with JavaFX/Swing and connect applications to databases using JDBC and Hibernate ORM.
7. Creating RESTful APIs and Microservices:
    - Develop and secure REST APIs using Spring Boot, implement role-based security, and

explore microservice architecture.

8. Deployment and Integration:
    - Deploy applications using Docker, integrate CI/CD with Jenkins, and publish to cloud platforms like AWS/GCP.

Outside the Classroom

1. Project-Based Learning:
    - Build full-stack Java applications with GUI, database, and backend integration as part of personal or group projects.
2. Exploration of Advanced Libraries and Tools:
    - Independently explore APIs, libraries (e.g., Twilio, Swagger), and frameworks not covered in-depth in lectures.
3. Hands-on Practice with Deployment:
    - Gain experience by deploying personal projects on cloud platforms using Docker and CI/CD pipelines.
4. Self-Driven Certification Preparation:
    - Prepare for certifications such as Oracle Certified Professional: Java SE Programmer, Spring Boot Microservices, or Docker Associate.
5. Peer Collaboration and Code Review:
    - Collaborate with peers on GitHub, participate in team-based software design activities, and provide/receive feedback.
6. Problem Solving in Real-World Scenarios:
    - Apply learned concepts to build systems like payment gateways, reporting tools, and

tracking systems based on real-world needs.

7. Exposure to Industry Practices:
   - Follow industry coding standards, apply secure coding principles, and practice writing production-grade documentation and APIs.

Lab Assignment

S.No.	Lab Task
1	LAB TASK 1: OOP and Exception Handling – Library Management System **Objective:** To implement a library system using class-based design and exception handling. **Activities:** • Create classes for Book, Member, and Library • Implement operations like issue, return, fine calculation • Handle runtime exceptions (e.g., book not found, max limit reached) **Tools:** Java (JDK), VS Code/IntelliJ **Learning Focus:** Object-oriented design, modular classes, exception-driven flow
2	**LAB TASK 2: Collections and File I/O – Product Catalog Application** **Objective:** To manage and persist product data using Java collection frameworks and file streams. **Activities:** • Use ArrayList and HashMap to store and manage products • Implement product search, update, delete functions

	• Save/load data using object streams (serialization) **Tools:** Java I/O, Java Collections, IntelliJ **Learning Focus:** Collections API, file handling, search optimization
3	**LAB TASK 3: GUI with Multithreading – Bank Transaction Simulator** **Objective:** To design a multithreaded GUI simulating real-time bank transactions. **Activities:** • Build JavaFX or Swing interface for bank operations • Use threads for deposit/withdraw actions across accounts • Implement synchronization and GUI feedback for actions **Tools:** JavaFX/Swing, Threads, Event Listeners **Learning Focus:** GUI design, event-driven programming, concurrency
4	**LAB TASK 4: JDBC Integration – Student Portal CRUD System** **Objective:** To build a console-based or GUI system connected to a relational database. **Activities:** • Connect to MySQL using JDBC • Perform CRUD operations via user input • Use PreparedStatement for secure queries **Tools:** JDBC, MySQL/Oracle, IntelliJ **Learning Focus:** Database integration, data persistence, security practices
5	**Capstone Project: Java-Based Inventory & Billing System**

	Objective: To build a fully functional Java application for managing inventory, transactions, and billing using file I/O, database, and GUI integration. **Project Tasks:** • Design reusable classes for products, bills, users • GUI interface for inventory operations • Save records in file/database • Generate invoice in PDF/text format using Java I/O or external libraries **Learning Focus:** Full-cycle Java application, OOP design, GUI + DB integration **Tools:** Java, JavaFX/Swing, JDBC, MySQL, Apache POI (optional)

ADVANCED SOFTWATRE ENGINEERING & AGILE PRACTICES

Course Name:	CourseCode	L-T-P	Credits
Advanced Software Engineering & Agile Practices	ETCCAP272	3-0-2	4
Type of Course:	Major		

Course Perspective. This course provides a hands-on approach to modern Software Engineering principles, focusing on Agile methodologies, DevOps, and secure software development. It covers fundamental software engineering concepts, software project management, software development models, and modern industry practices. Students will learn practical aspects of software development, including requirement analysis, Agile project management, software architecture, continuous integration/deployment (CI/CD), containerization, automated testing, and security best practices. The course emphasizes industry-relevant tools such as GitHub Actions, Jenkins, Docker, and Kubernetes.

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

COs	Statements
CO 1	Applying agile methodologies and frameworks (Scrum, Kanban, XP) to plan and manage software development projects.
CO 2	Designing modular, maintainable, and scalable software systems using advanced architecture and modeling techniques.

CO 3	Implementing automation, testing strategies, and continuous integration pipelines using industry tools.
CO 4	Evaluating and deliver high-quality software through effective collaboration, version control, and feedback loops.

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

Course Outline:

Unit Number:1	**Agile Foundations and Methodologies**	**No. of hours: 12**
Topics Covered: • Agile manifesto and principles • Comparison of agile vs. traditional models (Waterfall, Spiral, V-model) • Scrum roles, ceremonies, artifacts • Kanban, Extreme Programming (XP), Lean Software Development • **Real-World Use Case:** Managing a feature-rich mobile app project with distributed Scrum teams.		
Unit Number: 2	**Software Architecture and Design Patterns**	**No. of hours: 11**
Topics Covered: • Design principles: SOLID, DRY, KISS • UML modeling: use case, sequence, activity, class diagrams		

• Common design patterns: Singleton, Factory, Observer, MVC • Component-based architecture and microservices • **Real-World Use Case:** Designing a scalable architecture for an online ticket booking platform		
Unit Number:3	**Testing, DevOps, and CI/CD**	**No. of hours: 11**
Topic Covered: • Unit testing, integration testing, TDD (Test Driven Development) • CI/CD tools and workflows (Jenkins, GitHub Actions, GitLab CI) • Docker basics for containerization • Automation scripting, test coverage, and quality assurance • **Real-World Use Case:** Automating the build-test-deploy pipeline for a real-time		
Unit Number:4	**Project Management, Collaboration, and Metrics**	**No. of hours: 11**
Topic Covered: • Agile estimation: Story points, planning poker • Burndown charts, velocity, team metrics • Version control with Git and GitHub workflows • Communication and collaboration tools (JIRA, Trello, Slack) • **Real-World Use Case:** Managing a semester-long academic software project with SCRUM and GitHub Projects		

Standard Textbooks:

- Sommerville, I. (2022). *Software Engineering* (11th ed.). Pearson.
- Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill.
- Feathers, M. (2004). *Working Effectively with Legacy Code.* Prentice Hall.

References

- Mano M. Morris, "Computer System Architecture", Pearson.
- CarlHamache, "Computer Organization and Embedded Systems", 6th Edition, McGraw Hill Higher Education

Learning Outcomes

Classroom Learning Experience

- Interactive Lectures: Introduce advanced software engineering concepts and agile principles using PPTs, real-world project scenarios, and interactive discussions.
- Conceptual Understanding: Cover topics such as agile values and principles, Scrum framework, Kanban boards, software design patterns, and software architecture.
- Problem-Solving Sessions: Conduct sessions to practice requirement analysis, user story creation, sprint planning, and system design tasks using agile tools.
- Theory Assignments: Assign tasks on software modeling, agile documentation, and architectural evaluations to reinforce concepts discussed in class.
- Group Work: Collaborate in agile teams to simulate real software development cycles including sprints, retrospectives, and backlog grooming.

- Case Studies: Analyze real-world implementations of agile practices in companies such as Google, Spotify, and Microsoft.
- Continuous Feedback: Use daily stand-ups, peer reviews, and sprint retrospectives to assess project progress and individual contributions.

Outside Classroom Learning Experience

- Theory Assignments: Assign take-home work on software project documentation, quality metrics, and technical debt analysis.
- Lab Projects: Facilitate agile-driven development projects using tools like Git, JIRA, GitHub Projects, and CI/CD platforms like Jenkins or GitHub Actions.
- Question Bank: Provide practice problems and scenario-based questions related to agile workflows, software estimation, testing strategies, and architecture design.
- Online Forums: Create platforms for students to discuss challenges in agile project execution, share best practices, and collaborate on design/code reviews.
- "Computer Architecture and Organization", 3rd Edition by John P. Hayes,WCB/McGraw-Hill
- William Stallings "Computer Organization and Architecture: Designing forPerformance", 10th Edition, Pearson Education

K.R. MANGALAM UNIVERSITY

Lab Experiments

S.No.	Lab Tasks
1	**LAB TASK 1: Scrum Sprint Simulation – Task Tracker App** **Objective:** To simulate agile ceremonies and sprints using JIRA or Trello for managing a mini project. **Activities:** • Define product backlog and sprint goals • Assign team roles (Scrum Master, Product Owner, Developers) • Conduct sprint planning, daily stand-ups, reviews • Use burndown charts to track progress **Learning Focus:** Scrum execution, backlog grooming, sprint reviews **Tools:** JIRA, Trello, Miro
2	**LAB TASK 2: UML and Design Pattern Implementation** **Objective:** To use UML diagrams and implement key design patterns in a modular Java/Python project. **Activities:** • Create use case, class, and sequence diagrams for a given application • Implement Singleton and Factory pattern in code • Show benefits of Observer pattern in GUI updates **Learning Focus:** Software modeling, reusable design, object-oriented design principles **Tools:** StarUML, Lucidchart, Java/Python
3	**LAB TASK 3: DevOps Pipeline for Agile Projects**

	Objective: To set up CI/CD workflow for a full-stack or microservice-based project. **Activities:** • Write unit tests and configure GitHub Actions / Jenkins for CI • Use Docker to containerize the app • Deploy on Heroku/Render/Vercel using automated pipelines **Learning Focus:** CI/CD configuration, test automation, DevOps mindset **Tools:** GitHub Actions, Docker, Jenkins, Heroku
4	**LAB TASK 4: Version Control & Team Collaboration** **Objective:** To manage codebase changes collaboratively using Git and GitHub. **Activities:** • Fork, clone, commit, branch, merge, resolve conflicts • Review and approve pull requests • Link GitHub issues to commits and project boards **Learning Focus:** Version control, collaborative workflows, GitOps **Tools:** Git, GitHub, GitHub Projects
5	**Capstone Project: Agile Delivery of a Software Product** **Objective:** To collaboratively develop, test, deploy, and manage a full-cycle software application using agile and DevOps practices. **Project Tasks:** • Plan and document product backlog and architecture • Model system using UML and implement core features using design patterns

	• Apply CI/CD and containerization for deployment • Maintain user stories, sprints, and retrospectives in JIRA • Present sprint reviews, product demo, and project retrospective **Learning Focus:** End-to-end software development lifecycle with agile, testing, DevOps, and collaboration tools **Tools:** Java/Python + GitHub + JIRA + Docker + GitHub Actions

Cloud Computing

Course Name:	**CourseCode**	**L-T-P**	**Credits**
Cloud Computing	ETMCCC273	3-0-2	4
Type of Course:	Major		
Pre-requisite(s), if any:	Knowledge of computer networks, operating systems, and basic programming concepts.		

Course Perspective. This course introduces students to the fundamentals and applications of cloud computing, covering service models (IaaS, PaaS, SaaS), cloud deployment strategies, virtualization, containerization, and cloud-native development. It emphasizes practical use of public cloud platforms such as AWS, Azure, and Google Cloud, enabling learners to build scalable, reliable, and cost-effective cloud solutions.

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

COs	Statements
CO 1	Explaining the key concepts of cloud computing, service models, and cloud infrastructure.
CO 2	Deploying and managing cloud resources using virtualization, containers, and automation tools.
CO 3	Applying cloud-native design principles using public cloud services for scalable application deployment.

CO 4	Evaluating and implementing secure, cost-optimized, and resilient cloud architectures with real-time monitoring.

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

Course Outline:

Unit Number:1	**Introduction to Cloud Computing and Service Models**	**No. of hours: 12**
Topics Covered: • Definition, characteristics, benefits of cloud computing • Evolution from traditional computing to cloud paradigms • Service models: IaaS, PaaS, SaaS • Deployment models: Public, Private, Hybrid, Community clouds • **Real-World Use Case:** Hosting a startup's website and database using cloud infrastructure		
Unit Number: 2	**Virtualization and Containerization**	**No. of hours: 11**
Topics Covered: • Virtual machines vs containers • Hypervisors: Type 1 and Type 2 • Introduction to Docker and container lifecycle • Kubernetes basics: pods, services, deployments		

• **Real-World Use Case:** Deploying a containerized microservice architecture using Docker & Kubernetes		
Unit Number:3	**Cloud Platforms and DevOps Integration**	**No. of hours: 11**
Topic Covered: • Overview of AWS, Azure, and GCP services • Cloud storage (S3, Blob, Cloud Storage), compute (EC2, Lambda, App Engine) • CI/CD pipelines and Infrastructure as Code (Terraform, CloudFormation) • Monitoring and logging (CloudWatch, Stackdriver, Azure Monitor) • **Real-World Use Case:** Automating deployment and scaling of a web app using AWS CI/CD tools		
Unit Number:4	**Cloud Security, Cost Management, and Design Patterns**	**No. of hours: 11**
Topic Covered • Identity and Access Management (IAM) • Cloud encryption, security groups, firewalls • Cost optimization, pay-as-you-go, billing alerts • Cloud design patterns: Auto-scaling, fault tolerance, redundancy • **Real-World Use Case:** Designing a secure, multi-tiered cloud system for an e-commerce portal		

Text & Reference Books

- Rajkumar Buyya – *Cloud Computing: Principles and Paradigms*
- Thomas Erl – *Cloud Computing: Concepts, Technology & Architecture*
- Michael Hausenblas – *Containers & Kubernetes for Dummies*
- AWS, Azure, and Google Cloud Documentation
- A. Velte – *Cloud Computing: A Practical Approach*

Learning Outcomes

Inside the Classroom

1. **In-depth Understanding of Machine Learning Algorithms:**
   - Analyze and implement advanced algorithms such as ensemble methods, kernel-based learning, and boosting techniques.
2. **Mathematical Foundations and Optimization Techniques:**
   - Understand the role of convex optimization, gradient-based methods, and regularization techniques in improving algorithm performance.
3. **Algorithmic Design and Efficiency:**
   - Design scalable and efficient algorithms for classification, regression, and clustering with a focus on time and space complexity.
4. **Advanced Topics in Model Evaluation:**
   - Apply cross-validation, ROC analysis, and performance metrics tailored to imbalanced and noisy datasets.
5. **Graph-based and Probabilistic Learning:**

- Explore semi-supervised learning, Markov Random Fields, and Graph Neural Networks.

