K.R. MANGALAM UNIVERSITY
THE COMPLETE WORLD OF EDUCATION

School of Engineering and Technology

Programme Handbook

(Programme Study & Evaluation Scheme)

Bachelor's in science
(B. Sc. (Hons.) Computer Science)

Programme Code: 72

THREE YEAR UNDERGRADUATE PROGRAMME

WITH EFFECTIVE FROM 2025-26 SESSION

Approved in the 38th Meeting of Academic Council Held on 28 June 2025

Contents

3. Preface

Welcome to the School of Engineering and Technology at K. R. Mangalam University. It is with great enthusiasm that we introduce you to an institution dedicated to nurturing future leaders in engineering and technology.
Established in 2013, our School has rapidly evolved into a premier center for innovation, quality education, and skill development. With a focus on imparting advanced knowledge and fostering creativity, we are committed to providing a transformative educational experience. Our state-of-the-art infrastructure, cutting-edge laboratories, and a distinguished team of faculty members collectively create an environment where academic and professional excellence thrives.
Our diverse programs encompass undergraduate degrees (B. Tech, BCA, B. Sc), post- graduate 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. This curriculum emphasizes problem-solving, interdisciplinary learning, and innovative teaching methodologies.
Our emphasis on industry integration is reflected in our collaborations with renowned organizations such as IBM, Samatrix, Xebia, E.C Council, and ImaginXP. These partnerships ensure that our students gain practical experience and insights that are directly applicable to industry demands. Elective options across diverse domains, including AI, Cloud Computing, Cyber Security, and Full Stack Development, offer students the flexibility to tailor their educational experience to their career aspirations.
We are also dedicated to fostering a culture of innovation and entrepreneurship through our Entrepreneurship and Incubation Center and initiatives like ’MindBenders,’ ’Hack- KRMU,’ and participation in the ’Smart India Hackathon.’ These programs are designed to inspire and prepare students to become forward-thinking leaders in the technology sector.
Our modern computing facilities and comprehensive infrastructure support advanced research, simulations, and hands-on projects, ensuring that our students are well-prepared for the challenges of the professional world. K. R. Mangalam University is recognized for its commitment to providing quality education, and our alumni have made notable contributions across various sectors, from multinational corporations to public sector enterprises.
We are excited to accompany you on this journey and look forward to supporting your academic and professional growth. Welcome to a community where excellence and innovation are at the core of everything we do.

School of Engineering and Technology
K. R. Mangalam University.

NEP-2020

K.R. Mangalam University has adopted the National Education Policy (NEP-2020) to establish a holistic and multidisciplinary undergraduate education environment, aiming to equip our students for the demands of the 21st century. Following the guidelines of NEP- 2020 regarding the curriculum structure and duration of the undergraduate programme, we now offer a **Four-Year Undergraduate Programme** with multiple entry and exit points, along with re-entry options, and relevant certifications.

UG Certificate after completing 1 year (2 semesters with the required number of credits) of study, and an additional vocational course/internship of 4 credits during the summer vacation of the first year.

UG Diploma after completing 2 years (4 semesters with the required number of credits) of study, and an additional vocational course/internship of 4 credits during the summer vacation of the second year.

Bachelor's Degree after completing a 3-year (6 semesters with the required num- ber of credits) programme of study.

4-year Bachelor's Degree (Honours) with the required number of credits after an eight-semester programme of study.

Students who secure 75% marks and above in the first six semesters and wish to undertake research at the undergraduate level can choose a research stream in the fourth year. Upon completing a research project in their major area(s) of study in the 4th year, a student will be awarded a **Bachelor's Degree (Honours with Research)**.

Advantages of pursuing the 4-year Bachelor's degree programme with Hon- ours/Honours with Research include the possibility of completing a Master's degree in one year. Additionally, a 4-year degree programme will facilitate admission to foreign universities.

4.1 Categories of Courses

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

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

Multidisciplinary (Open Elective): These courses are intended to broaden 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 categories, is offered to the students. These courses enable students to expand their perspectives and gain a holistic understanding of various disciplines. Students can choose courses based on their areas of interest.

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

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

Value-Added Course (VAC): The Value-Added Courses (VAC) are aimed at inculcating Humanistic, Ethical, Constitutional, and Universal human values of truth, righteous conduct, peace, love, non-violence, scientific and technological advancements, global citizenship values, and life-skills falling under the below given categories:

- Understanding India
- Environmental Science/Education
- Digital and Technological Solutions
- Health & Wellness, Yoga education, Sports, and Fitness

Research Project / Dissertation: Students choosing a 4-Year Bachelor's degree (Honours with Research) are required to take up research projects under the guidance of a faculty member. The students are expected to complete the Research Project in the eighth 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.

S. No.	Broad Categories of Courses	Minimum Credit Requirement
1	Major (Core)	46
2	Discipline Specific Elective	16
3	Open Elective	09
4	Ability Enhancement Course (AEC)	08
5	Skill Enhancement Course (SEC)	10
6	Value-Added Course (VAC)	06
7	Summer Internship	12
8	Research Project/Dissertation	12
9	Community service	02
10	MOOC	04
9	**Total**	**125**

University Vision & Mission

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

Mission

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

4. 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 excellent infrastructure, state of the art Labs, and a team of qualified and research-oriented faculty members.

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

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

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

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

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

We prioritize career growth by offering comprehensive training, placements, inter- national internships, and preparation for further studies. Our commitment to nurturing globally competitive professionals is reflected in the diverse pathways we pave for our students. SOET aims at transforming the students into competitive engineers with adequate analytical skills, making them more acceptable to potential employers in the country. At our school, we emphasize learning through doing. Whether it's project-based learning, field projects, research projects, internships, or engaging in competitive coding, our students actively shape their futures by applying theoretical knowledge to practical scenarios. We provide opportunities for industrial projects, R&D projects, and start-up projects in the final year, ensuring that our students engage in real-world innovation.

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

We pride ourselves on providing state-of-the-art computing facilities and infrastructure. Our modern labs and computing resources are equipped to support the diverse needs of our students, enabling them to engage in advanced research, simulations, and hands-on projects. K.R. Mangalam University has marked its presence in Delhi NCR as a value-based university, successfully imparting quality education in all domains. Our alumni are working across all sectors of technology, from MNCs to PSUs.

5. 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 mind- set development.
- **Enhancing** collaborations with National and International organizations and institutions to develop cross-cultural understanding to adapt and thrive in the 21st century.

6. About the Programme

6.1 Definitions

- **Programme Outcomes (POs):** Programme Outcomes are statements that describe what the students are expected to know and would be able to do upon graduation. These relate to the skills, knowledge, and behaviour 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 their program 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 by which the coursework is measured. It determines the number of hours of instruction required per week. One credit is equivalent to 14-15 periods for theory, or 28-30 periods for workshop/labs and tutorials.

6.2 Programme Educational Objectives (PEO)

- **PEO1:** Successful professionals in industry, government, academia, research, entrepreneurial pursuits, and consulting firms.
- **PEO2:** Able to apply their knowledge of computer science & engineering principles to solve societal problems by exhibiting a strong foundation in both theoretical and practical aspects of the field.
- **PEO3:** Dedicated to upholding professional ethics and social responsibilities, with a strong commitment to advancing sustainability goals.
- **PEO4:** Demonstrating strong leadership skills and a proven ability to collaborate effectively in diverse, multidisciplinary teams to successfully achieve project objectives.

6.3 Programme Outcomes (PO)

Graduates will be able to:

- **PO1 Core Competencies in Engineering:** Graduates will possess a strong foundation in computer science principles, critical problem analysis, and solution design, equipped with skills for conducting thorough investigations to solve complex challenges.
- **PO2 Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern IT tools including prediction and modeling to complex computer science activities with an understanding of the limitations.

- **PO3 Societal and Environmental Responsibility:** Apply contextual knowledge to evaluate societal, health, safety, legal, and cultural issues, while under- standing the impact of engineering solutions on the environment and advocating for sustainable development.
- **PO4 Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the computer science practice.
- **PO5 Effective Communication and Team Collaboration:** Excel in both individual and team roles within diverse and multidisciplinary settings, while communicating complex computer science concepts clearly through effective reports, presentations, and interactions.
- **PO6 Project Management:** Apply engineering and management principles to lead and manage projects effectively in computer science contexts.
- **PO7 Life-long Learning:** Embrace and actively pursue continuous learning to stay current with technological advancements and evolving practices in computer science.