6. **Interpretability and Explainability:**
   - Understand techniques such as SHAP, LIME, and model-agnostic interpretability for black-box models.
7. **Ethical and Fair ML Practices:**
   - Recognize algorithmic bias, fairness, and transparency issues in deploying ML models.

Outside the Classroom

1. **Independent Exploration of Research Papers:**
   - Critically read and summarize recent research on novel ML algorithms from conferences like NeurIPS, ICML, and CVPR.
2. **Capstone and Mini Projects:**
   - Design, implement, and present real-world applications using advanced ML techniques such as fraud detection, recommendation engines, or anomaly detection.
3. **Use of ML Libraries and Frameworks:**
   - Gain practical experience with libraries like Scikit-learn, XGBoost, LightGBM, CatBoost, and TensorFlow/PyTorch for algorithm development.
4. **Kaggle and Competitive Learning:**
   - Participate in data science competitions to apply advanced ML algorithms on structured and unstructured datasets.
5. **Collaborative Learning and Code Sharing:**
   - Collaborate on GitHub, engage in peer code reviews, and contribute to open-source ML toolkits.
6. **End-to-End Model Deployment Skills:**

- Explore tools like MLflow, ONNX, and Docker for tracking, deploying, and scaling machine learning models in production.

7. **Ethics and Responsible AI Practice:**
   - Evaluate real-world case studies to understand the societal impact and limitations of algorithmic decision-making systems.

Lab Assignment

S.No	Lab Tasks
1.	**LAB TASK 1: Cloud Setup and Resource Provisioning on AWS** **Objective:** To create and manage virtual machines, storage buckets, and VPCs using AWS Console. **Activities:** • Launch EC2 instances and configure security groups • Create S3 buckets and upload/download files • Set up VPC and subnets with internet gateway **Learning Focus:** Resource provisioning, compute & storage services **Tools:** AWS Free Tier, EC2, S3, VPC
2	**LAB TASK 2: Docker Containerization and Kubernetes Deployment** **Objective:** To package applications using Docker and deploy them on Kubernetes. **Activities:** • Write Dockerfiles and build container images

	• Push/pull from Docker Hub • Deploy services using Kubernetes manifests • Scale and update deployments using kubectl **Learning Focus:** Container orchestration, deployment automation **Tools:** Docker, Kubernetes (Minikube/k3s), Docker Hub
3	**LAB TASK 3: Serverless Function and CI/CD Integration** **Objective:** To create a serverless function and automate its deployment using CI/CD tools. **Activities:** • Write and deploy a Lambda function or Azure Function • Connect GitHub repository to a CI/CD pipeline (GitHub Actions) • Deploy updated function on commit **Learning Focus:** Serverless computing, automation, GitOps **Tools:** AWS Lambda / Azure Functions, GitHub Actions
4	**LAB TASK 4: Cloud Monitoring and Security Configuration** **Objective:** To configure IAM, monitor resources, and manage logs. **Activities:** • Create IAM users with specific policies • Enable billing alerts and security audits • Monitor resource usage via CloudWatch or Azure Monitor **Learning Focus:** Identity management, billing control, observability **Tools:** IAM, AWS CloudWatch, Azure Monitor
5	**Capstone Project: Cloud-Based Scalable Web Application Deployment**

	Objective: To build, containerize, deploy, and monitor a full-stack web application on a public cloud using best practices in architecture, automation, and security. **Project Tasks:** • Deploy application with Docker and Kubernetes on AWS/GCP • Implement CI/CD using GitHub Actions or Jenkins • Integrate with cloud-native storage and database • Configure logging, alerts, and IAM policies for access control **Learning Focus:** Full-stack deployment, cost-effective scaling, security, and monitoring **Tools:** Docker, Kubernetes, AWS/GCP, GitHub, Terraform (optional)

Generative AI

Course Name: Generative AI	**CourseCode**	**L-T-P**	**Credits**
	ETMCGI274	3-0-2	4
Type of Course:	Major		
Pre-requisite(s), if any:	Prior knowledge of Python programming, linear algebra, probability, machine learning fundamentals, and deep learning basics.		

Course Perspective. This course introduces the foundations and applications of Generative Artificial Intelligence (GenAI), with a focus on deep generative models such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and Transformers. It equips students with practical skills for building GenAI models, applying prompt engineering, fine-tuning language models, and evaluating outputs across modalities like text, images, and audio.

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

COs	Statements
CO 1	Understanding the theoretical foundations and architectures of generative models including GANs, VAEs, and Transformers.
CO 2	Designing and implementing GenAI applications for text, image, and code generation using industry frameworks.
CO 3	Fine-tuning and deploying foundation models using prompt engineering, transfer learning, and open-source APIs.

CO 4	Evaluating generative models using performance metrics, interpretability tools, and ethical guidelines for responsible AI.

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

Course Outline:

Unit Number: 1	**Foundations of Generative Models**	**No. of hours: 12**
Topic Covered: • Generative vs Discriminative models • Latent variable models and sampling • Variational Autoencoders (VAE) – architecture, loss functions • Generative Adversarial Networks (GANs) – generator, discriminator, loss functions, stability • **Real-World Use Case:** Generating new product designs using GANs and VAEs		
Unit Number: 2	**Large Language Models and Transformers**	**No. of hours: 11**
Topic Covered: • Transformer architecture: self-attention, multi-head attention • BERT vs GPT models: masked vs autoregressive • Pretraining and fine-tuning workflows • Introduction to OpenAI, Hugging Face Transformers, and LLaMA • **Real-World Use Case:** Auto-generating technical documentation from code repositories using LLMs		

Unit Number:3	Prompt Engineering and Foundation Model Tuning	No. of hours: 11

Topic Covered:

- Prompt types: zero-shot, few-shot, chain-of-thought
- Instruction tuning and reinforcement learning from human feedback (RLHF)
- Fine-tuning LLMs using parameter-efficient methods (LoRA, adapters)
- Embeddings and retrieval-augmented generation (RAG)
- **Real-World Use Case:** Chatbots for customer service fine-tuned on organizational data

Unit Number:4	Evaluation, Ethics, and Multimodal GenAI	No. of hours: 11

Topics Covered:

- Evaluation metrics: BLEU, FID, perplexity, accuracy
- Hallucination, bias, and interpretability in GenAI
- Copyright, fairness, and ethical AI practices
- Introduction to text-to-image (DALL·E, Stable Diffusion) and audio generation (Voice Cloning, MusicLM)
- **Real-World Use Case:** Evaluating the factuality and safety of AI-generated news summaries

Text & Reference Books

- Ian Goodfellow et al. – *Deep Learning*
- Sebastian Raschka – *Machine Learning with PyTorch and Scikit-Learn*
- Hugging Face Course – *https://huggingface.co/learn*
- Jason Brownlee – *Generative Adversarial Networks with Python*

Learning Outcomes

Inside the Classroom

1. **Understanding Core Concepts:**
    - Explain the fundamentals of Generative AI, including GANs, VAEs, Diffusion Models, and Transformer-based architectures.
    - Differentiate between discriminative and generative models.
2. **Mathematical and Algorithmic Foundations:**
    - Apply probability, statistics, and linear algebra in the context of generative models.
    - Implement and analyze loss functions like adversarial loss, reconstruction loss, and KL divergence.
3. **Model Development Skills:**
    - Design, train, and evaluate generative models using frameworks like TensorFlow or PyTorch.
    - Tune hyperparameters for optimal generation performance.
4. **Ethical and Responsible AI Practices:**
    - Discuss bias, misinformation, and copyright concerns associated with generated content.
    - Evaluate the ethical use of generative models in different domains.
5. **Hands-on Experiments:**
    - Generate images, text, or music using trained models in lab sessions.
    - Conduct comparative performance analysis between different generative techniques.

Outside the Classroom

1. **Practical Applications:**
   - Apply generative AI in real-world scenarios such as content creation, code generation, image synthesis, and medical imaging.
   - Use tools like ChatGPT, DALL·E, Midjourney, or RunwayML to create innovative content.
2. **Collaborative Projects:**
   - Work in teams to develop generative AI-based applications for hackathons or academic projects.
   - Solve open-ended problems using generative techniques with minimal supervision.
3. **Research and Innovation:**
   - Explore recent academic papers, blogs, and case studies to stay current with breakthroughs like Sora, GPT-4o, and diffusion-based systems.
   - Contribute to open-source projects or Kaggle competitions involving generative tasks.
4. **Ethical Reflection and Community Impact:**
   - Reflect on the societal impact of AI-generated content and propose responsible use guidelines.
   - Participate in seminars/webinars or discussions on the future of generative AI.
5. **Portfolio and Career Development:**
   - Build a personal portfolio with generative AI projects (e.g., AI-generated art, AI-written poetry, or synthetic data generation).
   - Gain exposure to industry use cases and prepare for AI/ML roles in tech, media, gaming, healthcare, or marketing.

Lab Assignment

S.No	Lab Tasks
1	**LAB TASK 1: Generating Digits with VAE and GAN (MNIST)** **Objective:** To train a VAE and GAN on image datasets and visualize generated outputs. **Activities:** • Implement a VAE and GAN using PyTorch/TensorFlow • Train on MNIST or Fashion-MNIST dataset • Visualize latent space and generated samples **Learning Focus:** Deep generative modeling, latent vector manipulation **Tools:** Python, PyTorch, Matplotlib, TensorBoard
2	**LAB TASK 2: Building a Text Generator with Transformers** **Objective:** To generate coherent sentences using transformer-based language models. **Activities:** • Load pretrained GPT-2 using Hugging Face • Generate text using top-k sampling and temperature control • Experiment with prompt templates **Learning Focus:** Transformer inference, text generation, sampling strategies **Tools:** Hugging Face Transformers, Python, Google Colab
3	**LAB TASK 3: Prompt Engineering and LLM Fine-tuning**

	Objective: To apply prompt engineering strategies and fine-tune a small language model. Activities: • Test different prompt templates (zero-shot, CoT, few-shot) • Fine-tune a T5 model using LoRA on a custom dataset • Compare performance of tuned vs untuned models **Learning Focus:** Prompt crafting, few-shot learning, tuning best practices **Tools:** PEFT (Parameter Efficient Fine-Tuning), Hugging Face, Google Colab
4	**LAB TASK 4: Text-to-Image Generation and Evaluation** **Objective:** To create visual content from natural language prompts using diffusion models. **Activities:** • Generate images using Stable Diffusion / DALL·E • Evaluate image quality using FID or human feedback • Compare outputs from different models **Learning Focus:** Multimodal generation, prompt engineering, model comparison **Tools:** Stable Diffusion API, OpenAI DALL·E, Python
5	**Capstone Project: Domain-Specific Generative Assistant** **Objective:** To design, train, and deploy a domain-specific assistant (e.g., legal, healthcare, academic) using LLMs with prompt optimization and retrieval-augmented generation. **Project Tasks:**

	• Integrate open-source LLM (e.g., LLaMA or GPT-J) with embeddings and RAG • Collect domain-specific documents for knowledge grounding • Design advanced prompts and test different prompting strategies • Evaluate output for accuracy, bias, and factual grounding **Learning Focus:** LLM orchestration, RAG, prompt tuning, GenAI app delivery **Tools:** LangChain, OpenAI API / LLaMA.cpp, ChromaDB / FAISS, Streamlit / Flask

COMPETITIVE CODING-II

Course Name:	**CourseCode**	**L-T-P**	**Credits**
Competitive Coding-II	**SEC**	2-0-0	2
Type ofCourse:	Skill Enhancement Course (SEC)		
Pre-requisite(s), if any:	Competitive Coding-I, Fundamentals of programming & data structure		

Course Perspective: This course builds upon the fundamentals of competitive programming, focusing on advanced problem-solving techniques, complex data structures, and algorithmic paradigms. It aims to enhance students' ability to solve high-level coding challenges efficiently.

The Course Outcomes (COs). On completion of the course the participants willbeable to:

Cos	Statements
CO 1	Applying advanced string algorithms to solve complex problems.
CO 2	Analyzing and implementing efficient linked list operations and complex problem solutions.
CO 3	Evaluating and applying various tree traversal techniques to solve traversal and view-related problems.

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

Course Outline:

SESSION WISE DETAILS		
Session:1	Advance Array-I	No. of hours: 2
Content summary: Two sum, Best time to buy and sell stocks, Sort 0, 1 and 2(Dutch flag algorithm),		
Session: 2	Advance Array-II	No. of hours: 2
Content Summary: container with most water, merge sorted array, trapping rain water		
Session: 3	Binary Search-I	No. of hours: 2
Content Summary: lower bound , upper bound, koko eating bananas, first bad version		
Session: 4	Binary Search-II	
Content Summary: Search in rotated sorted array, Search in rotated sorted array II, aggressive cows		
Session: 5	Binary Tree Introduction	No. of hours: 2
Content Summary: Introduction of Tree, type of tree, implementation of tree.		
Session: 6	Binary Tree Traversal	No. of hours: 2
Content Summary: Tree Traversal, preorder traversal, inorder traversal, postorder traversal, level order traversal(Morris traversal).		
Session: 7	Binary Tree-III.	No. of hours: 2
Content Summary: Height of the tree, same tree, symmetric tree,		
Session: 8	Binary Tree-IV.	No. of hours: 2
Content Summary: diameter of tree, path sum, print left/right view of Binary tree.		
Session : 9	Binary Search Tree.	No. of hours: 2
Content Summary: Implementation of BST, check valid BST		

Session : 10	Binary Search-II	No. of hours: 2
Content Summary: convert sorted array to BST, Delete node in BST, lowest common ancestor		
Session : 11	Hashmap Introduction.	No. of hours: 2
Content Summary: HashMap Implementation (operations put, get, containsKey, KeySet)		
Session: 12	HashMap-II.	No. of hours: 2
Content Summary: Two Sum, highest frequency character, missing number		
Session:13	HashMap-III.	
Content Summary: intersection of two arrays, set matrix zeros, valid anagram		
Session: 14	hashmap/Sliding window-technique Algorithm	No. of hours:2
Content Summary:longest consecutive sequence, longest substring without repeating character, bulls and cows		
Session: 15	hashmap/Sliding window-technique Algorithm	No. of hours: 2
Content Summary: largest subarray with 0 sum, count of zero sum subarray, length of largest subarray with contiguous element		
Session: 16	Priority Queue	No. of hours: 2
Content Summary: Implementation of Priority queue, min and max Heap		
Session: 17	priority Queue-II	No. of hours: 2
Content Summary: Inplace heap sort, kth largest element, kth smallest element		
Session: 18	priority Queue-III	No. of hours: 2
Content Summary: check max heap, top k frequent element, sliding window maximum		
Session: 19	Sum up Binary tree and Binary search Tree	No. of hours: 2

Content Summary: sum of leaves, top view, bottom view,		
Session: 20	Sum up Hashmap / Sliding window technique.	No. of hours: 2
Content Summary: find all anagram in string, isomorphic string		
Reference Books: • "Introduction to Algorithms" by Cormen, Leiserson, Rivest, and Stein • "Cracking the Coding Interview" by Gayle Laakmann McDowell • "Elements of Programming Interviews" by Adnan Aziz, Tsung-Hsien Lee, and Amit Prakash		

Communication and Personality Development

Course Name: Communication and Personality Development	**Course Code**	**L-T-P**	**Credits**
	AEC	2-0-0	2
Type of Course:	Ability Enhancement Course (AEC)		

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

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

COs	Statements
CO1	Improving public speaking and presentation abilities to confidently convey ideas and information.
CO2	Understanding the framework of Communication to augment oratory skills and written English
CO3	Cultivate essential soft skills required at the different workplaces.