6.4 Programme Specific Outcomes (PSO)

- **PSO1:** Understanding the core concepts, theories, tools, techniques, and method- ologies of Computer Science.
- **PSO2:** Applying Computer Science principles to solve real-world challenges.
- **PSO3:** Analysing methodologies, problems, and issues related to Computer Science.
- **PSO4:** Evaluating alternative solutions and making informed decisions to solve problems in Computer Science.
- **PSO5:** Designing and developing innovative solutions to address complex problems in Computer Science.

6.5 Career Avenues

Diverse career avenues available to graduates of the B. Sc (Hons.) Computer Science program are as follows:

1. **Software Development:** Graduates can pursue careers as software developers, working on designing, coding, testing, and maintaining software applications and systems. They can specialize in areas such as web development, mobile app development, game development, or enterprise software development.
2. **Systems Analyst:** A systems analyst analyzes an organization's computer systems and procedures to improve efficiency and effectiveness. They work on designing and implementing new systems, conducting feasibility studies, and identifying areas for improvement in existing systems.

3. **Data Scientist:** With the increasing volume of data in various industries, data scientists are in high demand. They utilize their skills in data analysis, statistics, and machine learning to extract insights from large datasets, make data-driven decisions, and develop predictive models.
4. **Artificial Intelligence Engineer:** As AI technology continues to advance, there is a growing demand for professionals skilled in developing AI algorithms and systems. AI engineers work on creating intelligent machines, developing natural language processing systems, computer vision applications, and other AI-driven solutions.
5. **Cybersecurity Analyst:** In an era of heightened cybersecurity threats, organizations require experts who can protect their systems and data. Cybersecurity analysts identify vulnerabilities, implement security measures, conduct risk assessments, and respond to security incidents to safeguard computer systems and net- works.
6. **Network Engineer:** Network engineers are responsible for designing, implementing, and maintaining computer networks within organizations. They ensure network reliability, security, and performance, and troubleshoot network issues to ensure smooth operations.
7. **IT Project Manager:** IT project managers oversee the planning, execution, and delivery of technology projects within organizations. They manage project teams, coordinate resources, track progress, and ensure projects are completed within bud- get and on time.
8. **Database Administrator:** Database administrators manage and maintain databases, ensuring data integrity, security, and performance. They design database structures, implement backup and recovery procedures, and optimize database systems for efficient data storage and retrieval.
9. **Software Quality Assurance Engineer:** QA engineers are responsible for ensuring the quality and reliability of software applications. They develop and execute test plans, identify and report bugs and issues, and work closely with development teams to improve software quality.
10. **Research and Development:** Graduates can pursue careers in research and development, work on innovative projects, exploring new technologies, and pushing the boundaries of computer science. This can involve academic research, industry research labs, or research and development departments within companies.

6.6 Duration

3 Years (6 Semesters) - Full-Time Program

6.7 Eligibility Criteria

The candidate should have passed 10+2 or its equivalent examination from a recognized Board with a minimum of 50% marks in aggregate. The reservation and relaxation for SC/ST/OBC/PWD and other categories shall be as per the rules of central/state government, whichever is applicable.

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

7.1 University Education Objective

- **Focus Employability and Entrepreneurship through Holistic Education**

7.2 Importance of Structured Learning Experiences

- **Holistic and Structured Academic Journey:** The curriculum focuses on employability, entrepreneurship, and personal development through a balanced education that integrates major/minor selection, internships, projects, coding, and hands-on industry experience.
- **Comprehensive Learning and Support:** Students benefit from experiential learning through labs, workshops, and guest lectures, while academic support, mentor-mentee programs, and counselling services cater to the needs of both slow and advanced learners.
- **Continuous Evaluation and Growth:** A robust assessment framework with regular feedback, emphasis on academic integrity, and structured evaluation methods ensures ongoing student development and success.

7.3 Educational Planning and Execution

Effective educational planning and execution is a cornerstone of delivering a high-quality academic experience. At the School of Engineering & Technology, we focus on a structured and dynamic approach to ensure that students gain the knowledge, skills, and competencies required to excel in the rapidly evolving technological landscape. Our planning and execution encompass the following key aspects:

- **Curriculum Design:** The curriculum is meticulously crafted. It integrates theoretical learning with practical application, focusing on interdisciplinary knowledge, skill development, and the latest trends in technology, such as AI and ML, Cyber- security, Full Stack, Data Science, etc.
- **Learning Objectives:** Each course is designed with clear learning outcomes to ensure students understand the relevance of their education to real-world applications. The focus is on building foundational knowledge while advancing to specialized skills that enhance employability and entrepreneurship.
- **Academic Scheduling:** A well-structured academic calendar is developed, ensuring a balanced distribution of coursework, projects, internships, and examinations. Students are guided in course registration, ensuring a smooth academic journey through carefully timed assessments, practical experiences, and project work.

- **Project-Based Learning:** We emphasize experiential and project-based learning to foster critical thinking, problem-solving, and innovation. Through real-world projects, internships, contests, and collaborative opportunities, students gain practical insights into engineering challenges.

- **Continuous Evaluation:** An ongoing evaluation system is employed with periodic assessments, ensuring continuous learning and improvement. The system includes practical exams, theoretical assessments, internships, and project evaluations, providing a holistic view of student progress.

Through strategic educational planning and precise execution, we ensure that our students graduate with the skills and knowledge required to meet the demands of industry and society.

7.4 Academic Journey

7.5 Curriculum Structure and Degree Requirements

The B. Sc (Hons.) Computer Science program at the School of Engineering & Technology offers a comprehensive, structured curriculum designed to meet the demands of the rapidly evolving field of computer science. The program is spread across six semesters, balancing foundational knowledge, advanced topics, practical skills, and interdisciplinary learning. The total credit requirement for the B. Sc (Hons.) Computer Science program has 125 credits.

Course Categories and Credit Distribution:

1. **Ability Enhancement Courses (AEC) – 8 credits:** Courses designed to enhance students' communication skills, critical thinking, and other essential abilities necessary for academic and professional success.

2. **MOOC – 4 credits: MOOC** courses aimed at sharpening students' skills in specific areas such as Competitive Programming, providing opportunities to engage in coding practice and challenges.

3. **Internships (INT) – 12 credits:** Practical industry experience through intern- ships, allowing students to apply theoretical knowledge in real-world settings and gain valuable professional exposure.

4. **Major Courses – 46 credits:** Core courses in computer science covering key topics such as programming, data structures, algorithms, databases, software engineering, and computer networks, forming the foundation of the degree.

5. **Discipline Specific Elective Courses – 16 credits:** Elective courses that allow students to specialize in areas outside their major, fostering interdisciplinary learning and providing flexibility in their academic journey.

6. **Open Electives – 9 credits:** Courses offered from other disciplines, enabling students to explore a variety of subjects beyond their primary field of study in computer science.

7. **Skill Enhancement Courses (SEC) – 10 credits:** Practical, hands-on courses focused on developing specific technical skills, including programming languages, software tools, and other industry-relevant proficiencies.

8. **Projects (PROJ.) – 12 credits:** Significant project work, both individual and group-based, encouraging students to apply their knowledge to solve complex problems and innovate within the field.

9. **Value-Added Courses (VAC) – 6 credits:** Courses aimed at fostering holistic development, emphasizing ethics, social responsibility, entrepreneurship, and leadership capabilities.

10. **Community Service- 02 credits:** Courses related to clubs and society in university with different domains.

Course Category and Credits Summary:

Course Category	**Credits**
Ability Enhancement Courses (AEC)	8
Discipline Specific Elective	16
Internships (INT)	12
Major Courses	46
MOOC	04
Open Electives	9
Skill Enhancement Courses (SEC)	10
Projects (PROJ)	12
Value-Added Courses (VAC)	06
Community Service	02
Total	**125**

Program Name	**Semester**						**Total Credits**
	I	**II**	**III**	**IV**	**V**	**VI**	
B.Sc (H) CS	21	22	24	24	20	14	**125**

7.6 Course Registration and Scheduling

7.6.1 Major and Minor Selection

In the B.Sc (Hons.) Computer Science program, students can specialize in their field through Major and Minor selections, which allow them to tailor their academic journey according to their interests and career aspirations.

The Major component comprises core Computer Science and Engineering courses, providing a strong foundation in fundamental concepts such as algorithms, data structures, software engineering, and computer networks.

For the Minor component, students can choose from department-specific electives offered in four cutting-edge domains: Artificial Intelligence, Full Stack Development (FSD), Cyber Security, and Cloud Computing. This flexibility empowers students to gain specialized knowledge and skills in areas of their choice, enabling them to pursue diverse career paths or advanced studies in these high-demand fields. The Minor selection allows students to complement their major coursework with focused expertise, preparing them for dynamic roles in the evolving tech industry.

7.6.2 Internships/Projects/Dissertations/Apprenticeships

In alignment with our commitment to providing practical, hands-on experiences that complement academic learning, our curriculum incorporates a range of internships, projects, and other experiential learning opportunities. These components are designed to enhance students' skills, apply theoretical knowledge in real-world settings, and prepare them for successful careers in computer science and technology. Below is an overview of the related courses offered:

1. **Summer Internships:** Summer internships are integrated into our curriculum at various stages to ensure students gain relevant industry experience. These intern- ships are structured as follows:

   - **ETCCIN304: Summer Internship I (2 Credits)**
     - **Objective:** To provide students with their first exposure to a professional work environment, where they can apply fundamental concepts learned in their coursework.
     - **Duration:** Summer term, typically spanning 6-8 weeks.
     - **Focus:** Gaining initial industry experience, understanding professional practices, and developing workplace skills.
   - **ETCCIN504: Summer Internship II (2 Credits)**
     - **Objective:** To deepen students' industry experience by engaging them in more complex tasks and projects.
     - **Duration:** Summer term, typically spanning 6-8 weeks.
     - **Focus:** Applying advanced concepts and techniques, participating in meaningful projects, and refining technical and soft skills.

2. **Minor Projects:** Minor projects offer students the opportunity to engage in focused, short-term projects that emphasize the application of core concepts in practical scenarios:

   - **ETCCPR205: Minor Project-I (2 Credits)**
     - **Objective:** To introduce students to project-based learning, where they work on small-scale projects that require the application of fundamental principles.
     - **Focus:** Developing project planning, execution, and presentation skills.
   - **ETCCPR404: Minor Project-II (2 Credits)**
     - **Objective:** To build on the skills acquired in Minor Project-I, with increased complexity and scope.
     - **Focus:** Enhancing problem-solving abilities, project management skills, and technical proficiency.

3. **Major Project/Industrial Training/Start-up Project:** This capstone project is designed for students to engage in extensive, real-world problem-solving in a professional context:

   - **ETCCPR503 and ETCCPR602: Major Project/Industrial Training/Start-up Project (08 Credits)**
     - **Objective:** To provide comprehensive, in-depth project experience in collaboration with industry partners, research institutions, or start-ups.

– **Focus:** Addressing complex computer science problems, engaging in innovative research, or contributing to entrepreneurial ventures. This project requires a significant time commitment and culminates in a detailed report and presentation.

These experiential learning components are integral to our educational philosophy, ensuring that students not only acquire theoretical knowledge but also develop practical skills and gain valuable industry experience. Our structured approach to internships and projects prepares students to meet the demands of the computer science profession and excel in their future careers.

7.6.3 Academic Support Services (Slow & Advanced Learners)

Identifying slow and advanced learners is crucial for providing personalized and effective education. By assessing students' performance through mid-term evaluations and internal assessments, institutions can tailor support to meet individual learning needs. Slow learners benefit from remedial classes, extra assignments, and personalized attention, helping them bridge knowledge gaps and progress academically. Meanwhile, advanced learners are offered opportunities for academic enrichment, including participation in technical events, research projects, and career counseling. This structured approach ensures holistic development, fosters individual growth, and enhances overall academic performance.

Mechanism for identifying slow and advanced learners

- **Mid-term Evaluation:** Conduct mid-term exams with a weightage of 20%.
- **Assessment of Learning Levels:** Evaluate students based on their performance in the mid-term and other internal assessments.
- **Slow Learners Identification:**
  - If a student's marks are ≤55%, they are categorized as slow learners.
  - Remedial support includes additional classes, extra assignments, and notes.
- **Advanced Learners Identification:**
  - If a student's marks are ≥80%, they are categorized as advanced learners.
  - They are provided with opportunities for career counseling, participation in technical events, and advanced courses.

7.7 Student Support Services

- **Mentor-Mentee**
- **Counselling and Wellness Services**
- **Career Services and Training**

7.8 Learning and Development Opportunities

- **Laboratories and Practical Learning**
- **Experiential Learning**
- **Case-Based Learning/Problem-Based Learning/Project Based Learning**
- **Workshops, Seminars, Guest Lectures**
- **Inside & Outside Classroom Learning**
- **Holistic Education**

7.9 Assessment and Evaluation

7.9.1 Grading Policies and Procedures for Theory Courses, Practical Courses, Projects, Internships, Dissertation

Theory Courses: Grading for theory courses is based on a comprehensive evaluation scheme designed to assess both continuous performance and final examination results. The assessment is divided as follows:

- **Continuous Assessment (30 Marks):** Includes diverse components such as project work, quizzes, assignments, essays, presentations, participation, case studies, and reflective journals. These components are evenly spaced throughout the semester to gauge ongoing student performance.
- **Mid-Term Exam (20 Marks):** A formal examination conducted midway through the semester to assess understanding and retention of the material covered up to that point.
- **End-Term Examination (50 Marks):** A comprehensive exam covering the entire syllabus, designed to evaluate students' overall understanding and application of the course content.

Practical Courses: Practical assessments focus on the students' ability to apply theoretical knowledge to practical tasks. The evaluation components include:

- **Conduct of Experiment (10 Marks):** Assessment of the student's practical skills and ability to follow experimental procedures.
- **Lab Records (10 Marks):** Evaluation of the completeness and accuracy of lab documentation.
- **Lab Participation (10 Marks):** Involvement and engagement in laboratory activities.
- **Lab Project (20 Marks):** Performance and results of a project-based laboratory assignment.
- **End-Term Practical Exam and Viva Voce (50 Marks):** A comprehensive practical examination and oral assessment to evaluate the application of lab skills and theoretical knowledge.

Note: Students must secure at least 40% in both internal and external components (theory and practical) to pass the course.

7.9.2 Feedback and Continuous Improvement Mechanisms (Details to be provided by school)

Feedback is integral to fostering continuous improvement and enhancing the learning experience. Our feedback mechanisms include:

- **Student Feedback Surveys:** Regular surveys to collect student opinions on course content, teaching methods, and overall learning experience.
- **Peer Reviews:** Evaluation of teaching practices by peers to ensure adherence to educational standards and continuous improvement.
- **Faculty Reviews:** Periodic reviews of faculty performance based on feedback and assessment outcomes to support professional development.
- Continuous improvement is driven by analyzing feedback, implementing necessary changes, and reviewing the effectiveness of modifications.

7.9.3 Academic Integrity and Ethics

Upholding academic integrity and ethical standards is crucial in maintaining the quality of education. Our policies include:

- **Cheating Prevention:** Procedures to prevent and address cheating during exams and assignments.
- **Ethical Conduct:** Clear guidelines on academic conduct and ethics, with emphasis on honesty and responsibility in all academic work.

7.9.4 Examination and Evaluation Methods (Details to be provided by school)

Our examination and evaluation methods ensure a fair and comprehensive assessment of student performance:

- **Theory Examinations:** A mix of continuous assessments and formal exams to evaluate students' understanding and application of course material.
- **Practical Examinations:** Assessment of hands-on skills and practical knowledge through lab experiments and projects.
- **ICT Tools:** Utilization of Moodle LMS and interactive teaching boards to support teaching and assessment, providing students with access to course materials, assignments, and feedback.

Evaluation Components:

- **Lecture PPTs and Video Lectures:** Used to deliver course content and reinforce key concepts.
- **Problem-Based and Project-Based Assignments:** Encourage practical application of theoretical knowledge and critical thinking.
- **Question Banks and Model Papers:** Provide practice and preparation for exams.

- **Continuous Assessment and Support:** Regular assessments and feedback to monitor and support student progress throughout the semester.

These evaluation methods are designed to provide a holistic assessment of students' knowledge and skills, ensuring they meet the required learning outcomes and standards.