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

Course Outline:

Unit Number:1	**Title: Developing self and others**	**No. of hours:8**
Content Summary: Self Awareness, Personality Concepts (Personality Assessments -Johari Window, Myers & Brigg), Self-Management, Self Esteem, Self-Efficacy, Interpersonal skills, mindset, grit and working in teams.		
Unit Number: 2	**Title: Enhancing Reading and Writing Skills**	**No. of hours:6**
Content Summary: Speed reading and its importance in competitive examinations, techniques for speed reading, note-taking, and critical analysis. Paragraph Writing, Essay and Summary writing, Business Letter, Email writing		
Unit Number: 3	**Title: Effective Communication and Public Speaking**	**No. of hours:7**
Content Summary: Communication Framework, barriers & overcoming these barriers, Group Discussions, Extempore & Public Speaking drills, to manage stage fright and anxiety. Structuring and organizing a presentation (Oral & PPT), Etiquettes, Grooming, Body Language and Conversation starters, TMAY.		
Unit Number: 4	**Title: Career Guide and readiness**	**No. of hours:9**
Content Summary: Cover Letter, ATS friendly resume, Elevator Pitch, Video Resume (Visume), Networking, Group Discussion, Mock Interviews. Capstone Project		

References

R1 Talking to Strangers – Malcom Gladwell

R2 Fierce Conversation - Scot Susan

R3 Public Speaking - William S. Pfeiffer, Pearson

R4 Soft Skills for Everyone – Jeff Butterfield

R5 Business Communication – Rajendra Pal, J S Korlahalli

R6 The power of Positive Attitude -Roger Fritz

R7 Believe in Yourself – Dr. Joseph Murphy

J. Additional Readings

Websites & MOOCs

www.16personalities.com

www.tonyrobbins.com

Specific Research Papers

Applied System Design

Course Name: Applied System Design	Course Code	L-T-P	Credits
	ETCCSD275	4-0-0	4
Type of Course:	Major		
Pre-requisite(s):	Software Engineering principles, Operating Systems, and familiarity with system architecture concepts.		

Course Perspective:

This course provides an in-depth exploration of system design principles and practices in building scalable, robust, and maintainable software systems. It emphasizes design patterns, architectural styles, performance considerations, and integration strategies for distributed systems. Learners will analyze real-world case studies and design blueprints to enhance their ability to design production-ready systems in various domains such as e-commerce, IoT, finance, and cloud-native platforms.

The Course Outcomes (COs).

COs	Statements
CO 1	Describing the principles of applied system design and architectural styles for scalable systems.
CO 2	Applying design patterns and modeling techniques to develop modular and maintainable architectures.
CO 3	Analyzing trade-offs in system design related to scalability, availability, performance, and cost.

CO 4	Designing end-to-end system blueprints based on use cases, functional requirements, and constraints.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Introduction to System Design and Architectural Thinking**	**No. of hours: 15**
Topics Covered: • System design vs software design • Key quality attributes: scalability, availability, reliability, latency, maintainability • Monolithic vs microservices architecture • Real-World Use Case: Scaling a monolithic e-commerce application to microservices		
Unit Number: 2	**Design Patterns and Modeling for Large Systems**	**No. of hours: 15**
Topics Covered: • Object-oriented and component-based design principles • Design patterns: Singleton, Factory, Observer, Proxy, Adapter, Circuit Breaker • UML diagrams: Class, Sequence, Component, Deployment • Real-World Use Case: Designing a payment gateway integration system using design patterns		
Unit Number: 3	**Scalable and Distributed System Design**	**No. of hours: 15**

Topics Covered:		
• Load balancing, caching strategies (LRU, LFU), CDN • Database sharding, replication, and eventual consistency • CAP Theorem and distributed transaction patterns • Real-World Use Case: Designing a real-time collaborative document editing platform		
Unit Number: 4	**System Design Strategies for Performance, Security, and Maintainability**	**No. of hours: 15**
Topics Covered: • Performance tuning: profiling, benchmarking, bottleneck identification • Security by design: authentication, authorization, encryption • API design best practices: REST, GraphQL, throttling, rate-limiting • Maintainability: logging, monitoring, documentation • Real-World Use Case: Designing a secure and observable ride-sharing system backend		

Text and Reference Books:

- Martin Kleppmann – Designing Data-Intensive Applications
- Sam Newman – Building Microservices
- Eric Evans – Domain-Driven Design
- Mark Richards – Software Architecture Patterns
- Bass, Clements & Kazman – Software Architecture in Practice

Learning Outcomes

Inside the Classroom:

1. **Conceptual Understanding:**

- Students will gain deep knowledge of key principles such as modularity, abstraction, scalability, and reliability in system design.
- Analyze system design trade-offs (e.g., consistency vs availability, cost vs performance).

2. **Architectural Thinking:**
   - Apply architectural patterns like layered, microservices, client-server, and event-driven designs through interactive lectures and case-based discussions.
3. **Modelling and Design Patterns:**
   - Construct system models using UML diagrams and apply appropriate design patterns in case scenarios discussed in class.
4. **Analytical and Problem-Solving Skills:**
   - Evaluate the impact of different design choices through classroom design exercises and real-world problem walkthroughs.
5. **Communication and Collaboration:**
   - Develop the ability to communicate system architecture clearly using diagrams, flowcharts, and design documentation during group activities.

Outside the Classroom Learning:

1. **Application of Concepts:**
   - Apply architectural thinking and design strategies to small-scale projects or prototypes as homework or term assignments.
2. **Real-World Case Study Analysis:**
   - Analyze well-known system design case studies (e.g., YouTube, Netflix, Uber) through research-based tasks and reflective write-ups.
3. **Self-Directed Learning:**

- Explore current industry practices in system architecture, distributed systems, and cloud design patterns via online learning platforms (e.g., System Design Primer on GitHub, YouTube lectures by Gaurav Sen).

4. **Technical Communication:**
    - Prepare and present system design proposals outside class hours, improving technical documentation and presentation skills.
5. **Peer Learning and Collaboration:**
    - Collaborate in small groups to critique and improve each other's system design blueprints through informal study groups and peer review sessions.

Minor Project

COURSE NAME: Minor Project	COURSE CODE	L-T-P	CREDITS
	ETCCPR276	0-0-4	2
TYPE OF COURSE:	Project (Proj)		

Introduction:

The objective of Minor Project for the MCA is to provide students with the opportunity to apply theoretical knowledge to real-world societal problems. This course aims to develop students' ability to identify and understand complex societal issues relevant to computer science, engage in critical thinking to formulate and analyze problems, and conduct comprehensive literature reviews to evaluate existing solutions. Through this project, students will enhance their research skills, document their findings in a well-structured manner, and effectively present their analysis and conclusions. Minor project should encourage students to approach problems from multiple perspectives, develop innovative solutions, and improve their communication and documentation skills. Ultimately, the Minor Project-I course seeks to prepare students for future professional challenges by integrating academic knowledge with practical problem-solving experiences.

Duration: 12-16 weeks.

Project must focus on following aspects:

1Standard Operating Procedure (SOP)

Minor Project – I (2 Credits, 12-14 Weeks)

MCA (Master of Computer Application)

2 Purpose

Minor Project immerses second-year students in problem discovery, research, and critical analysis of a real societal challenge addressable via computing.

From the 2025-26 session onward, every step—topic selection, mentoring, submission, feedback, grading, and reporting—will be executed and audited inside Projexa.

This SOP unifies academic requirements with Projexa's digital workflow to guarantee transparency, consistency, and accreditation-ready records.

3 Scope

- Applies to: All MCA students registered for Minor Project.
- Duration: 12–14 teaching weeks (one full semester block).

4 Learning Outcomes (LO)

LO	Student will be able to…	Evidence Captured by Projexa
LO-1	Identify a specific, socially relevant computing problem	"Problem Synopsis" form
LO-2	Conduct & critique a structured literature review	Lit-Review PDF + Gap-Matrix worksheet
LO-3	Analyse & synthesize existing solutions, exposing gaps	Mid-term viva recording + mentor comments
LO-4	Document & present findings in professional formats	Auto-generated tech report + slide decks
LO-5	Operate a project-management platform ethically & professionally	Timestamped activity log, on-time submissions

5 Projexa: Core Functions Used in Minor Project

Module	Purpose
Team Workspace	Topic discussion, mentor chat, file vault
Milestone Engine	Proposal → Mentor Approval → Mid-Review → Final Review
Rubric Builder	Digitised grading templates for mentor & PEC
Analytics Dashboard	Real-time progress, CO/PO attainment, mentor load
Integrity Ledger	Log of late submissions, plagiarism flags, change requests

6 Roles & Responsibilities

Role	Key Responsibilities	Projexa Permissions
Student Team (2–4)	Draft synopsis, upload artefacts, attend vivas, act on feedback, complete reflection survey	Create files, comment, view deadlines
Project Mentor (Faculty)	Weekly guidance, approve milestones, grade mentor components, impose ±3 effort modifier	Approve/Reject, rubric scoring, notes
Project Evaluation Committee (PEC) (3 faculty including Coordinator)	Evaluate Synopsis, Mid-term, Final; moderate mentor marks; resolve disputes	Rubric scoring, moderation tools
Project Coordinator	Configure rubrics & deadlines, monitor cohorts, author reports,	Admin dashboard, deadline override

	manage change-requests	
Dept. Admin	Oversight, accreditation data exports, technical ticket escalation	Read-only analytics, export

7 Semester Timeline (12–14 Weeks)

Week	Status Change in Projexa	Student Deliverable	Mentor / PEC Action
0	Team formation	—	Verify teams
1	Draft → Submitted	1-slide Idea Pitch	Feasibility comment
2	Draft → Submitted	Problem Synopsis (2 pp)	Phase A rubric (Mentor 5 / PEC 15)
3	Mentor-Approved	Revised synopsis (if required)	Mark "Synopsis Approved"
4	—	Literature-Review Dossier + Gap-Matrix	Inline feedback
5	—	Logbook entries	Progress check
6	Mid-Review	Mid-term Deck + Viva	Phase B rubric (Mentor 10 / PEC 20)
7–8	—	Deep-dive analysis, data gathering	Weekly mentor comments
9	—	Draft Tech Report	Mentor inline edits
10	Mentor-Approved	Revised draft report	Set "Ready for Final" flag
11	—	Demo video (≤3 min) rehearsal	Dry-run feedback

12	Final Review	Final Deck + Report	Phase C rubric (Mentor 10 / PEC 30)
13	—	Scholarly evidence (paper / competition)	Phase D score (0–10)
14	Closed	Reflection survey	Grade release, closure

8 Deliverables & Format Standards

Artefact	**Mandatory Format**	**Upload Location**
Idea Pitch	1-slide PDF	Proposal module
Problem Synopsis	PDF (Dept template)	Proposal module
Literature Review	PDF + XLS Gap-Matrix	Docs upload
Mid-term Slides	PPT/PDF (12–15 slides)	Presentation module
Logbook	Auto-captured	Activity Log
Draft & Final Report	IEEE 2-column PDF (8–10 pp)	Report upload
Demo Video	MP4 link (YouTube Unlisted / Drive)	Media tab
Scholarly Output	PDF of submission or award certificate	Evidence upload

9 Evaluation Scheme (100 Marks)

Phase	Timing	Total	Mentor	PEC	Criteria (digital rubric)
A Synopsis	Week 2-3	20	5	15	Problem relevance, objective clarity, presentation quality, Q&A

B Mid-term	Week 6	30	10	20	Lit-review depth, gap analysis, methodology soundness, modern tools, logbook rigour
C Final	Week 12	40	10	30	Findings vs objectives, societal impact, report quality, viva professionalism
D Scholarly / Outreach	Week 13	10	—	10	5 pts for manuscript/competition entry +5 bonus for acceptance/award (max 10)
Continuous Effort Modifier	Whole semester	±3	Mentor	—	Consistent diligence (+) or chronic non-compliance (–)

Rubrics contain 4 performance levels (Excellent, Good, Satisfactory, Poor); descriptors are stored in Projexa's Rubric Builder.

10 Grading & Publication

1. Weighted calculation auto-executes when PEC submits final rubric.
2. Students see marks & comments but cannot edit rubrics.
3. A random 10 % sample is second-marked by another PEC member for moderation.
4. Pass requirement: ≥ 50 % overall AND ≥ 40 % in each Phase A-C.
5. Grades are posted to the LMS via Projexa API within 72 h of final review.

SEMESTER-III

ARITHMETIC AND REASONING SKILLS

Course Name:	Course Code	L-T-P	Credits
Arithmetic and Reasoning Skills	**AEC**	2-0-0	2
Type of Course:	Ability Enhancement Course (AEC)		

Course Perspective: The course aims to improve basic arithmetic skills, speed, and accuracy in mental calculations, and logical reasoning. These abilities are essential for a strong math foundation, helping students succeed in academics and various practical fields.

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

COs	Statements
CO1	Understanding arithmetic algorithms required for solving mathematical problems.
CO2	Applying arithmetic algorithms to improve proficiency in calculations.
CO3	Analyzing cases, scenarios, contexts and variables, and understanding their inter-connections in a given problem.
CO4	Evaluating & deciding approaches and algorithms to solve mathematical & reasoning problems.