Program Scheme

Program Name	B.Sc (Hons.) Computer Science
Total Credits	125
Total Semesters	6

Semester I (Odd Semester)

S.N	Course_Code	Category	Course Title	L	T	P	Credits
1	ETCCCS101	Major	Mathematical Foundations for Computer Science	3	0	0	3
2	ETCCWD102	Major	Web Development Essentials	3	0	2	4
3	ETCCPP103	Major	Foundations of Programming using Python	3	0	2	4
4	ETCCCR104	Major	Foundations of Computing and Career Readiness	4	0	0	4
5	SEC	SEC	Analytics for Decision Making	2	0	0	2
6	SEC	SEC	Introduction to Design Thinking and Prototyping	1	0	2	2
7	VAC	VAC	VAC-I (Environmental Studies)	2	0	0	2
		TOTAL		**18**	**0**	**6**	**21**

Semester II (Even Semester)

S.N	Course_Code	Category	Course Title	L	T	P	Credits
1	ETCCDS201	Major	Basics of Data Structures	3	0	2	4
2	ETBCCC202	DSE	Specializaton Course-I (Cloud Computing Foundations and Practices)	3	0	2	4
3	ETCCWD203	Major	Responsive and Dynamic Web Design	3	0	2	4
4	OEC	Open Elective	Open Elective-I	3	0	0	3
5	AEC	AEC	Verbal Ability	2	0	0	2
6	ETCCOS204	Major	Operating System Fundamentals	3	0	0	3
7	ETCCPR205	Proj	Minor Project-I	2	0	0	2
			TOTAL	**19**	**0**	**6**	**22**

Semester III (Odd Semester)

S.N	Course_Code	Category	Course Title	L	T	P	Credits
1	ETCCDA301	Major	Principles of Algorithm Design and Analysis	3	0	2	4
2	ETCCWB302	Major	Building Scalable Web Backends	3	0	2	4
3	ETBCML303	DSE	Specializaton Course-II (Machine Learning: Core Concepts and Techniques)	3	0	2	4
4	OEC	Open Elective	Open Elective-II	3	0	0	3
5	AEC	AEC	Communication & Personality Development	2	0	0	2

6	ETCCIN304	INT	Summer Internship-I (Assessment)	0	0	4	2
7	SEC	SEC	Competitive Coding - I	2	0	0	2
8	CS	CS	Community Service	1	0	0	1
9	VAC	VAC	VAC-II	2	0	0	2
TOTAL				19	0	10	24

[1]Note: For "**Summer Internship-I**" students have to complete 6 weeks internship during the summers and submit a completion certificate. Students will be evaluated on a scale of 100 based on their learning outcomes during the 3rd semester for allocation of marks for internship.

Semester IV (Even Semester)

	Course_Code	Category	Course Title	L	T	P	Credits
1	ETCCJO401	Major	Fundamentals of Java and Object-Oriented Design	3	0	2	4
2	ETCCMM402	Major	Introduction to Database Management Systems	3	0	2	4
3	ETCCEH403	DSE	Specializaton Course-III (Principles of Information Security and Ethical Hacking)	3	0	2	4
4	OEC	Open Elective	Open Elective-III	3	0	0	3
5	AEC	AEC	Arithmetic and Reasoning Skills	2	0	0	2
6	ETCCPR404	Proj	Minor Project-II	0	0	4	2
7	SEC	SEC	Competitive Coding - II	2	0	0	2
8	CS	CS	Club/Society	1	0	0	1
9	VAC	VAC	VAC-III	2	0	0	2

		TOTAL	19	0	10	24

[2]Note: For the "**Minor Project-II**," students will undergo internal evaluation, which will be graded on a scale of 100 marks.

Semester V (Odd Semester)

S.N	Course_Code	Category	Course Title	L	T	P	Credits
1	ETCCGA501	DSE	Specializaton Course-IV (Practical Applications of Generative AI)	3	0	2	4
2	ETCCSE502	Major	Essentials of Software Engineering	3	0	2	4
3	SEC	SEC	Competitive Coding - III	2	0	0	2
4	**ETCCIN504**	INT	Summer Internship-II (Assessment)	0	0	4	2
5	AEC	AEC	Comprehensive Placement Preparation	2	0	0	**2**
6	ETCCPR503	PROJ	Major Project-I	0	0	0	4
7	MOOC	MOOC	MOOC (Swayam/ NPTEL/AICTE's ELIS)	2	0	0	2
TOTAL				**12**	**0**	**8**	**20**

Semester VI (Even Semester)

	Course_Code	Category	Course Title	L	T	P	Credits
1	ETCCIN601	INT	Industry Internship	0	0	16	8
2	ETCCPR602	PROJ	Major Project-II	0	0	8	4
3	MOOC	MOOC	MOOC (Swayam/ NPTEL/AICTE's ELIS)	2	0	0	2
				2	0	24	14

[3]**Note:**

- For the "Summer Internship," students are required to complete a 6-week full-time industry in- ternship during the summer and submit a completion certificate. The evaluation will occur in the 7th semester, with students graded on a scale of 100 marks.
- Students are required to undertake a full-time industry internship for the entire semester. They are not permitted to enroll in any courses as an alternative to the internship. Students can choose from the following internship options:
  - Industry project
  - Research & Development project
  - Start-up project
- Evaluation will be based on internal assessments, with no end-term exams applicable.

Evaluation for the “Comprehensive Placement **Preparation Boot Camp**" will be done at the internal level. There will be no end term exams for this. Students will be required to undergo a specialized placement preparation boot camp program offered by the CDC/School in a hybrid mode. No end-term exams will be conducted; instead, internal evaluations will be carried out, with a total of 100 marks. The modules will focus on preparing students for technical interviews, coding questions, aptitude and soft skills, and mock interviews.

VAC-III

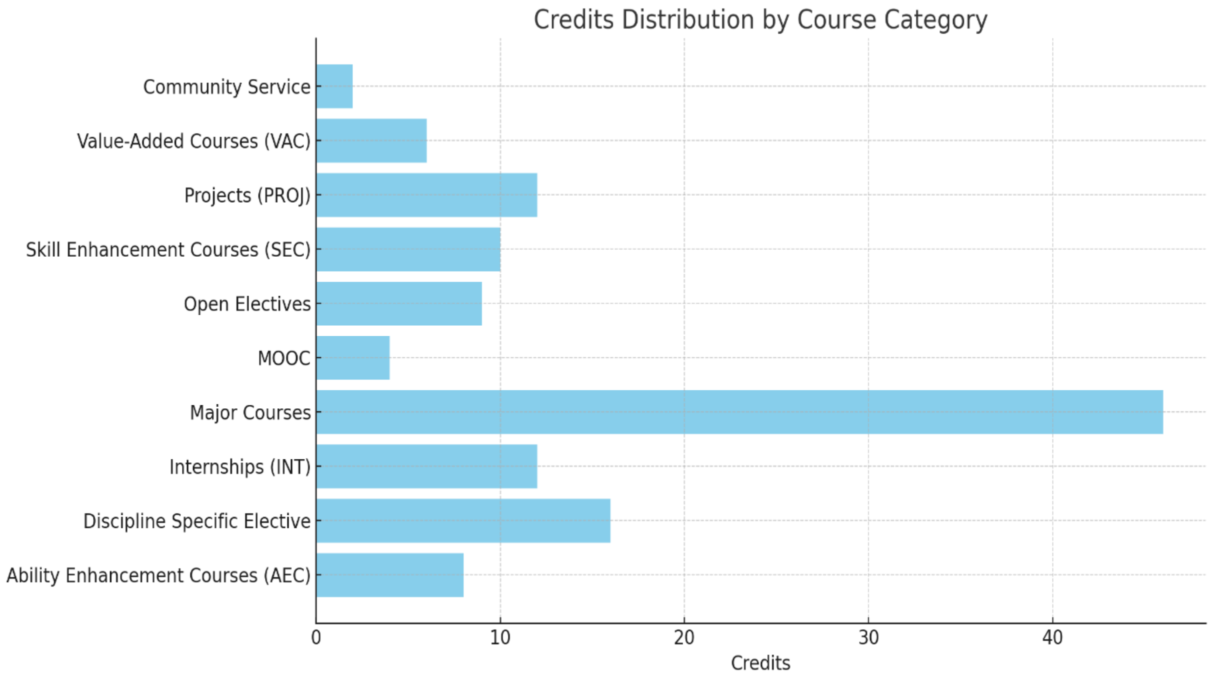
S.No	**Course Code**	**Course Title**	**L**	**T**	**P**	**C**
1	**VAC170**	Design Thinking & Innovations for Engineers	-	-	-	2
2	**VAC171**	AWS Cloud Fundamentals	-	-	-	2
3	**VAC172**	Web Development with Open Source Frameworks	-	-	-	2
4	**VAC173**	Google Data Analytics	-	-	-	2
5	**VAC174**	Software Testing using Open Source Frameworks	-	-	-	2
6	**VAC175**	Database Management with Open Source Frameworks	-	-	-	2
7	**VAC176**	Cyber Security with Open Source Frameworks	-	-	-	2
8	**VAC185**	Practical Robotics and UAV Applications	-	-	-	2
9	**VAC186**	Applied Automotive Engineering: Hands-On Practices and Innovations	-	-	-	2
10	**VAC187**	Practical Research Methodology for Engineers	-	-	-	2

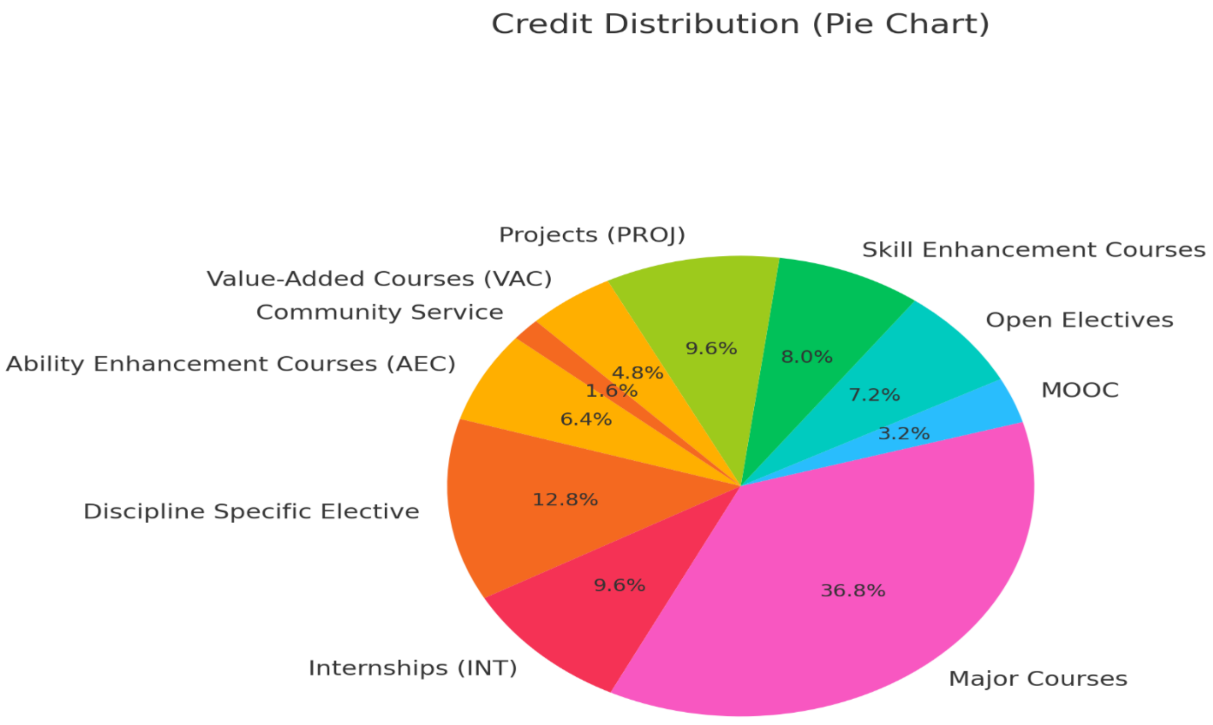
Program Credits

Program Name	**Semester**						**Total Credits**
	I	**II**	**III**	**IV**	**V**	**VI**	
B.Sc (H) CS	21	22	24	24	20	14	**125**

Total Credits: ***125***

Course Categories Distribution





Evaluation Scheme (Theory)

Evaluation Components	**Weightage**
Internal Marks (Theory) 1. **Continuous Assessment (30 Marks)** (All the components to be evenly spaced) Project/ Quizzes/ Assignments and Essays/ Presentations/ Participa- tion/ Case Studies/ Reflective Journals (minimum of five components to be evaluated)	**30 Marks**
2. **Internal Marks (Theory) – Mid Term Exam**	**20 Marks**
External Marks (Theory): - End term Examination	**50 Marks**
Total	**100 Marks**

Note: *It is compulsory for a student to secure **40%** marks in Internal and End Term Examination separately to secure minimum passing grade.*

Evaluation Scheme (Laboratory)

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

Note: *It is compulsory for a student to secure* ***40%*** *marks in Internal and End Term Practical Exam and Viva Voce separately to secure minimum passing grade.*

Detailed Syllabus

Semester: 1

Mathematical Foundations for Computer Science

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Mathematical Foundations for Computer Science	ETCCCS101	3-0-0	3
Type of Course:	Major		
Pre-requisite(s):	None		

Course Description:

This course introduces essential mathematical foundations required for computer science. It focuses on logic, discrete structures, linear algebra, probability, and calculus. Each topic is linked to practical computing applications such as algorithm design, data analysis, and machine learning.

The Course Outcomes (COs)

On completion of the course the participants will be:

COs	Statements
CO 1	Applying logic, set theory, and graph theory to computational problems.
CO 2	Utilizing linear algebra methods to solve computing and data problems.
CO 3	Understanding data using probability and statistics.
CO 4	Analyzing calculus concepts and apply optimization techniques in computing contexts.

Unit-Wise Syllabus

Unit Number: 1	**Title: Discrete Mathematics**	**No. of hours: 15**
Content:		
• Propositional and Predicate Logic • Proof Techniques: Direct, Indirect, Mathematical Induction • Set Theory and Functions • Combinatorics: Permutations and Combinations • Basic Graph Theory: BFS, DFS **Hands-On:** • Python programs for logical evaluation and set operations		

• Simulations for combinatorics and graph traversal		
Unit Number: 2	**Title: Linear Algebra**	**No. of hours: 10**
Content:		
• Vectors and Matrices • Matrix Operations: Addition, Multiplication, Transpose • Solving Linear Systems of Equations • Applications in Computer Graphics and Machine Learning **Hands-On:** • Matrix operations using NumPy • Linear system solvers and dot product visualizations		
Unit Number: 3	**Title: Probability and Statistics**	**No. of hours: 10**
Content:		
• Conditional Probability and Bayes' Theorem • Random Variables and Distributions (Bernoulli, Binomial, Normal) • Measures of Central Tendency and Dispersion • Correlation and Simple Linear Regression **Hands-On:** • Data analysis using pandas and matplotlib • Probability simulations and regression plots		
Unit Number: 4	**Title: Calculus and Optimization**	**No. of hours: 10**
Content:		
• Limits and Continuity • Differentiation and Integration (Single-variable) • Functions of Several Variables, Partial Derivatives • Gradient Descent and Optimization **Hands-On:** • Symbolic calculus with SymPy • Gradient descent implementation in Python		

Learning Experience for Mathematical Foundations for Computer Science

• Interactive Lectures:
Students will be engaged through concept-driven lectures enriched with real-world computing applications. Each topic—ranging from logic and proof strategies to linear algebra and calculus—will be introduced using interactive visualizations, algorithmic examples, and relatable use cases in computer science (e.g., AI logic models, graphics transformations, and optimization in machine learning).
• Hands-On Programming Labs:
Every unit integrates Python-based labs to reinforce theory through practice. Students will implement logic evaluators and set operations, simulate combinatorics problems, traverse graphs using BFS/DFS, manipulate matrices with NumPy, and run probability simulations and regression models using pandas and matplotlib. For calculus, symbolic computation and gradient descent will be implemented using SymPy and custom Python code, fostering a strong link between mathematical formulation and computational execution.
• Collaborative Activities:
Pair and group-based coding tasks will allow students to co-develop simulation programs for combinatorics, graph algorithms, linear system solvers, and data-driven statistical models. These activities promote teamwork, logical reasoning, and real-time debugging in a collaborative environment.
• Assignments & Case Studies:
Weekly assignments will include problem sets and mini-projects such as:

- Modeling logic circuits and evaluating logical expressions.
- Performing matrix transformations relevant to 2D graphics.
- Analyzing real datasets for statistical inference.
- Implementing and tuning gradient descent in small ML tasks. These tasks will bridge theoretical knowledge with computational problem-solving.

• Support & Feedback:
Students will receive continuous academic support via coding clinics, peer-assisted learning sessions, and instructor-led forums. Feedback will be detailed and focused on not only correctness but also efficiency, clarity, and programming style in submitted code and reports.
• Assessments:
Assessment will be multifaceted, including:

- Conceptual quizzes to test theoretical foundations.
- Python coding tests for evaluating hands-on proficiency.
- Mid-term and final exams featuring logic proofs, matrix operations, probability applications, and calculus problems.
- A final mini-capstone project integrating multiple units (e.g., a simulation involving statistics and optimization techniques).

Textbooks and References

1. Kenneth H. Rosen – *Discrete Mathematics and Its Applications*
2. Gilbert Strang – *Introduction to Linear Algebra*
3. Sheldon Ross – *Introduction to Probability and Statistics*
4. Allen Downey – *Think Stats* (Free online)
5. S. Sastry – *Introductory Methods of Numerical Analysis*

Web Development Essentials

Program Name:	B.Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Web Development Essentials	ETCCWD102	3-0-2	4
Type of Course:	Major		
Pre-requisite(s):	Programming Fundamentals (Python/JavaScript)		

Course Description: This course equips BCA students with essential skills in modern web development. It introduces core technologies like HTML, CSS, JavaScript, along with frontend frameworks, backend basics, version control, and deployment. By the end, students will be capable of designing, developing, and deploying complete responsive web applications.