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

Course Outline:

Unit Number:1	**Title: Mathematical Essentials**	**No. of hours:15**
Content Summary: Classification of Numbers and Divisibility Rule, Percentage, Ratio and Proportion		
Unit Number: 2	**Title: Fundamentals of Logical Reasoning**	**No. of hours:6**
Content Summary: Blood Relations, Direction Sense, Coding Decoding		
Unit Number: 3	**Title: Elementary Quantitative Skills**	**No. of hours:18**
Content Summary: Simple and Compound Interest, Average, Partnership, Time and Work, Time Speed & Distance.		
Unit Number: 4	**Title: Advanced Quantitative Skills**	**No. of hours:6**
Content Summary: Permutation & Combination, Probability		

References

R1 Talking to Strangers – Malcom Gladwell

R2 Fierce Conversation - Scot Susan

R3 Public Speaking - William S. Pfeiffer, Pearson

R4 Soft Skills for Everyone – Jeff Butterfield

R5 Business Communication – Rajendra Pal, J S Korlahalli

R6 The power of Positive Attitude -Roger Fritz

R7 Believe in Yourself – Dr. Joseph Murphy

COMPREHENSIVE PLACEMENT PREPARATION

Course Name: Comprehensive Placement Readiness	**Course Code**	**L-T-P**	**Credits**
	AEC	2-0-0	2
Type of Course:	Ability Enhancement Course (SEC)		

Course Perspective:

The Comprehensive Placement Preparation Program is strategically designed to foster employability by equipping students with essential skills in aptitude, communication, personal branding, and professional behavior. Rooted in industry-specific demands and global expectations, the program integrates mock placement simulations, digital portfolio development, and structured evaluation to bridge the gap between academic learning and professional readiness.

Key Features of the Course:

- Hands-on workshops on LinkedIn branding, resume writing, and email etiquette.
- Practice-driven sessions on quantitative aptitude, reasoning, and verbal ability tailored to top recruiters.
- Mock interviews, group discussions, and video resume creation aligned with global campus placement formats.
- Emphasis on professional body language, ethics, and industry-aligned communication.
- Learner-centric, outcomes-based approach focused on real-time feedback, peer review, and progressive pedagogy.

The course embodies the university's mission by promoting lifelong learning, nurturing ethical and industry-relevant youth leadership, and fostering entrepreneurial skills through a forward-thinking curriculum.

Course Outcomes (COs)

On successful completion of the course, students will be able to:

CO's	Statement
CO1	Developing a digital professional identity through optimized LinkedIn profiles, customized resumes, and tailored cover letters, showcasing readiness for industry and entrepreneurship.
CO2	Applying quantitative, analytical, and verbal reasoning skills to solve placement-oriented problems, enhancing employability through structured problem-solving approaches.
CO3	Demonstrating effective communication and writing skills, including professional email drafting, paragraph structuring, and vocabulary enhancement, aligning with workplace expectations.
CO4	Displaying confidence, ethical behavior, and professional etiquette during group discussions, mock interviews, and public interactions, reflecting leadership and responsible citizenship.
CO5	Engaging in experiential and outcomes-based learning through practical simulations and peer-reviewed exercises that promote critical thinking, self-assessment, and continuous improvement.

UNIT STRUCTURE

Unit I: Professional Branding & Profiling

- Session 1: Digital Profile Workshop & Photoshoot
- Session 4: Resume & Cover Letter Writing Workshop

- Session 6: Resume & Cover Letter Submission & Feedback
- Session 14: Mock Interview + Video Resume Workshop
- Session 15: Mock PI Round + Student Video Resume Showcase

Unit II: Quantitative & Analytical Reasoning Practice

- Session 2: Ratio, Proportion, Averages, Percentages & Shortcuts
- Session 5: Number & Alphabet Series, Divisibility & Patterns
- Session 8: Time, Work, Time-Speed-Distance & Shortcuts
- Session 11: Remainders, Unit Digits & Last Two Digits
- Session 12: Profit, Loss, S.I., C.I., Discounts & Shortcuts

Unit III: Communication Mastery & Etiquette

- Session 3: Vocabulary Quest – Word Power Enhancement
- Session 9: Email Etiquette + Paragraph Writing Workshop
- Session 10: Professional Etiquette + Body Language Workshop

Unit IV: Placement Simulation, Engagement & Evaluation

- Session 7: Company-Specific Test-1 + Discussion
- Session 13: Group Discussion Workshop + Mock GD Rounds
- Session 14: Mock Interview + Video Resume Workshop
- Session 15: Mock PI Round + Student Video Resume Showcase

.

Discipline-Specific Elective - I

Blockchain Technologies

Course Name: Blockchain Technologies	**Course Code**	**L-T-P**	**Credits**
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Cryptography, distributed systems, computer networks, programming		

Course Perspective:

This course provides a technical and practical introduction to blockchain technology and its ecosystem. It covers consensus mechanisms, smart contracts, decentralized applications (DApps), and blockchain platforms such as Ethereum and Hyperledger.

The Course Outcomes (COs).

COs	**Statements**
CO 1	Explaining the structure and functioning of blockchain technologies and distributed ledgers.
CO 2	Implementing smart contracts and deploy decentralized applications on blockchain platforms.
CO 3	Analyzing consensus algorithms and cryptographic techniques in blockchain systems.
CO 4	Evaluating blockchain use cases, challenges, and their impact on security and privacy.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Blockchain Fundamentals and Architecture**	**No. of hours: 15**
Topics Covered: • Blockchain structure, blocks, hash chaining • Distributed ledger and decentralization • Cryptographic hash functions and Merkle trees • **Real-World Use Case:** Supply chain transparency using blockchain		
Unit Number: 2	**Consensus Mechanisms and Security**	**No. of hours: 15**
Topics Covered: • Proof of Work, Proof of Stake, Delegated Proof of Stake • Byzantine fault tolerance, Sybil attacks, 51% attack • Mining economics and block finality • **Real-World Use Case:** Securing a digital voting system using PoS-based consensus		
Unit Number: 3	**Smart Contracts and Ethereum**	**No. of hours: 15**
Topics Covered: • Ethereum Virtual Machine (EVM) • Solidity programming: data types, functions, events • Gas optimization, Remix IDE, Web3.js • ERC-20 and ERC-721 token standards • **Real-World Use Case:** Creating a decentralized crowdfunding platform		

Unit Number: 4	Blockchain Ecosystem and DApps	No. of hours: 15
Topics Covered: • Hyperledger Fabric basics • IPFS for decentralized file storage • Case studies: NFTs, DeFi, identity, healthcare • Scalability and energy efficiency challenges **Real-World Use Case:** Using blockchain for electronic health record sharing		

Text and Reference Books:

- Martin Kleppmann – Designing Data-Intensive Applications
- Sam Newman – Building Microservices
- Eric Evans – Domain-Driven Design
- Mark Richards – Software Architecture Patterns
- Bass, Clements & Kazman – Software Architecture in Practice

Learning Outcomes

Inside the Classroom

1. **Conceptual Understanding:**
   - Explain the core concepts of blockchain, including distributed ledger, cryptographic hashing, consensus mechanisms, and smart contracts.
   - Differentiate between public, private, and consortium blockchains.
2. **Technical Competency:**
   - Understand the working of key blockchain platforms like Ethereum, Hyperledger Fabric, and Bitcoin.
   - Develop basic smart contracts using Solidity and deploy them on test networks.
3. **Mathematical and Cryptographic Foundations:**

- Apply cryptographic principles such as hash functions, digital signatures, Merkle trees, and public-private key encryption.

4. **Consensus Algorithms:**
   - Analyze and compare consensus algorithms like Proof of Work (PoW), Proof of Stake (PoS), Delegated PoS, and Byzantine Fault Tolerance.
5. **Hands-on Labs:**
   - Perform blockchain setup, wallet creation, transaction validation, and smart contract execution in lab environments.
   - Simulate blockchain networks to understand peer-to-peer communication and ledger updates.
6. **Security and Integrity:**
   - Evaluate vulnerabilities, attacks (e.g., 51% attack, double spending), and countermeasures in blockchain ecosystems.

Outside the Classroom

1. **Real-world Application:**
   - Identify and analyze use cases of blockchain across sectors like supply chain, healthcare, finance, identity management, and voting systems.
   - Build decentralized applications (DApps) with Web3.js, MetaMask, and blockchain APIs.
2. **Project-Based Learning:**
   - Collaborate on capstone or personal projects such as NFT marketplaces, crypto wallets, or smart contract-based voting systems.
   - Integrate blockchain with other emerging techs like IoT, AI, and cloud platforms.
3. **Industry Awareness and Trends:**
   - Track global blockchain trends including DeFi, CBDCs, tokenization, and Layer 2 scaling solutions.
   - Explore new blockchain networks like Polkadot, Avalanche, and Solana.
4. **Entrepreneurial Thinking:**
   - Evaluate blockchain's potential in solving socio-economic problems and explore startup opportunities in the Web3 space.

- Understand the legal, regulatory, and compliance aspects related to crypto assets and blockchain applications.

5. **Professional Development:**
    - Contribute to open-source blockchain projects or communities (e.g., Ethereum Foundation, Hyperledger).
    - Prepare for certifications such as Certified Blockchain Developer, Ethereum Developer Certification, or IBM Blockchain Foundation.

Game Development using Unity and C#

Course Name: Game Development using Unity and C#	Course Code	L-T-P	Credits
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Object-oriented programming (preferably in C# or Java), basic knowledge of graphics and mathematics (vectors, transformations).		

Course Perspective:

This course provides a comprehensive foundation for building 2D and 3D games using the Unity engine and C#. It explores game physics, animation, input systems, UI, level design, and scripting for interactive behaviour. Learners will prototype real-time games and gain practical skills in Unity's component-based architecture.

The Course Outcomes (COs).

COs	Statements
CO 1	Developing 2D and 3D games using Unity's editor, physics engine, and animation tools.
CO 2	Implementing game mechanics and interactivity using C# scripting.
CO 3	Applying game design principles including UI/UX, scene management, and performance optimization.

CO 4	Testing, debugging, and deploying Unity games across multiple platforms.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Unity Environment and 2D Game Development**	**No. of hours: 15**
Topics Covered: • Unity interface, assets, prefabs, GameObjects • Scene creation, lighting, and sprite management • 2D physics: rigidbody, colliders, triggers • Creating tilemaps and camera control • **Real-World Use Case:** Creating a side-scroller game like Flappy Bird		
Unit Number: 2	**Scripting in C# and Game Logic**	**No. of hours: 15**
Topics Covered: • C# for Unity: classes, inheritance, MonoBehaviour lifecycle • Input handling and event-driven design • Game state management and timers • Audio integration and animation controllers • **Real-World Use Case:** Implementing a shooting mechanic in a space invaders game		
Unit Number: 3	**3D Game Development and Physics**	**No. of hours: 15**

Topics Covered:		
• 3D models, lighting, materials, and shadows • Rigidbodies, colliders, gravity, and physics materials • NavMesh for AI pathfinding • Character controllers and camera follow systems • **Real-World Use Case:** Developing a third-person exploration game		
Unit Number: 4	**UI, Testing, and Game Deployment**	**No. of hours: 15**
Topics Covered: • Unity UI system: Canvas, buttons, sliders • Scene transitions, save/load system • Build settings, performance profiling • Publishing to PC/Web/Android • **Real-World Use Case:** Deploying a multiplayer trivia game with score saving		

Text and Reference Books:

- Alan Thorn – *Unity 2021 By Example*
- Jesse Freeman – *Introducing Unity*
- Joseph Hocking – *Unity in Action*
- Mike Geig – *Unity Game Development Cookbook*

Learning Outcomes

Inside the Classroom

1. **Understanding Game Engine Architecture**

   Gain foundational knowledge of Unity's interface, components, and workflow for 2D and 3D game development.

2. **C# Programming for Game Logic**

   Write efficient, reusable C# scripts to control game mechanics, player actions, AI behaviors, and game physics.

3. **Object-Oriented Programming Concepts**

   Apply OOP principles such as inheritance, encapsulation, and polymorphism in game development projects.

4. **Scene and Asset Management**

   Manage game scenes, lighting, assets, and prefabs for optimized design and performance.

5. **Animation and UI Integration**

   Implement animations, transitions, and user interfaces using Unity’s Animator, UI toolkit, and event systems.

6. **Physics and Game Mechanics**

   Use Unity’s physics engine to simulate gravity, collisions, forces, and triggers in gameplay.

7. **Debugging and Optimization**

   Learn to debug, test, and optimize game scripts and performance using Unity Profiler and Visual Studio tools.

8. **Team Collaboration and Version Control**

   Use Git and Unity Collaborate for project sharing and version control in team-based game development.

Outside the Classroom

1. **Participation in Game Jams and Hackathons**

   Apply learned skills in real-world, time-bound challenges like Global Game Jam, enhancing creativity and speed.

2. **Exposure to Industry Trends**

   Follow game dev forums, blogs, and Unity community to stay updated on new features, tools, and best practices.

3. **Indie Game Development**

   Initiate independent projects for Android, iOS, or PC, applying complete development cycles from ideation to deployment.

4. **Collaboration with Artists and Sound Designers**

   Work with peers or online collaborators to integrate 3D models, textures, and audio for immersive experiences.

5. **Learning from Online Tutorials and Documentation**

   Deepen understanding by exploring Unity Learn, YouTube tutorials, and official C# and Unity documentation.

6. **Soft Skills Enhancement**

   Develop time management, creative thinking, communication, and problem-solving abilities during game design iterations.

7. **Commercialization and Publishing**

   Learn how to monetize games using ads, in-app purchases, or premium models, and publish to platforms like Google Play or Steam.

Image Processing and Computer Vision

Course Name: Image Processing and Computer Vision	Course Code	L-T-P	Credits
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Linear algebra, probability, basic signal processing, and Python programming.		

Course Perspective:

This course focuses on foundational and advanced techniques in digital image processing and computer vision. Topics include filtering, feature extraction, segmentation, object detection, and deep learning-based vision techniques. Emphasis is placed on both algorithmic understanding and practical implementation using OpenCV and deep learning frameworks.

The Course Outcomes (COs).