The Course Outcomes (COs)

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

COs	Statements
CO 1	Creating visually responsive web pages using HTML, CSS, and JavaScript.
CO 2	Developing interactive frontend components using JavaScript and popular UI libraries.
CO 3	Set up basic backend functionality and link it to web frontends.
CO 4	Deploying complete web applications using accessible cloud platforms.

Course Outline

Unit Number: 1	**Title:** ***Web Foundations & Responsive Design***	**No. of hours: 15**
Content:		
• Basics of Web, HTTP/S, and client-server architecture • HTML5 elements, structure, forms, semantic tags • CSS3: Box Model, Selectors, Flexbox, Grid • Responsive design with media queries • *Hands-on:* Build a personal portfolio site with responsive layout		
Unit Number: 2	**Title:** ***JavaScript for Interactivity***	**No. of hours: 15**

Content:		
• JavaScript syntax, variables, operators, types • Arrays, objects, functions • DOM manipulation and event handling • Form validation and dynamic interactions *Hands-on:* Create a quiz or calculator using JavaScript		
Unit Number: 3	**Title:** ***UI Libraries & GitHub Workflow***	**No. of hours: 15**
Content:		
• Introduction to Bootstrap or Tailwind CSS • Basics of JavaScript libraries (React or Vue overview) • Git fundamentals: add, commit, push, branches • Collaboration using GitHub and GitHub Pages *Hands-on:* Redesign site with a UI library & deploy on GitHub		
Unit Number: 4	**Title:** ***Backend Integration & Hosting***	**No. of hours: 15**
Content:		
• Introduction to Node.js and Express.js • Creating RESTful APIs and handling form data • Storing input with MongoDB (or basic SQL alternatives) • Deployment using Vercel, Render, or Netlify • *Hands-on:* Create a contact form with backend integration		

Learning Experience for Fundamentals of Web Development Essentials

- **Interactive Lectures:** Engage students with detailed explanations of web technology concepts using diagrams, live coding demonstrations, and interactive discussions on client-side and server-side technologies.
- **Hands-On Labs:** Provide practical sessions where students design and develop web pages, implement JavaScript functionality, and create interactive web elements using HTML, CSS, and JavaScript.
- **Group Projects:** Encourage collaborative learning through group projects where students design and develop complete websites, applying concepts like responsive design, form validation, and dynamic content generation.
- **Assignments & Case Studies:** Assign tasks that involve real-world scenarios, such as creating web pages that follow industry standards, optimizing web perfor- mance, and implementing security measures.
- **Support & Feedback:** Offer ongoing support through office hours, online forums, and coding workshops. Provide detailed feedback on assignments, projects, and lab work to guide students in improving their technical skills.
- **Assessments:** Include quizzes, coding exercises, and final exams that test students' understanding of web technologies, their ability to implement web solutions, and their design proficiency.

Text Books & References

- *HTML and CSS: Design and Build Websites* – Jon Duckett
- *Head First JavaScript Programming* – E. Freeman
- *Learning PHP, MySQL & JavaScript* – Robin Nixon
- Mozilla Developer Network (MDN): https://developer.mozilla.org
- GitHub Docs: https://docs.github.com

Web Development Essentials Lab

Lab Task 1: Responsive Portfolio Website

- Objective: Create a personal portfolio with mobile-first design
- Activities: Use HTML5, Flexbox, and Grid. Add responsive navbar and host on GitHub Pages
- Focus: Semantic HTML, layout, responsive design

Lab Task 2: JavaScript To-Do List

- Objective: Build a task manager with interactive features
- Activities: Use DOM manipulation, add/remove tasks, store in localStorage
- Focus: JS logic, real-time updates, data persistence

Lab Task 3: GitHub Collaboration Project

- Objective: Collaborate on a simple team project

- Activities: Use branches, pull requests, manage issues, deploy with GitHub Pages
- Focus: Git workflows, collaboration, version control

Lab Task 4: Full-Stack Contact Form

- Objective: Build a contact form connected to backend and database
- Activities: Use Node.js, Express, and MongoDB. Deploy to Vercel or Render
- Focus: REST APIs, backend integration, deployment flow

Capstone Project: Live Event Web Application

- Objective: Develop and launch a complete web application for managing event registrations and schedules
- Activities:
  - Design frontend for signups, schedules, announcements
  - Build backend APIs and store data in MongoDB or Firebase
  - Test with users, debug, and deploy live
- Focus: End-to-end full-stack app creation, real-world usability
- Tools: HTML, CSS, JS, Node.js, Express, MongoDB/Firebase, Vercel, GitHub

Foundations of Programming using Python

Program Name:	B.Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Foundations of Programming using Python	ETCCPP103	3-0-2	4
Type of Course:	Major		
Pre-requisite(s):	Basic Computer Knowledge/ Introduction to Computing)		

Course Description: This practical course introduces BSc students to programming using Python. From writing simple scripts to visualizing data, learners will develop problem-solving skills using core Python syntax, control structures, functions, file handling, and beginner-level libraries like NumPy, Pandas, and Matplotlib. The course uses small, real-world problems and step-by-step lab exercises to reinforce key concepts.

The Course Outcomes (COs)

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

COs	Statements
CO 1	Understand Python basics like variables, data types, operators, and string operations.
CO 2	Use decision-making, loops, and functions to solve basic problems.

CO 3	Work with files, exceptions, and simple class-based programs.
CO 4	Load and visualize small datasets using Pandas and Matplotlib.

Course Outline

Unit Number: 1	Title: *Python Basics & Setup*	No. of hours: 15
Content:		
• Installing Python and using IDLE/VS Code • - Writing simple scripts and using the Python shell • - Variables, data types (int, float, string, bool) • - Basic I/O and string formatting with f-strings • - Operators and expressions • - Quick Task: Set up a project folder, install Python, write a "Hello, BSc!" program, and use basic input/output.		
Unit Number: 2	Title: *Control Structures & Functions*	No. of hours: 15
Content:		
• - if-else and nested conditionals • - for and while loops • - Simple function definition and use • - List, tuple, dictionary basics • - Practice with basic algorithms (e.g., sum, max, sort) • - Quick Task: Write a function to calculate average marks of a student and use loops for repeated analysis.		
Unit Number: 3	Title: *Classes, Files & Error Handling*	No. of hours: 15
Content:		
• - Understanding classes with basic attributes • - Simple object creation, __init__, and method definitions • - Reading/writing text files • - Try–except blocks for error handling • - Simple JSON format intro (load/save) • - Quick Task: Build a basic class for storing book info, and read/write book data from a text file.		

Unit Number: 4	**Title:** ***Basic Data Analysis & Visualization***	**No. of hours: 15**
Content:		
• Intro to NumPy: arrays and basic operations • Pandas: reading CSV, basic filtering, sorting • Matplotlib: line/bar/scatter charts • Data cleaning (drop NA, fill NA) • Export plots as PNG • Quick Task: Load a weather dataset and create a temperature trend chart.		

Learning Experience for Foundations of Programming using Python

• Interactive Lectures:

Students will be introduced to programming concepts through engaging lectures that combine conceptual explanation with real-time code demonstrations. Concepts such as variables, data types, control flow, functions, object-oriented programming, and file handling are taught using examples that relate to real-life scenarios and problem-solving contexts.

• Hands-On Coding Labs:

Hands-on lab sessions form the core of the learning experience, where students write, run, and debug Python programs in every class. Practical exercises include:
Basic scripting and arithmetic operations.
Decision-making and looping constructs.
List and dictionary manipulation.
Function design and modular programming.
Reading from and writing to files.
These labs help students build coding confidence and reinforce logical thinking.

• Mini Projects:

To apply the acquired knowledge in a practical context, students will work on mini-projects such as:
A simple calculator or quiz app.
A student record management system using file handling.
A basic text-based game or chatbot.
Projects help develop critical thinking and give students a sense of ownership over their learning.

• Pair Programming and Peer Learning:

Students collaborate through pair programming exercises where they alternate between writing code and reviewing a peer’s code. This helps build communication skills and exposes learners to different coding styles and debugging approaches.

• Assignments & Practice Problems:

Weekly assignments covering programming logic, syntax mastery, and use of built-in Python libraries are given. Additionally, online platforms such as HackerRank, LeetCode, or Replit Classroom are used for practice problems, reinforcing algorithmic thinking and problem-solving skills.

• Support & Feedback:

Students receive timely support through dedicated lab sessions, mentoring hours, and discussion forums. Individualized feedback is provided on assignments and projects to help students identify strengths and areas for improvement in their coding practices.

• Assessments:

Evaluations include:
Quizzes focusing on syntax, concepts, and flow control.
Lab tests assessing coding skills and logic implementation.
A mid-term and final exam that include both theoretical and practical components.
A capstone project that demonstrates the student’s ability to build a functional Python application.

Text Books & References

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.

Foundations of Programming using Python Lab

Lab Task 1 – "Simple Expense Tracker" (Unit 1)

- Track daily expenses with categories and amounts.
- Input and store expenses in lists
- Total and average expense
- Basic text output with string formatting
- Bonus: Save to text file

Lab Task 2 – "Student Marks Analyzer" (Unit 2)

- Process and analyze class test scores.
- Input or read from CSV
- Find top score, pass/fail count
- Assign grades based on if-else
- Loop menu for repeated checks

Lab Task 3 – "Mini Library System" (Unit 3)

- Track books using classes and file handling.
- Define Book class with title, author, etc.
- Add/remove/search book
- Save/load records to/from JSON
- Try–except for file errors

Lab Task 4 – "Basic Data Visualizer" (Unit 4)

- Analyze temperature from a CSV dataset.
- Load data with Pandas
- Clean missing entries
- Create line and bar plots
- Save plots as images

Capstone Project – "Electricity Use Summary" (Capstone Project)

- Basic dashboard for household electricity use.
- Load 2–3 CSV files
- Use loops/functions to calculate total use
- Plot usage trend over time
- Export results and chart

Textbooks & References

- 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.

Foundations of Computing and Career Readiness

Program Name:	B.Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Foundations of Computing and Career Readiness	ETCCR104	4-0-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Description: This course introduces foundational concepts in computing and enhances career readiness by integrating digital literacy, problem-solving approaches, and professional development. Students will gain a theoretical understanding of computing systems and apply hands-on digital practices to build confidence in both technical and soft skills. The Course Outcomes (COs)

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

COs	Statements
CO 1	Explaining the core concepts of computers, software, and operating systems.
CO 2	Using digital tools and applying basic cybersecurity practices.
CO 3	Applying logical thinking and flowcharts to solve simple problems.
CO 4	Building soft skills like digital resume creation and career goal setting.

Course Outline

Unit Number: 1	Title: *Basics of Computer Science & Computational Thinking*	No. of hours: 12
Content:		
• Evolution of computing and major milestones. • Basic computer components: CPU, memory, storage, I/O. • Number systems: decimal, binary, hexadecimal conversions. • Simple encoding schemes: ASCII and Unicode. • Types of software: system, application, utility. • Algorithm basics and input-process-output model. • Concepts of computational thinking: abstraction, decomposition, pattern recognition, algorithm design. • Flowcharts and pseudocode basics. • Introduction to ethical computing and responsible tech usage.		
Unit Number: 2	**Title:** ***Linux Basics & Open-Source Tools***	**No. of hours: 12**
Content:		
• Introduction to Linux and its role in IT and development. • Understanding kernel, shell, and filesystem basics. • Using Linux via WSL or VirtualBox. • Common terminal commands: ls, cd, pwd, cp, mv, rm, cat, mkdir. • User and file permissions: chmod, chown. • Monitoring tools: top, ps, kill. • Basic networking tools: ping, wget. • Writing simple shell scripts. • Open-source philosophy and licensing (MIT, GPL).		
Unit Number: 3	**Title:** ***Exploring Technologies & Career Possibilities***	**No. of hours: 12**
Content:		
• Overview of major technology domains: web, AI, cybersecurity, etc. • India and global case studies of tech in action. • Student task: research one domain, job roles, and skills required. • Present via blog post, infographic, or short video. • Mapping subjects to job roles (e.g., DSA → Backend). • Examples of CS in healthcare, agriculture, etc.		

- Highlighting Indian startups and student innovation stories.

Unit Number: 4	Title: *Tools for Coding, Learning, and Collaboration*	No. of hours: 12

Content:

- Using Visual Studio Code and Jupyter for development.
- Git and GitHub basics: add, commit, push, clone.
- Creating simple repositories and README files.
- Using. gitignore and markdown files.
- Learning with HackerRank, W3Schools, and Kaggle basics.
- Organizing study with Notion or Obsidian. - Using MS Teams or Trello for collaboration.

Unit Number: 5	Title: *Career Planning & Skill Building*	No. of hours: 12

Content:

- Set SMART goals for academic and professional development.
- Understanding popular IT certifications.
- Creating a digital professional identity (e.g., LinkedIn, GitHub).
- Participating in coding events, internships, and open-source.
- Awareness of opportunities like Smart India Hackathon and GSoC.
- Importance of networking, time management, and growth mindset.

Learning Experience for Foundations of Foundations of Computing and Career Readiness

• Interactive Lectures & Conceptual Discussions

Students will be introduced to fundamental computing principles, starting with the evolution of computer science and core components of a computer system. Concepts such as number systems, encoding schemes, types of software, and computational thinking will be taught using visual aids, animations, real-life analogies, and flowcharting tools. Ethical computing and responsible technology use will be discussed through scenarios and case studies.

• Practical Labs and Hands-On Sessions

Each unit integrates hands-on learning with tools and environments relevant to the modern computing ecosystem:

- Linux Basics (Unit 2) will be reinforced by command-line practice on WSL or VirtualBox, basic shell scripting, and file system navigation.
- Tool Exploration (Unit 4) includes version control using Git/GitHub, repository setup, markdown writing, and using platforms like VS Code, Jupyter, Notion, and Trello.
- Number system conversions and algorithm modeling will be reinforced through interactive coding or flowcharting tasks.

• Self-Directed and Inquiry-Based Learning

In Unit 3, students will explore emerging technology domains (e.g., AI, Web, Cybersecurity) and research real-world job roles and applications. They will present their findings creatively via blog posts, infographics, or videos, encouraging digital communication skills and independent learning.

• Project-Based and Peer Learning

Group projects will be facilitated, such as:

- Collaborative development of flowcharts and pseudocode for a problem.
- Team blog or GitHub documentation projects.
- Joint presentations mapping subject areas to career roles.

This structure encourages mutual learning and improves communication and collaboration skills.

• Career Exploration and Professional Development

In Unit 5, students will:

- Create SMART goals for their learning journey.
- Build or enhance their LinkedIn and GitHub profiles.
- Explore opportunities like Smart India Hackathon, GSoC, or open-source programs.
- Participate in peer-review activities for resume building and online presence evaluation.

• Support & Mentoring

Students will have access to:

- Faculty guidance through office hours and LMS discussions.
- Peer mentoring from senior students or alumni to explore career paths and learning platforms.
- Resource banks with tutorials, tools, and open-source repositories.

• Assessments

- Formative assessments via quizzes on logic, number systems, and Linux commands.
- Practical assessments through GitHub submissions, terminal tasks, and tool-based mini-projects.
- Summative evaluation through presentations, blog posts, shell scripts, and infographics that reflect learning and career understanding.

Textbooks & References

1. Computer Science: An Overview – J.G. Brookshear & D. Brylow, Pearson Education, 2021

2. The Linux Command Line: A Complete Introduction, William E. Shotts Jr., 2019, 13th Edition, ISBN-13: 978-1593279523

Analytics for Decision Making

Program Name:	B.Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Analytics for Decision Making		2-0-0	2
Type of Course:	SEC		
Pre-requisite(s):	Basic Statistics and Spreadsheet Usage		

Course Description: This course equips students with practical data analysis techniques to support decision-making in business and technological environments. While primarily theory-based, the course includes structured hands-on activities using spreadsheets, visualization tools, and real-world scenarios to bridge theory with practice. On completion of the course, the participants will be able to:

COs	**Statements**
CO 1	Understand the importance of analytics in strategic and operational decision-making.
CO 2	Apply descriptive and diagnostic analytics techniques to real-world problems.
CO 3	Use statistical methods to interpret, analyze, and infer data-driven insights.
CO 4	Communicate decisions effectively using data visualization and storytelling.