COs	Statements
CO 1	Applying image preprocessing and enhancement techniques to improve visual data quality.
CO 2	Extracting features and detecting objects using classical and modern computer vision algorithms.
CO 3	Implementing deep learning-based vision systems for classification, detection, and

	segmentation.
CO 4	Evaluating and optimizing vision systems using performance metrics and real-world datasets.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Fundamentals of Image Processing**	**No. of hours: 15**
Topics Covered: • Digital image representation and color models • Point operations: histogram equalization, contrast stretching • Filtering: smoothing, sharpening (Gaussian, median, Sobel) • Thresholding and binary image operations • **Real-World Use Case:** Enhancing low-quality CCTV footage for identification		
Unit Number: 2	**Feature Extraction and Matching**	**No. of hours: 15**
Topics Covered: • Edge detection: Canny, Laplacian • Keypoint detectors: Harris, FAST • Descriptors: SIFT, SURF, ORB • Feature matching and RANSAC • **Real-World Use Case:** Building a robust image-matching system for photo deduplication		
Unit Number: 3	**Object Detection and Motion Analysis**	**No. of hours: 15**

Topics Covered: • Contours, bounding boxes, and region proposals • Background subtraction and optical flow • Object detection using Haar cascades and YOLO • Tracking: Kalman filters, mean-shift • **Real-World Use Case:** Detecting and tracking vehicles from drone footage		
Unit Number: 4	**Deep Learning for Computer Vision**	**No. of hours: 15**
Topics Covered: • CNN architectures: LeNet, AlexNet, ResNet • Image classification and segmentation • Transfer learning and fine-tuning • Evaluation metrics: IoU, mAP, F1-Score • **Real-World Use Case:** Real-time facial recognition system using pretrained CNNs		

Text and Reference Books:

- Rafael C. Gonzalez, Richard E. Woods – *Digital Image Processing*
- Richard Szeliski – *Computer Vision: Algorithms and Applications*
- Simon J. D. Prince – *Computer Vision: Models, Learning, and Inference*
- Adrian Rosebrock – *Practical Python and OpenCV*
- Ian Goodfellow – *Deep Learning* (for vision with neural networks)

Learning Outcomes

Inside the Classroom

1. **Fundamentals of Image Processing**

   Understand core concepts such as image representation, color models, histograms, and image enhancement techniques.

2. **Image Filtering and Transformation**

   Apply spatial and frequency domain filters (e.g., Gaussian, Sobel, Fourier Transform) for image smoothing and sharpening.

3. **Feature Detection and Extraction**

   Detect edges, corners, blobs, and keypoints using algorithms like Canny, Harris, SIFT, and SURF.

4. **Image Segmentation and Morphological Operations**

   Segment images using thresholding, region growing, and morphological techniques like dilation, erosion, opening, and closing.

5. **Camera Geometry and 3D Vision**

   Learn principles of camera models, calibration, epipolar geometry, and stereo vision for depth estimation.

6. **Motion Analysis and Object Tracking**

   Implement optical flow, background subtraction, and tracking algorithms such as Kalman filter and Meanshift.

7. **Machine Learning in Vision**

   Integrate basic machine learning algorithms for classification and recognition tasks in visual datasets.

8. **Programming with OpenCV and Python**

   Use OpenCV, NumPy, and related Python libraries for implementing and visualizing vision algorithms.

Outside the Classroom

1. **Practical Projects and Applications**

   Design real-world applications like face recognition, number plate detection, or augmented reality apps.

2. **Research and Innovation**

   Explore research papers and contribute to ongoing projects in biomedical imaging, surveillance, or autonomous vehicles.

3. **Competitions and Hackathons**

   Participate in AI/vision-based competitions (e.g., Kaggle challenges, Smart India Hackathon) to apply skills under real constraints.

4. **Self-learning through Online Courses and Tutorials**

   Deepen skills using platforms like Coursera, Udemy, YouTube, and OpenCV documentation.

5. **Building OpenCV and ML Portfolios**

   Develop and showcase projects on GitHub, demonstrating integration of image processing with deep learning models.

6. **Collaborations and Interdisciplinary Work**

   Work with robotics, AI, and hardware teams on vision-based projects such as gesture control or drone navigation.

7. **Industry Exposure and Internships**

   Apply theoretical knowledge during internships in industries like healthcare, surveillance, and automotive vision systems.

8. **Ethical and Societal Awareness**

   Understand privacy, surveillance ethics, and responsible AI usage in visual data collection and processing.

Natural Language Processing

Course Name: Natural Language Processing	Course Code	L-T-P	Credits
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Python programming, machine learning, probability/statistics, linear algebra.		

Course Perspective:

This course introduces natural language processing (NLP) for understanding and generating human language using rule-based, statistical, and deep learning methods. It includes tokenization, parsing, sentiment analysis, word embeddings, and language modeling using modern NLP frameworks.

The Course Outcomes (COs).

COs	**Statements**
CO 1	Understanding and applying foundational techniques in text preprocessing and representation.
CO 2	Building statistical and deep learning models for NLP tasks like classification, translation, and generation.
CO 3	Analyzing semantic, syntactic, and contextual relationships using embeddings and transformers.
CO 4	Evaluating NLP systems with standard metrics and explore ethical implications in

	language technologies.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Text Processing and Linguistic Foundations**	**No. of hours: 15**
Topics Covered: • Text normalization: tokenization, stemming, lemmatization • POS tagging, chunking, parsing • Language models: n-gram, smoothing techniques • **Real-World Use Case:** Auto-summarizing product reviews		
Unit Number: 2	**Classical NLP Models**	**No. of hours: 15**
Topics Covered: • Bag-of-Words, TF-IDF • Naïve Bayes, Logistic Regression for text classification • Word2Vec, GloVe, similarity measures • **Real-World Use Case:** Spam classification using TF-IDF and Naïve Bayes		
Unit Number: 3	**Deep Learning for NLP**	**No. of hours: 15**
Topics Covered: • RNNs, LSTMs, GRUs for sequence modeling • Attention and Transformer architectures • Pretrained language models (BERT, GPT) • **Real-World Use Case:** Chatbot development using a transformer-based encoder-decoder model		

Unit Number: 4	Evaluation, Ethics, and Applications	No. of hours: 15
Topics Covered: • BLEU, ROUGE, perplexity, F1-score • Bias, fairness, hallucination, disinformation • Applications: summarization, translation, sentiment analysis • **Real-World Use Case:** Evaluating a political speech summarization model		

Text and Reference Books:

- Jurafsky & Martin – *Speech and Language Processing*
- Jacob Eisenstein – *Introduction to Natural Language Processing*
- Yoav Goldberg – *Neural Network Methods in NLP*

Learning Outcomes

Inside the Classroom

1. **Text Preprocessing Techniques**

   Understand and apply preprocessing steps like tokenization, stemming, lemmatization, POS tagging, and stop-word removal.

2. **Linguistic Foundations of NLP**

   Learn about morphology, syntax, semantics, and pragmatics to analyze and interpret natural language structures.

3. **Feature Engineering and Text Representation**

   Implement Bag-of-Words, TF-IDF, and word embeddings (e.g., Word2Vec, GloVe) for converting text into numerical features.

4. **Language Modeling**

   Understand n-gram models, probabilistic language models, and introduction to neural language models.

5. **Text Classification and Sentiment Analysis**

   Build supervised NLP models using Naive Bayes, Logistic Regression, and SVM for text classification tasks.

6. **Named Entity Recognition (NER) and Information Extraction**

   Extract entities, relationships, and events from text using rule-based and statistical approaches.

7. **Parsing and Syntax Analysis**

   Use dependency and constituency parsers to analyze sentence structures and build parse trees.

8. **Hands-on Programming with Python Libraries**

   Gain proficiency in using NLTK, spaCy, scikit-learn, and Hugging Face Transformers for developing NLP pipelines.

Outside the Classroom

1. **Applied Projects and Capstone Work**

   Develop real-world NLP applications like chatbots, resume parsers, fake news detectors, or recommendation systems.

2. **Participation in NLP Competitions**

   Join online challenges on platforms like Kaggle, AIcrowd, and Zindi to practice on real datasets.

3. **Exploring NLP Research and Trends**

   Read and review academic papers from ACL, NAACL, and arXiv to stay updated on emerging techniques like LLMs and prompt engineering.

4. **Open Source Contributions**

   Contribute to NLP libraries or tools like spaCy, Hugging Face, or fastText via GitHub and issue trackers.

5. **Learning Through MOOCs and Tutorials**

   Enhance understanding through specialized courses from Stanford NLP, DeepLearning.AI, or fast.ai.

6. **Interdisciplinary Applications**

   Apply NLP in other domains such as healthcare (clinical NLP), law (legal text analysis), or education (automated grading).

7. **Ethics and Bias in Language Models**

   Investigate ethical implications such as algorithmic bias, misinformation, and fairness in NLP models.

8. **Building NLP Portfolio and Blog Writing**

   Document learning and showcase projects via GitHub repos, personal blogs, or technical write-ups on Medium/Dev.to.

Principles and Practices of System Design

Course Name: Principles and Practices of System Design	Course Code	L-T-P	Credits
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Computer architecture, software engineering, data structures, and object-oriented programming.		

Course Perspective:

This course explores the systematic design of large-scale, modular, and maintainable systems. It emphasizes design principles, system modeling, architectural trade-offs, fault-tolerance, scalability, and deployment strategies. Students will develop a practical understanding of how to architect real-world software and hardware-integrated systems.

The Course Outcomes (COs).

COs	Statements
CO 1	Applying system design principles to build scalable, modular, and high-performance systems.
CO 2	Modeling system components and interactions using design abstractions and architectural patterns.
CO 3	Evaluating trade-offs in performance, reliability, cost, and maintainability of design

	choices.
CO 4	Documenting, simulating, and validating systems using industry-standard tools and methodologies.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Introduction to System Design Principles**	**No. of hours: 15**
Topics Covered: • System complexity, decomposition, and abstraction • Coupling and cohesion • Design for maintainability and extensibility • Reusability, modularity, separation of concerns • **Real-World Use Case:** Designing a modular system for online education platforms		
Unit Number: 2	**Architectural Design and System Modelling**	**No. of hours: 15**
Topics Covered: • System architecture styles: layered, client-server, microservices • Unified Modeling Language (UML) diagrams • Component, sequence, and deployment diagrams • System modeling using SysML • **Real-World Use Case:** Architecting a scalable multi-tier banking platform		
Unit Number: 3	**Fault Tolerance, Scalability, and Performance**	**No. of hours: 15**

Topics Covered: • Redundancy and failover design • Scalability patterns: horizontal/vertical scaling, partitioning • Load balancing, caching, asynchronous messaging • Latency vs throughput trade-offs • **Real-World Use Case:** Building a resilient video streaming infrastructure		
Unit Number: 4	**Design Validation, Documentation, and Tools**	**No. of hours: 15**
Topics Covered: • Prototyping and simulation strategies • Design documentation and review practices • Performance benchmarking and profiling • Tools: Lucidchart, StarUML, PlantUML, Figma for system mockups • **Real-World Use Case:** Preparing a system design document for a logistics platform		

Text and Reference Books:

- Eberhardt Rechtin – *Systems Architecting: Creating & Building Complex Systems*
- Ian Sommerville – *Software Engineering*
- Rajkumar Buyya – *Designing Systems for Internet of Things*
- Grady Booch – *Object-Oriented Analysis and Design with Applications*
- Martin Fowler – *Patterns of Enterprise Application Architecture*

Learning Outcomes

Inside the Classroom

1. **Understanding System Design Fundamentals**

   Grasp the foundational principles such as modularity, abstraction, scalability, fault tolerance, and cohesion.

2. **Design Patterns and Architectural Styles**

   Learn and apply common design patterns (e.g., Singleton, Factory, MVC) and architectural styles (e.g., layered, microservices, client-server).

3. **Requirements Analysis and Specification**

   Understand functional and non-functional requirements, and develop system specifications through use cases and user stories.

4. **Component-Based and Service-Oriented Design**

   Model system components and their interactions using UML diagrams, sequence diagrams, and interface definitions.

5. **Scalability and Performance Considerations**

   Analyze how systems scale horizontally/vertically and design for high availability, load balancing, and caching.

6. **Security and Reliability in Design**

   Incorporate secure design principles and mechanisms for system reliability, recovery, and failure handling.

7. **Software Design Life Cycle**

   Apply the principles of software/system development life cycle (SDLC) and iterative design processes like Agile and DevOps.

8. **Hands-On Case Studies and Mini-Projects**

   Engage in in-class exercises and case studies involving the design of e-commerce platforms, social media apps, or distributed systems.

Outside the Classroom

1. **Capstone Design Projects**

   Undertake full-fledged system design projects (e.g., event management systems, real-time chat apps, ride-sharing systems) from scratch to deployment.

2. **Interview and Industry Preparation**

   Prepare for system design interviews by solving problems like designing Twitter, Uber, Netflix, or a scalable URL shortener.

3. **Exploring Scalable Cloud Architectures**

   Learn about cloud-based design solutions using AWS, Azure, or GCP for real-world deployment and scaling.

4. **Interdisciplinary Collaborations**

   Collaborate with business or product students to analyze requirements, propose solutions, and build prototypes.

5. **Community Engagement and Code Reviews**

   Participate in open-source projects, GitHub reviews, and tech forums like Stack Overflow and Reddit's r/systemdesign.

6. **Learning from Real-World Systems**

   Study and document architectural decisions from companies like Google, Amazon, or Netflix through engineering blogs and whitepapers.

7. **Ethics, Privacy, and Sustainability**

   Understand ethical considerations such as privacy by design, data minimization, and sustainable system design practices.

8. **Technical Blogging and Documentation**

   Write detailed design documentation or blog posts explaining your design decisions and architectural trade-offs.

Swarm Intelligence and Nature-Inspired Optimization

Course Name: Swarm Intelligence and Nature-Inspired Optimization	Course Code	L-T-P	Credits
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Algorithms, optimization techniques, linear algebra, probability.		

Course Perspective:

This course introduces swarm intelligence and nature-inspired metaheuristics like Ant Colony Optimization, Particle Swarm Optimization, Genetic Algorithms, and Artificial Bee Colony. It emphasizes their application in real-world optimization problems across engineering, logistics, and machine learning.

The Course Outcomes (COs).

COs	Statements
CO 1	Understanding and simulating swarm-based and evolutionary optimization algorithms.
CO 2	Applying population-based methods to solve real-world constrained and unconstrained optimization problems.
CO 3	Comparing the effectiveness of nature-inspired approaches against classical optimization methods.

CO 4	Designing hybrid or domain-adapted algorithms for multi-objective optimization problems.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Optimization and Metaheuristics Overview**	**No. of hours: 15**
Topics Covered: • Optimization problem types: linear, nonlinear, discrete, combinatorial • Deterministic vs stochastic methods • Exploration vs exploitation • **Real-World Use Case:** Parameter tuning for machine learning algorithms		
Unit Number: 2	**Evolutionary Algorithms**	**No. of hours: 15**
Topics Covered: • Genetic algorithms: crossover, mutation, fitness selection • Selection strategies: roulette wheel, tournament • Elitism and convergence analysis • **Real-World Use Case:** Scheduling staff for multiple shifts with constraints		
Unit Number: 3	**Swarm Intelligence Algorithms**	**No. of hours: 15**
Topics Covered: • Ant Colony Optimization for shortest path problems • Particle Swarm Optimization for continuous domains • Artificial Bee Colony algorithm • **Real-World Use Case:** Optimal routing of delivery vehicles using PSO		

Unit Number: 4	Hybrid and Multi-objective Optimization	No. of hours: 15
Topics Covered: • NSGA-II for multi-objective problems • Hybridization strategies: combining GAs and PSO • Applications in feature selection, IoT, robotics, bioinformatics • **Real-World Use Case:** Energy optimization in sensor networks using ABC + PSO hybrid		

Text and Reference Books:

- Kennedy & Eberhart – *Swarm Intelligence*
- Kalyanmoy Deb – *Multi-objective Optimization using Evolutionary Algorithms*
- Marco Dorigo – *Ant Colony Optimization*
- Xin-She Yang – *Nature-Inspired Metaheuristic Algorithms*
- Goldberg – *Genetic Algorithms in Search, Optimization and Machine Learning*

Learning Outcomes

Inside the Classroom

1. **Foundations of Swarm Intelligence**

   Understand the biological and behavioral basis of swarm systems including ant colonies, bird flocking, and fish schooling.

2. **Exploration of Key Algorithms**

   Study and implement core nature-inspired algorithms such as:

   - Particle Swarm Optimization (PSO)

- Ant Colony Optimization (ACO)
- Bee Colony Optimization
- Firefly Algorithm
- Bat Algorithm

3. **Mathematical Modelling of Optimization Problems**

   Formulate real-world problems as objective functions and apply nature-inspired algorithms to find near-optimal solutions.

4. **Performance Evaluation and Convergence Analysis**

   Analyze the performance, convergence behavior, and parameter tuning of different algorithms across benchmark functions.

5. **Hybrid and Multi-objective Optimization**

   Explore hybrid approaches (e.g., PSO-GA) and techniques for solving multi-objective optimization problems using Pareto fronts.

6. **Hands-On Simulation and Coding**

   Implement swarm algorithms using Python/Matlab and apply them to optimization problems in scheduling, routing, or clustering.

7. **Comparative Analysis**

   Compare nature-inspired methods with classical techniques like gradient descent, linear programming, and genetic algorithms.

8. **Ethical and Theoretical Understanding**

   Discuss the ethical implications of autonomous decision-making and the theoretical limitations of metaheuristic algorithms.

Outside the Classroom

1. **Capstone Projects on Real-World Optimization**

   Design and implement solutions using swarm intelligence for applications like:

   - Smart traffic routing
   - Task scheduling in cloud computing
   - Path planning for robotics
   - Energy management in IoT systems

2. **Participation in Research and Innovation**

   Read, summarize, and contribute to ongoing research in bio-inspired computing, optimization journals, or IEEE conferences.

3. **Competitions and Hackathons**

   Compete in global coding contests and AI/optimization challenges on platforms like Kaggle, CodaLab, or IEEE Xplore datasets.

4. **Collaborative Interdisciplinary Projects**

   Apply swarm techniques in fields like environmental modeling, bioinformatics, logistics, or economics.

5. **Self-Learning through Simulation Tools**

   Explore frameworks and tools such as DEAP (Python), MATLAB toolboxes, or NetLogo for simulating agent-based systems.

6. **Open Source Contributions and Algorithm Development**

   Develop, document, and publish custom variants of nature-inspired algorithms on GitHub for community use.

7. **Ethics and Responsible Use**

   Reflect on responsible algorithm design especially in autonomous systems, swarm robotics, and evolutionary modelling.

8. **Presentation and Documentation Skills**

   Create academic posters, technical blogs, or video demonstrations to communicate project outcomes and innovations.

Discipline-Specific Elective - II

Advanced Computer Networking

Course Name: Advanced Computer Networking	Course Code	L-T-P	Credits
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Computer Networks, TCP/IP, OSI model, basic routing/switching.		

Course Perspective:

This course offers a deep dive into enterprise and next-generation networking. It includes advanced routing, network security, software-defined networking (SDN), and performance evaluation using simulation/emulation tools.

The Course Outcomes (COs).

COs	Statements
CO 1	Analyzing routing protocols, QoS, and MPLS for high-performance networks.
CO 2	Designing secure and scalable architectures using firewalls, IDS/IPS, and VPNs.
CO 3	Evaluating modern networking technologies like SDN, NFV, and data center fabrics.
CO 4	Simulating networks and assess performance metrics using network emulation tools.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Advanced Routing and Switching**	**No. of hours: 15**
Topics Covered: • OSPF, BGP, EIGRP, VLANs • MPLS, NAT, tunnelling • **Real-World Use Case:** Routing optimization for content delivery networks		
Unit Number: 2	**Network Security and Management**	**No. of hours: 15**
Topics Covered: • Firewall design, IDS/IPS, ACLs • VPNs, IPsec, packet filtering • **Real-World Use Case:** Implementing a secure perimeter for a corporate intranet		
Unit Number: 3	**Software-Defined Networking and Virtualization**	**No. of hours: 15**
Topics Covered: • SDN principles, OpenFlow, controllers • Network Function Virtualization (NFV) • Data center network architecture • **Real-World Use Case:** Managing traffic flows dynamically in a cloud data center using SDN		
Unit Number: 4	**Performance Evaluation and Simulation**	**No. of hours: 15**
Topics Covered: • NS2/NS3, Mininet, Wireshark • Throughput, latency, jitter, loss • **Real-World Use Case:** Simulating real-time VoIP traffic under congestion scenarios		

Text and Reference Books:

- Kurose & Ross – *Computer Networking: A Top-Down Approach*
- Peterson & Davie – *Computer Networks: A Systems Approach*
- Feamster et al. – *Software-Defined Networking*

Learning Outcomes

Inside the Classroom

1. **Comprehend Advanced Protocols**

   Understand the working of advanced networking protocols including BGP, MPLS, and IPv6, and analyze their roles in modern network architecture.

2. **Design Scalable Networks**

   Apply concepts of routing, switching, and subnetting to design scalable and fault-tolerant enterprise networks.

3. **Evaluate Network Performance**

   Use tools and models to measure and optimize network performance parameters such as latency, jitter, and throughput.

4. **Implement Secure Networking Practices**

   Demonstrate knowledge of advanced security mechanisms like VPNs, IPsec, firewalls, and intrusion detection/prevention systems (IDS/IPS).

5. **Hands-on Configuration and Troubleshooting**

   Configure routers, switches, and other networking devices in simulated environments using tools such as Cisco Packet Tracer or GNS3.

6. **Apply QoS and Traffic Engineering**

   Understand and implement Quality of Service policies and traffic shaping mechanisms to prioritize network traffic effectively.

Outside the Classroom

1. **Real-World Network Simulation Projects**

   Engage in mini-projects or capstone activities to design and simulate real-world network topologies with redundancy, load balancing, and failover mechanisms.

2. **Industry-Oriented Certifications**

   Prepare for and pursue certifications such as Cisco CCNP, CompTIA Network+, or Juniper Networks certifications to enhance employability.

3. **Collaborative Learning and Peer Teaching**

   Participate in group discussions, peer tutoring, and network troubleshooting workshops to foster collaborative learning.

4. **Network Security Awareness**

   Stay updated with emerging threats and vulnerabilities by reading cybersecurity blogs, CVE databases, and security advisories.

5. **Explore Open-Source Network Tools**

   Experiment with open-source network analysis tools like Wireshark, Nmap, and SNORT for deeper understanding of traffic and attacks.

6. **Engage with Online Networking Communities**

   Participate in forums such as Stack Exchange, Cisco DevNet, or Reddit's networking communities to discuss problems and gain insights from professionals.

Swarm Intelligence and Nature-Inspired Optimization

Course Name: Swarm Intelligence and Nature-Inspired Optimization	Course Code	L-T-P	Credits
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Algorithms, optimization techniques, linear algebra, probability.		

Course Perspective:

This course introduces swarm intelligence and nature-inspired metaheuristics like Ant Colony Optimization, Particle Swarm Optimization, Genetic Algorithms, and Artificial Bee Colony. It emphasizes their application in real-world optimization problems across engineering, logistics, and machine learning.

The Course Outcomes (COs).

COs	Statements
CO 1	Understanding and simulating swarm-based and evolutionary optimization algorithms.
CO 2	Applying population-based methods to solve real-world constrained and unconstrained optimization problems.
CO 3	Comparing the effectiveness of nature-inspired approaches against classical optimization methods.

CO 4	Designing hybrid or domain-adapted algorithms for multi-objective optimization problems.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Optimization and Metaheuristics Overview**	**No. of hours: 15**
Topics Covered: • Optimization problem types: linear, nonlinear, discrete, combinatorial • Deterministic vs stochastic methods • Exploration vs exploitation • **Real-World Use Case:** Parameter tuning for machine learning algorithms		
Unit Number: 2	**Evolutionary Algorithms**	**No. of hours: 15**
Topics Covered: • Genetic algorithms: crossover, mutation, fitness selection • Selection strategies: roulette wheel, tournament • Elitism and convergence analysis • **Real-World Use Case:** Scheduling staff for multiple shifts with constraints		
Unit Number: 3	**Swarm Intelligence Algorithms**	**No. of hours: 15**
Topics Covered: • Ant Colony Optimization for shortest path problems • Particle Swarm Optimization for continuous domains • Artificial Bee Colony algorithm • **Real-World Use Case:** Optimal routing of delivery vehicles using PSO		

Unit Number: 4	Hybrid and Multi-objective Optimization	No. of hours: 15
Topics Covered: • NSGA-II for multi-objective problems • Hybridization strategies: combining GAs and PSO • Applications in feature selection, IoT, robotics, bioinformatics • **Real-World Use Case:** Energy optimization in sensor networks using ABC + PSO hybrid		

Text and Reference Books:

- Kennedy & Eberhart – *Swarm Intelligence*
- Kalyanmoy Deb – *Multi-objective Optimization using Evolutionary Algorithms*
- Marco Dorigo – *Ant Colony Optimization*
- Xin-She Yang – *Nature-Inspired Metaheuristic Algorithms*
- Goldberg – *Genetic Algorithms in Search, Optimization and Machine Learning*

Learning Outcomes

Inside the Classroom

9. **Foundations of Swarm Intelligence**

   Understand the biological and behavioral basis of swarm systems including ant colonies, bird flocking, and fish schooling.

10. **Exploration of Key Algorithms**

    Study and implement core nature-inspired algorithms such as:

    - Particle Swarm Optimization (PSO)

- Ant Colony Optimization (ACO)
- Bee Colony Optimization
- Firefly Algorithm
- Bat Algorithm

11. **Mathematical Modeling of Optimization Problems**

    Formulate real-world problems as objective functions and apply nature-inspired algorithms to find near-optimal solutions.

12. **Performance Evaluation and Convergence Analysis**

    Analyze the performance, convergence behavior, and parameter tuning of different algorithms across benchmark functions.

13. **Hybrid and Multi-objective Optimization**

    Explore hybrid approaches (e.g., PSO-GA) and techniques for solving multi-objective optimization problems using Pareto fronts.

14. **Hands-On Simulation and Coding**

    Implement swarm algorithms using Python/Matlab and apply them to optimization problems in scheduling, routing, or clustering.

15. **Comparative Analysis**

    Compare nature-inspired methods with classical techniques like gradient descent, linear programming, and genetic algorithms.

16. **Ethical and Theoretical Understanding**

    Discuss the ethical implications of autonomous decision-making and the theoretical limitations of metaheuristic algorithms.

Outside the Classroom

9. **Capstone Projects on Real-World Optimization**

   Design and implement solutions using swarm intelligence for applications like:

   - Smart traffic routing
   - Task scheduling in cloud computing
   - Path planning for robotics
   - Energy management in IoT systems

10. **Participation in Research and Innovation**

    Read, summarize, and contribute to ongoing research in bio-inspired computing, optimization journals, or IEEE conferences.

11. **Competitions and Hackathons**

    Compete in global coding contests and AI/optimization challenges on platforms like Kaggle, CodaLab, or IEEE Xplore datasets.

12. **Collaborative Interdisciplinary Projects**

    Apply swarm techniques in fields like environmental modeling, bioinformatics, logistics, or economics.

13. **Self-Learning through Simulation Tools**

    Explore frameworks and tools such as DEAP (Python), MATLAB toolboxes, or NetLogo for simulating agent-based systems.

14. **Open Source Contributions and Algorithm Development**

    Develop, document, and publish custom variants of nature-inspired algorithms on GitHub for community use.

15. **Ethics and Responsible Use**

    Reflect on responsible algorithm design especially in autonomous systems, swarm robotics, and evolutionary modeling.

16. **Presentation and Documentation Skills**

    Create academic posters, technical blogs, or video demonstrations to communicate project outcomes and innovations.

Swarm Intelligence and Nature-Inspired Optimization

Course Name: Swarm Intelligence and Nature-Inspired Optimization	Course Code	L-T-P	Credits
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Algorithms, optimization techniques, linear algebra, probability.		

Course Perspective:

This course introduces swarm intelligence and nature-inspired metaheuristics like Ant Colony Optimization, Particle Swarm Optimization, Genetic Algorithms, and Artificial Bee Colony. It emphasizes their application in real-world optimization problems across engineering, logistics, and machine learning.

The Course Outcomes (COs).

COs	Statements
CO 1	Understanding and simulating swarm-based and evolutionary optimization algorithms.
CO 2	Applying population-based methods to solve real-world constrained and unconstrained optimization problems.
CO 3	Comparing the effectiveness of nature-inspired approaches against classical optimization methods.

CO 4	Designing hybrid or domain-adapted algorithms for multi-objective optimization problems.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Optimization and Metaheuristics Overview**	**No. of hours: 15**
Topics Covered: • Optimization problem types: linear, nonlinear, discrete, combinatorial • Deterministic vs stochastic methods • Exploration vs exploitation • **Real-World Use Case:** Parameter tuning for machine learning algorithms		
Unit Number: 2	**Evolutionary Algorithms**	**No. of hours: 15**
Topics Covered: • Genetic algorithms: crossover, mutation, fitness selection • Selection strategies: roulette wheel, tournament • Elitism and convergence analysis • **Real-World Use Case:** Scheduling staff for multiple shifts with constraints		
Unit Number: 3	**Swarm Intelligence Algorithms**	**No. of hours: 15**
Topics Covered: • Ant Colony Optimization for shortest path problems • Particle Swarm Optimization for continuous domains • Artificial Bee Colony algorithm • **Real-World Use Case:** Optimal routing of delivery vehicles using PSO		

Unit Number: 4	Hybrid and Multi-objective Optimization	No. of hours: 15
Topics Covered: • NSGA-II for multi-objective problems • Hybridization strategies: combining GAs and PSO • Applications in feature selection, IoT, robotics, bioinformatics • **Real-World Use Case:** Energy optimization in sensor networks using ABC + PSO hybrid		

Text and Reference Books:

- Kennedy & Eberhart – *Swarm Intelligence*
- Kalyanmoy Deb – *Multi-objective Optimization using Evolutionary Algorithms*
- Marco Dorigo – *Ant Colony Optimization*
- Xin-She Yang – *Nature-Inspired Metaheuristic Algorithms*
- Goldberg – *Genetic Algorithms in Search, Optimization and Machine Learning*

Learning Outcomes

Inside the Classroom

17. **Foundations of Swarm Intelligence**

    Understand the biological and behavioral basis of swarm systems including ant colonies, bird flocking, and fish schooling.

18. **Exploration of Key Algorithms**

    Study and implement core nature-inspired algorithms such as:

    - Particle Swarm Optimization (PSO)

- Ant Colony Optimization (ACO)
- Bee Colony Optimization
- Firefly Algorithm
- Bat Algorithm

19. **Mathematical Modeling of Optimization Problems**

    Formulate real-world problems as objective functions and apply nature-inspired algorithms to find near-optimal solutions.

20. **Performance Evaluation and Convergence Analysis**

    Analyze the performance, convergence behavior, and parameter tuning of different algorithms across benchmark functions.

21. **Hybrid and Multi-objective Optimization**

    Explore hybrid approaches (e.g., PSO-GA) and techniques for solving multi-objective optimization problems using Pareto fronts.

22. **Hands-On Simulation and Coding**

    Implement swarm algorithms using Python/Matlab and apply them to optimization problems in scheduling, routing, or clustering.

23. **Comparative Analysis**

    Compare nature-inspired methods with classical techniques like gradient descent, linear programming, and genetic algorithms.

24. **Ethical and Theoretical Understanding**

    Discuss the ethical implications of autonomous decision-making and the theoretical limitations of metaheuristic algorithms.

Outside the Classroom

17. **Capstone Projects on Real-World Optimization**

    Design and implement solutions using swarm intelligence for applications like:

    - Smart traffic routing
    - Task scheduling in cloud computing
    - Path planning for robotics
    - Energy management in IoT systems

18. **Participation in Research and Innovation**

    Read, summarize, and contribute to ongoing research in bio-inspired computing, optimization journals, or IEEE conferences.

19. **Competitions and Hackathons**

    Compete in global coding contests and AI/optimization challenges on platforms like Kaggle, CodaLab, or IEEE Xplore datasets.

20. **Collaborative Interdisciplinary Projects**

    Apply swarm techniques in fields like environmental modeling, bioinformatics, logistics, or economics.

21. **Self-Learning through Simulation Tools**

    Explore frameworks and tools such as DEAP (Python), MATLAB toolboxes, or NetLogo for simulating agent-based systems.

22. **Open Source Contributions and Algorithm Development**

    Develop, document, and publish custom variants of nature-inspired algorithms on GitHub for community use.

23. **Ethics and Responsible Use**

    Reflect on responsible algorithm design especially in autonomous systems, swarm robotics, and evolutionary modeling.

24. **Presentation and Documentation Skills**

    Create academic posters, technical blogs, or video demonstrations to communicate project outcomes and innovations.

Swarm Intelligence and Nature-Inspired Optimization

Course Name: Swarm Intelligence and Nature-Inspired Optimization	Course Code	L-T-P	Credits
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Algorithms, optimization techniques, linear algebra, probability.		

Course Perspective:

This course introduces swarm intelligence and nature-inspired metaheuristics like Ant Colony Optimization, Particle Swarm Optimization, Genetic Algorithms, and Artificial Bee Colony. It emphasizes their application in real-world optimization problems across engineering, logistics, and machine learning.

The Course Outcomes (COs).

COs	Statements
CO 1	Understanding and simulating swarm-based and evolutionary optimization algorithms.
CO 2	Applying population-based methods to solve real-world constrained and unconstrained optimization problems.
CO 3	Comparing the effectiveness of nature-inspired approaches against classical optimization methods.

CO 4	Designing hybrid or domain-adapted algorithms for multi-objective optimization problems.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Optimization and Metaheuristics Overview**	**No. of hours: 15**
Topics Covered: • Optimization problem types: linear, nonlinear, discrete, combinatorial • Deterministic vs stochastic methods • Exploration vs exploitation • **Real-World Use Case:** Parameter tuning for machine learning algorithms		
Unit Number: 2	**Evolutionary Algorithms**	**No. of hours: 15**
Topics Covered: • Genetic algorithms: crossover, mutation, fitness selection • Selection strategies: roulette wheel, tournament • Elitism and convergence analysis • **Real-World Use Case:** Scheduling staff for multiple shifts with constraints		
Unit Number: 3	**Swarm Intelligence Algorithms**	**No. of hours: 15**
Topics Covered: • Ant Colony Optimization for shortest path problems • Particle Swarm Optimization for continuous domains • Artificial Bee Colony algorithm • **Real-World Use Case:** Optimal routing of delivery vehicles using PSO		

Unit Number: 4	Hybrid and Multi-objective Optimization	No. of hours: 15
Topics Covered: • NSGA-II for multi-objective problems • Hybridization strategies: combining GAs and PSO • Applications in feature selection, IoT, robotics, bioinformatics • **Real-World Use Case:** Energy optimization in sensor networks using ABC + PSO hybrid		

Text and Reference Books:

- Kennedy & Eberhart – *Swarm Intelligence*
- Kalyanmoy Deb – *Multi-objective Optimization using Evolutionary Algorithms*
- Marco Dorigo – *Ant Colony Optimization*
- Xin-She Yang – *Nature-Inspired Metaheuristic Algorithms*
- Goldberg – *Genetic Algorithms in Search, Optimization and Machine Learning*

Learning Outcomes

Inside the Classroom

25. **Foundations of Swarm Intelligence**

    Understand the biological and behavioral basis of swarm systems including ant colonies, bird flocking, and fish schooling.

26. **Exploration of Key Algorithms**

    Study and implement core nature-inspired algorithms such as:

    - Particle Swarm Optimization (PSO)

- Ant Colony Optimization (ACO)
- Bee Colony Optimization
- Firefly Algorithm
- Bat Algorithm

27. **Mathematical Modeling of Optimization Problems**

    Formulate real-world problems as objective functions and apply nature-inspired algorithms to find near-optimal solutions.

28. **Performance Evaluation and Convergence Analysis**

    Analyze the performance, convergence behavior, and parameter tuning of different algorithms across benchmark functions.

29. **Hybrid and Multi-objective Optimization**

    Explore hybrid approaches (e.g., PSO-GA) and techniques for solving multi-objective optimization problems using Pareto fronts.

30. **Hands-On Simulation and Coding**

    Implement swarm algorithms using Python/Matlab and apply them to optimization problems in scheduling, routing, or clustering.

31. **Comparative Analysis**

    Compare nature-inspired methods with classical techniques like gradient descent, linear programming, and genetic algorithms.

32. **Ethical and Theoretical Understanding**

Discuss the ethical implications of autonomous decision-making and the theoretical limitations of metaheuristic algorithms.

Outside the Classroom

25. **Capstone Projects on Real-World Optimization**

Design and implement solutions using swarm intelligence for applications like:

- Smart traffic routing
- Task scheduling in cloud computing
- Path planning for robotics
- Energy management in IoT systems

26. **Participation in Research and Innovation**

Read, summarize, and contribute to ongoing research in bio-inspired computing, optimization journals, or IEEE conferences.

27. **Competitions and Hackathons**

Compete in global coding contests and AI/optimization challenges on platforms like Kaggle, CodaLab, or IEEE Xplore datasets.

28. **Collaborative Interdisciplinary Projects**

Apply swarm techniques in fields like environmental modeling, bioinformatics, logistics, or economics.

29. **Self-Learning through Simulation Tools**

    Explore frameworks and tools such as DEAP (Python), MATLAB toolboxes, or NetLogo for simulating agent-based systems.

30. **Open Source Contributions and Algorithm Development**

    Develop, document, and publish custom variants of nature-inspired algorithms on GitHub for community use.

31. **Ethics and Responsible Use**

    Reflect on responsible algorithm design especially in autonomous systems, swarm robotics, and evolutionary modeling.

32. **Presentation and Documentation Skills**

    Create academic posters, technical blogs, or video demonstrations to communicate project outcomes and innovations.

Swarm Intelligence and Nature-Inspired Optimization

Course Name: Swarm Intelligence and Nature-Inspired Optimization	Course Code	L-T-P	Credits
	DSE	4-0-0	4
Type of Course:	Discipline-Specific Elective (DSE)		
Pre-requisite(s):	Algorithms, optimization techniques, linear algebra, probability.		

Course Perspective:

This course introduces swarm intelligence and nature-inspired metaheuristics like Ant Colony Optimization, Particle Swarm Optimization, Genetic Algorithms, and Artificial Bee Colony. It emphasizes their application in real-world optimization problems across engineering, logistics, and machine learning.

The Course Outcomes (COs).

COs	Statements
CO 1	Understanding and simulating swarm-based and evolutionary optimization algorithms.
CO 2	Applying population-based methods to solve real-world constrained and unconstrained optimization problems.
CO 3	Comparing the effectiveness of nature-inspired approaches against classical optimization methods.

CO 4	Designing hybrid or domain-adapted algorithms for multi-objective optimization problems.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Optimization and Metaheuristics Overview**	**No. of hours: 15**
Topics Covered: • Optimization problem types: linear, nonlinear, discrete, combinatorial • Deterministic vs stochastic methods • Exploration vs exploitation • **Real-World Use Case:** Parameter tuning for machine learning algorithms		
Unit Number: 2	**Evolutionary Algorithms**	**No. of hours: 15**
Topics Covered: • Genetic algorithms: crossover, mutation, fitness selection • Selection strategies: roulette wheel, tournament • Elitism and convergence analysis • **Real-World Use Case:** Scheduling staff for multiple shifts with constraints		
Unit Number: 3	**Swarm Intelligence Algorithms**	**No. of hours: 15**
Topics Covered: • Ant Colony Optimization for shortest path problems • Particle Swarm Optimization for continuous domains • Artificial Bee Colony algorithm • **Real-World Use Case:** Optimal routing of delivery vehicles using PSO		

Unit Number: 4	Hybrid and Multi-objective Optimization	No. of hours: 15
Topics Covered: • NSGA-II for multi-objective problems • Hybridization strategies: combining GAs and PSO • Applications in feature selection, IoT, robotics, bioinformatics • **Real-World Use Case:** Energy optimization in sensor networks using ABC + PSO hybrid		

Text and Reference Books:

- Kennedy & Eberhart – *Swarm Intelligence*
- Kalyanmoy Deb – *Multi-objective Optimization using Evolutionary Algorithms*
- Marco Dorigo – *Ant Colony Optimization*
- Xin-She Yang – *Nature-Inspired Metaheuristic Algorithms*
- Goldberg – *Genetic Algorithms in Search, Optimization and Machine Learning*

Learning Outcomes

Inside the Classroom

33. **Foundations of Swarm Intelligence**

    Understand the biological and behavioral basis of swarm systems including ant colonies, bird flocking, and fish schooling.

34. **Exploration of Key Algorithms**

    Study and implement core nature-inspired algorithms such as:

    - Particle Swarm Optimization (PSO)

- Ant Colony Optimization (ACO)
- Bee Colony Optimization
- Firefly Algorithm
- Bat Algorithm

35. **Mathematical Modeling of Optimization Problems**

    Formulate real-world problems as objective functions and apply nature-inspired algorithms to find near-optimal solutions.

36. **Performance Evaluation and Convergence Analysis**

    Analyze the performance, convergence behavior, and parameter tuning of different algorithms across benchmark functions.

37. **Hybrid and Multi-objective Optimization**

    Explore hybrid approaches (e.g., PSO-GA) and techniques for solving multi-objective optimization problems using Pareto fronts.

38. **Hands-On Simulation and Coding**

    Implement swarm algorithms using Python/Matlab and apply them to optimization problems in scheduling, routing, or clustering.

39. **Comparative Analysis**

    Compare nature-inspired methods with classical techniques like gradient descent, linear programming, and genetic algorithms.

40. **Ethical and Theoretical Understanding**

    Discuss the ethical implications of autonomous decision-making and the theoretical limitations of metaheuristic algorithms.

Outside the Classroom

33. **Capstone Projects on Real-World Optimization**

    Design and implement solutions using swarm intelligence for applications like:

    - Smart traffic routing
    - Task scheduling in cloud computing
    - Path planning for robotics
    - Energy management in IoT systems

34. **Participation in Research and Innovation**

    Read, summarize, and contribute to ongoing research in bio-inspired computing, optimization journals, or IEEE conferences.

35. **Competitions and Hackathons**

    Compete in global coding contests and AI/optimization challenges on platforms like Kaggle, CodaLab, or IEEE Xplore datasets.

36. **Collaborative Interdisciplinary Projects**

    Apply swarm techniques in fields like environmental modeling, bioinformatics, logistics, or economics.

37. **Self-Learning through Simulation Tools**

    Explore frameworks and tools such as DEAP (Python), MATLAB toolboxes, or NetLogo for simulating agent-based systems.

38. **Open Source Contributions and Algorithm Development**

    Develop, document, and publish custom variants of nature-inspired algorithms on GitHub for community use.

39. **Ethics and Responsible Use**

    Reflect on responsible algorithm design especially in autonomous systems, swarm robotics, and evolutionary modeling.

40. **Presentation and Documentation Skills**

    Create academic posters, technical blogs, or video demonstrations to communicate project outcomes and innovations.

SUMMER INTERNSHIP

Course Name: Summer Internship	**Course Code**	**L-T-P**	**Credits**
	ETCCIN301	**0-0-0**	**2**
Type of Course:	**Internship (INT)**		

The Summer Internship Program (1st June – 31st July) is designed to integrate academic learning with real-world professional experiences, enabling students to apply theoretical knowledge to practical situations. It forms a mandatory part of the Semester III for students currently in Semester II, carrying a weightage of 2 academic credits.

The key objectives of the Summer Internship Program are:

- To enhance professional skills and industry readiness.
- To expose students to real-world technical, managerial, and research practices.
- To promote self-learning, professional responsibility, and critical thinking.
- To foster connections between academic knowledge and industry practices.

Duration

The duration of the internship will be 6-8 weeks. It will take place after the completion of the 2nd semester and before the commencement of the 3rd semester.

Internship Options

Students can choose from the following options:

1. **Industry Internship (Online/Offline):**

Students must produce a joining letter at the start and a relieving letter upon completion.

2. **Global Certifications:**

Students can opt for globally recognized certification programs relevant to their field of study.

3. **Government/Research Institution Internship:**

Students can engage in a research internship with premier government or research organizations such as IITs, IISc, ISRO, DRDO, CSIR, NPL, etc.

4. **On-Campus Industry Internship Programs:**

The university will offer on-campus internships in collaboration with industry partners.

Deliverables and Documentation:

Each student must submit the following after completing their internship/certification:

Deliverable	**Description**	**Marks**
Summer Internship File	A detailed report/file based on the provided format including objectives, methodology, learnings, and reflections.	10 Marks
Video Presentation	A 7–10-minute recorded video presentation showcasing work done during the internship/certification. The template of slides will be shared.	20 Marks
Certificate of Completion	A color-printed certificate on bond paper from the host organization/certification body, mentioning duration, role/project.	70 Marks

Evaluation Metrics

The Summer Internship will be evaluated based on the following comprehensive criteria:

Evaluation Component	**Weightage**	**Description**
Internship Report/File	10%	Completeness, professional formatting, relevance to internship tasks.
Video Presentation	20%	Content quality, clarity, communication skills, professional presentation.
Certificate of Completion	70%	Authenticity, completion of internship/certification within stipulated time, relevance to program objectives.

Internship Evaluation Rubric:

S. No.	Component	Sub-Component / Criteria	Marks
1	Internship Certificate	Relevance to Core Subjects	20 Marks
		- Directly relates to core subjects	20
		- Partially relates to core subjects	15
		- Minimally relates to core subjects	10
		- Not relevant	0
2	Report Submission	Structure and Organization	10 Marks
		- Well-structured and organized report	10
		- Moderately structured report	7
		- Poorly structured report	3
		- No structure	0
3	Solo Video-Based Evaluation	a. Technical / Professional / Soft Skills Acquired	10 Marks
		- Highly relevant and advanced technical skills	10
		- Moderately relevant technical skills	8
		- Basic technical skills	5
		- No new skills acquired	0
		b. Content Delivery	10 Marks
		- Clear, engaging, and thorough delivery	10
		- Clear but less engaging delivery	7
		- Somewhat clear and engaging delivery	3
		- Unclear and disengaging delivery	0
		c. Visual Aids & Communication Skills	10 Marks
		- Effective visual aids + excellent communication skills	10

		- Moderate visual aids + good communication skills	7
		- Basic visual aids + fair communication skills	3
		- No visual aids + poor communication skills	0
4	Internship Duration	Weeks Completed	10 Marks
		- 6–8 weeks completed	10
		- 4–6 weeks completed	8
		- Less than 1 month	5
5	Outcome of the Internship	Application / Project / Key Learnings & Findings	30 Marks
		- Clear, outcome-based project with applied learnings and key findings	25–30
		- Moderate outcome with partial application and findings	15–24
		- Minimal outcome, unclear learning/application	0–14

Course Outcomes:

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

- Applying Theoretical Knowledge:

  - Integrating and applying theoretical knowledge gained during coursework to real- world industry or research problems.

- Developing Technical Skills:

  - Acquiring and demonstrating advanced technical skills relevant to the field of computer science and engineering through practical experience.

- Conducting Independent Research:

- Executing independent research projects, including problem identification, literature review, methodology design, data collection, and analysis.

- Preparing Professional Reports:

  - Compiling comprehensive and well-structured reports that document the intern- ship experience, project details, research findings, and conclusions.

- Enhancing Problem-Solving Abilities:

  - Developing enhanced problem-solving and critical thinking skills by tackling practical challenges encountered during the internship.

Improve Professional and Soft Skills:

- Exhibit improved professional and soft skills, including communication, team- work, time management, and adaptability in a professional setting.

- Present Findings Effectively:

  - Deliver clear and engaging presentations to effectively communicate project outcomes, research findings, and acquire knowledge to peers and faculty members.

- Pursue Lifelong Learning:

  - Demonstrate a commitment to lifelong learning by engaging in continuous skill development and staying updated with emerging trends and technologies in the field.

MOOC IN THE RELEVANT DOMAIN OF SPECIALIZATION

Course Name: MOOC in the Relevant Domain of Specialization	**Course Code**	**L-T-P**	**Credits**
	MOOC	2-0-0	2
Type of Course:	Online Learning-Based Course		

Course Perspective:

This course allows students to gain knowledge in their domain of specialization through online platforms such as Swayam, NPTEL, or AICTE's ELIS. It encourages self-paced learning, industry-relevant skill development, and certification from reputed institutions.

The Course Outcomes (COs).

COs	Statements
CO 1	Enhance subject knowledge through certified online courses.
CO 2	Develop skills in the chosen domain of specialization.
CO 3	Apply self-learning strategies for continuous knowledge enhancement.
CO 4	Successfully complete and obtain certification from recognized MOOC platforms.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Guidelines:

- Students must select a MOOC course relevant to their field of specialization.
- The course should be taken from recognized platforms such as Swayam, NPTEL, AICTE's ELIS, Coursera, Udemy, or edX (as per institutional guidelines).
- Prior approval from faculty is required before enrolling in the course.
- Completion of assignments, quizzes, and final assessment is mandatory.
- A completion certificate from the MOOC provider must be submitted for evaluation.

COMPETITIVE CODING -III

Course Name:	Course Code	L-T-P	Credits
Competitive Coding-III	SEC	2-0-0	2
Type of Course:	Skill Enhancement Course (SEC)		
Prerequisite(s), if any: Competitive Coding-II, Fundamentals of Data Structures and Algorithms			
Course Outcome: • Applying bit manipulation, number theory, and string algorithms to solve computational problems. • Analyzing and implement advanced backtracking and recursion techniques for combinatorial			

problems.

- Evaluating sliding window techniques and two-pointer algorithms for efficient solutions.
- Solving graph problems using foundational and advanced concepts in competitive programming.

SESSION WISE DETAILS

Session 1	Bit Manipulation Introduction.	No. of hours: 2
Content Summary: Introduction to AND, OR, XOR operations, Count Set/unset Bits, Toggle a given kth bit, check if nth bit is set or unset, Check Power of Two/Four.		
Session: 2	Bit Manipulation-II.	No. of hours: 2
Content Summary: Counting Single Number 1, Single number 2, Subsets using Bits (power set problem) , Find Missing number, Duplicate Numbers.		
Session: 3	Number theory basics.	No. of hours: 2
Content Summary: Sieve of Eratosthenes, Modular Arithmetic, Modular Exponentiation, Chinese Remainder Theorem		
Session: 4	Mathematical Algorithms.	No. of hours: 2
Content Summary: Euler's Totient Function, Permutations and Combinations, Inclusion-Exclusion Principle, Catalan Numbers.		
Session 5	Advance Recursion.	No. of hours: 2
Content Summary: print all subset, permutation of a string, find all unique subset		
Session: 6	Backtracking I	No. of hours: 2
Content Summary: rat in maze, rat in a maze all path, N Queens		
Session: 7	Backtracking-2	No. of hours: 2

Content Summary: combination, combination sum, combination sum-2		
Session: 8	Backtracking-3	No. of hours: 2
Content Summary: generate parentheses, subset-2, sudoku solver		
Session : 9	Greedy I	No. of hours: 2
Content Summary: assign cookies, array partition, can place flower, lemonade change		
Session: 10	Greedy-II.	No. of hours: 2
Content Summary: Activity selection, minimum platform, coin change		
Session : 11	Greedy-III.	No. of hours: 2
Content Summary: max chunk to make sorted, max chunk to make sorted-2, 0/1 knapsack.		
Session: 12	Graph Introduction and representation.	No. of hours: 2
Content Summary: Introduction, Representation using adjacency matrix and adjacency list		
Session: 13	Graph-Traversal Algorithm.	No. of hours: 2
Content Summary: Graph Traversal BFS(Breadth first search) and DFS(Depth first search)		
Session: 14	Graph-III	No. of hours: 2
Content Summary : Connected Components, Detecting Cycles in Graphs		
Session: 15	Graph Problems-IV.	No. of hours: 2
Content summary: find if path exist(has path), print all path from source to destination, Number of Island		
Session: 16	Advanced Graph.	No. of hours: 2
Content summary: Number of Provinces, Flood Fill, Number of closed islands.		
Session: 17	Minimum Spanning Tree algorithms.	No. of hours: 2

Content summary: Prim’s Algorithm, Kruskal's algorithm.		
Session: 18	Shortest Path Algorithm.	No. of hours: 2
Content summary: Dijkstra algorithm, Bellman ford algorithm.		
Session: 19	Summarizing the Semester 5.	No. of hours: 2
Content summary: Company specific problems on Graphs, sliding window and recursion.		
Session: 20	Summarizing the Semester 5.	No. of hours: 2
Content summary: Company specific problems on Graphs, sliding window and recursion.		
Reference Books: • *Elements of Programming Interviews (EPI)* – Adnan Aziz, Tsung-Hsien Lee, Amit Prakash Great for interview-style problems with solutions in C++, Java, and Python editions. • *Cracking the Coding Interview* – Gayle Laakmann McDowell		

MAJOR PROJECT-I

Course Name: Major Project-I	**Course Code**	**L-T-P**	**Credits**
	ETCCPR302	0-0-0	4
Type of Course:	Project-Based Course (Proj)		
Pre-requisite(s), if any:	Completion of core courses in the respective domain.		

Course Perspective:

The Major Project provides students with an opportunity to work on real-world problems in an industrial or research setting. The course is designed to enhance problem-solving, technical expertise, and innovation by engaging students in significant industry-aligned or research-driven projects.

The Course Outcomes (COs).

COs	Statements
CO 1	Identify and analyze real-world challenges in industry or research.
CO 2	Design and implement solutions using advanced technical concepts.
CO 3	Develop critical thinking, problem-solving, and teamwork skills.
CO 4	Prepare comprehensive project reports and present findings effectively.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Project Guidelines:

- Students must select a project in collaboration with an industry partner or research organization.
- The project should be approved by the faculty mentor before commencement.
- Regular progress meetings with the mentor are required.
- A detailed project report and final presentation must be submitted.
- The project should demonstrate originality, innovation, and practical application.

Evaluation Criteria:

- Project Proposal & Approval: 10%
- Mid-Term Progress Evaluation: 30%
- Final Report & Presentation: 40%
- Industry/Research Mentor Feedback: 20%

SEMESTER-IV

INDUSTRY/RESEARCH INTERNSHIP (FULL SEMESTER)

Course Name: Industry/Research Internship (Full Semester)	**Course Code**	**L-T-P**	**Credits**
	ETCCIN471	0-0-0	8
Type of Course:	Internship-Based Course (INT)		
Pre-requisite(s), if any:	Completion of core courses in the respective domain.		

Course Perspective:

This full-semester internship provides students with hands-on industry or research experience in their field of specialization. It helps them bridge the gap between theoretical knowledge and real-world applications, enhancing their professional skills, problem-solving abilities, and research acumen.

The Course Outcomes (COs).

COs	Statements
CO 1	Gain practical experience in industry or research environments.
CO 2	Apply theoretical knowledge to solve real-world problems.
CO 3	Develop professional, communication, and teamwork skills.
CO 4	Prepare technical reports and presentations based on internship work.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Internship Guidelines:

- Students must secure an internship with an approved industry or research organization.
- The internship should be aligned with the student's domain of study.
- A faculty mentor will be assigned to monitor progress.
- Regular progress reports must be submitted as per guidelines.
- At the end of the internship, students must submit a final report and give a presentation.

Evaluation Criteria:

- Internship Proposal & Approval: 10%
- Mid-Term Progress Report: 30%
- Final Internship Report & Presentation: 40%
- Industry/Research Supervisor Feedback: 20%

MAJOR PROJECT-II

Course Name: Major Project-II	**Course Code**	**L-T-P**	**Credits**
	ETCCPR472	0-0-0	4
Type of Course:	Project-Based Course (Proj)		
Pre-requisite(s), if any:	Completion of core courses in the respective domain.		

Course Perspective:

The Major Project provides students with an opportunity to work on real-world problems in an industrial or research setting. The course is designed to enhance problem-solving, technical expertise, and innovation by engaging students in significant industry-aligned or research-driven projects.

The Course Outcomes (COs).

COs	Statements
CO 1	Identifying and analyzing real-world challenges in industry or research.
CO 2	Designing and implementing solutions using advanced technical concepts.
CO 3	Developing critical thinking, problem-solving, and teamwork skills.
CO 4	Preparing comprehensive project reports and present findings effectively.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Project Guidelines:

- Students must select a project in collaboration with an industry partner or research organization.
- The project should be approved by the faculty mentor before commencement.
- Regular progress meetings with the mentor are required.
- A detailed project report and final presentation must be submitted.
- The project should demonstrate originality, innovation, and practical application.

Evaluation Criteria:

- Project Proposal & Approval: 10%
- Mid-Term Progress Evaluation: 30%
- Final Report & Presentation: 40%
- Industry/Research Mentor Feedback: 20%

MOOC IN THE RELEVANT DOMAIN OF SPECIALIZATION

Course Name: MOOC in the Relevant Domain of Specialization	**Course Code**	**L-T-P**	**Credits**
	MOOC	2-0-0	2
Type of Course:	Online Learning-Based Course		
Pre-requisite(s), if any:	As per the selected MOOC course requirements.		

Course Perspective:

This course enables students to pursue online learning in specialized domains through platforms such as Swayam, NPTEL, or AICTE's ELIS. The course provides flexibility for self-paced learning while enhancing technical and professional skills in emerging areas.

The Course Outcomes (COs).

COs	Statements
CO 1	Identify and enroll in a MOOC course relevant to their specialization.
CO 2	Acquire in-depth knowledge of advanced topics through online learning.
CO 3	Develop self-learning and time-management skills.
CO 4	Demonstrate acquired knowledge through assessments, projects, or certification.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Guidelines:

- Students must select a MOOC course from approved platforms (Swayam/NPTEL/AICTE ELIS).
- The course should align with the student's academic and professional goals.
- Regular progress tracking and completion of assignments/quizzes are required.
- Upon completion, students must submit proof of certification and a brief report summarizing key learnings.

Evaluation Criteria:

- MOOC Course Selection & Approval: 10%
- Regular Progress Tracking: 30%
- Final Certification: 40%
- Summary Report & Presentation: 20%