Course Outline

Unit Number: 1	**Title:** ***Introduction to Analytics and Decision Making***	**No. of hours: 8**
Content:		
• What is Analytics? Types: Descriptive, Diagnostic, Predictive, Prescriptive • Structured vs Unstructured Data • Role of analytics in modern organizations • Decision-making frameworks **Hands-On Task**: • Analyze a business scenario to identify types of analytics involved • Class discussion: Data-driven vs intuition-based decisions in caselets		
Unit Number: 2	**Title:** ***Descriptive and Diagnostic Analytics***	**No. of hours: 8**
Content:		
• Data summarization: Mean, Median, Mode, Range, Variance • Frequency tables, cross-tabulation • Pareto charts, Fishbone diagrams for cause analysis **Hands-On Task**: • Use Excel to analyze a small sales dataset and summarize key metrics • Construct a Pareto chart to identify top contributing factors in a problem		
Unit Number: 3	**Title:** ***Statistical Thinking and Business Inference***	**No. of hours: 8**
Content:		
• Basics of correlation and regression • Probability concepts with business applications • Hypothesis testing: concepts, types of errors, interpreting results **Hands-On Task**: • Conduct simple correlation and linear regression using Excel or Google Sheets • Group activity: Develop and test a hypothesis using sample survey dat		
Unit Number: 4	**Title:** ***Data Visualization and Insight Communication***	**No. of hours: 6**
Content:		
• Visual analytics: importance and best practices		

- Chart types: bar, pie, line, histogram, scatter
- Introduction to tools: Tableau Public / MS Excel / Power BI
- Communicating decisions from charts

Hands-On Task:

- Create dashboards in Excel (Pivot Charts, Conditional Formatting)
- Use Tableau Public or Power BI to visualize and narrate a case (demo level)
- Storytelling activity: Present insights and recommendations from visuals

Learning Experience for Analytics and Decision Making

• Interactive Lectures & Concept Demonstration

Each unit begins with instructor-led sessions that break down analytical concepts using real-world examples and business scenarios. Concepts such as types of analytics, hypothesis testing, data visualization, and decision-making frameworks are explained using:

- Case Study and storytelling
- Industry examples
- Live demonstrations using tools like Excel, Tableau, and Power BI

These interactive lectures aim to build foundational understanding and contextual relevance.

• Hands-On Practice and Tool-Based Learning

Students will work on curated hands-on tasks after every unit, ensuring they not only understand but can also apply analytical methods:

- Excel-based analysis for descriptive statistics, cross-tabulations, and Pareto charts.
- Regression and correlation implementation using spreadsheet tools.
- Dashboard creation and visual storytelling using Tableau Public, MS Excel, or Power BI.
- Flow-based hypothesis testing with real or simulated data.

This lab-oriented practice reinforces learning through doing.

• Collaborative Group Activities

Group-based tasks are incorporated in every unit to encourage peer learning, communication, and collaboration. Activities include:

- Debates and discussions on data-driven vs. intuition-based decision-making.
- Group hypothesis testing projects using survey data.
- Joint visual storytelling presentations using created dashboards.

Students learn to analyze collaboratively, communicate insights effectively, and present business recommendations with clarity.

• Real-World Case-Based Learning

Caselets from various sectors like retail, healthcare, and banking are used to connect analytics concepts to business decision-making. These are integrated into:

- Decision-making simulations
- Root cause analysis exercises using Fishbone and Pareto methods
- Role-play activities for data analyst and business stakeholder interaction

• Digital Literacy and Tool Exploration

The course emphasizes digital fluency by introducing students to:

- Data analytics platforms (Excel, Google Sheets, Tableau Public, Power BI)
- Data storytelling tools (infographics, slide decks)
- Online learning and collaboration tools (Notion, Trello)

Students develop both analytical and communication skills essential for the modern workplace.

• Feedback & Mentorship

- Regular feedback is provided on assignments, visualizations, and group projects.
- Peer reviews and reflection sessions help students improve communication and presentation skills.
- Faculty mentorship supports students in aligning analytics skills with career goals.

• Assessments and Evaluation

Assessment is multifaceted to capture both conceptual understanding and practical application:

- Quizzes and Concept Checks after each module
- Tool-based assignments on Excel/Tableau
- Mini-projects and Presentations involving real-world data scenarios
- Group reports and storytelling sessions as summative assessments

Textbooks & References

- James R. Evans – *Business Analytics*
- Foster Provost & Tom Fawcett – *Data Science for Business*
- Nitin Kale & Nancy Jones – *Practical Analytics*
- Tableau Public Gallery – https://public.tableau.com
- Microsoft Learn – Power BI & Excel tutorials

Introduction to Design Thinking & Prototyping

Program Name:	B.Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Design Thinking & Prototyping		1-0-2	2
Type of Course:	SEC		

Course Description: This course equips first-semester engineering students with critical problem-solving skills using Design Thinking methodology. Through hands-on studio and lab sessions, students will apply iterative prototyping, user-centered design principles, and digital tools to create meaningful solutions addressing real-world problems on campus.

COs	Statements
CO 1	Identify user needs and frame design challenges through empathy-driven research.
CO 2	Generate multiple innovative solutions utilizing creative ideation techniques.
CO 3	Develop functional digital prototypes using Figma.
CO 4	Conduct usability evaluations and effectively analyze feedback.
CO 5	Present and communicate design ideas persuasively, demonstrating iterative improvement.

Course Outline

Unit Number: 1	Title: *Understanding Users and Problem Framing*	No. of hours: 8
Content:		
• Foundations of Design Thinking (Definition, need, relevance to computing) • Empathy Methods: interviews (types), observation (direct/contextual), empathy maps • Defining User Personas (goals, frustrations, motivations) • Formulating problem statements using POV and HMW techniques • Lab & Studio Activities: • Analyze the IDEO Shopping Cart case • Conduct user interviews (2–3 users per team) • Create empathy maps and personas from real observations • Frame HMW questions for selected problem areas		
Unit Number: 2	**Title:** ***Creative Ideation and Low-Fidelity Prototypes***	**No. of hours: 8**
Content:		
• Ideation frameworks (SCAMPER, mind mapping) • Principles of brainstorming and idea selection • Paper prototyping: elements of layout and navigation • Structuring feedback via peer reviews **Lab & Studio Activities:** • Perform a team mind-mapping session to explore solution space • Apply SCAMPER to generate variant ideas • Sketch UI on paper with detailed user flow • Conduct structured feedback rounds with peer groups		
Unit Number: 3	**Title:** ***Digital Prototyping Using Figma***	**No. of hours: 8**
Content:		

- Introduction to Figma: workspace, frames, components
- Creating interactive mid-fidelity screens (buttons, links, transitions)
- Collaboration tools in Figma (team feedback, versioning basics)
- **Lab & Studio Activities:**
- Figma tool exploration and recreation of simple UI components
- Convert paper sketches into mid-fidelity clickable prototypes
- Prepare mid-sprint review showcasing problem framing to digital mockup

Unit Number: 4	**Title:** ***Usability Testing, Refinement, and Showcase***	**No. of hours: 6**
Content:		

- Basics of usability testing (creating test tasks, observing behavior)
- Heuristic evaluation (Nielsen's heuristics in UI review)
- Core UI/UX concepts: visual hierarchy, color, typography, alignment
- Iterative refinement based on test feedback
- Visual storytelling and presentation tips
- **Lab & Studio Activities:**
- Conduct peer usability testing sessions
- Document and implement iterative improvements
- Prepare pitch decks: journey maps, personas, problem/solution fit
- Final presentation of project prototypes to a review panel

Evaluation Scheme (with Rubrics):

- Studio Work & Participation (20%): Regularity, quality of discussions, in-class assignments (ideation boards, journey maps).
- Midterm Project Pitch (20%): Clarity of problem definition, innovativeness of solutions, depth of empathy research, initial paper prototype.
- Final Prototype (40%): Complexity and interactivity of the digital prototype, quality of iterative documentation.
- Final Presentation & Viva (20%): Effectiveness of presentation, storytelling clarity, response to questions, individual contribution reflection.

Instructor Guidelines:

- Adopt a coaching mindset, encouraging autonomy and creativity.
- Regularly review progress through shared trackers (Google Sheets/Notion).
- Conduct periodic design critiques in the "design studio" format.
- Schedule mid-course retrospectives for adjustments in teaching strategies.

Teaching Resources:

- Core References:
  - Stanford d.school Bootcamp Bootleg
  - IDEO Design Kit (https://www.designkit.org/)
  - Don Norman, The Design of Everyday Things
  - Steve Krug, Don't Make Me Think
- Figma Resources:
  - Figma Education Resources
  - FreeCodeCamp Figma Crash Course (YouTube)

Textbooks

1. The Design Thinking Toolbox: A Guide to Mastering the Most Popular and Valuable Innovation Methods; Michael Lewrick, Patrick Link, Larry Leifer; John Wiley & Sons, Inc., 1st Edition, 2020, **ISBN-13:** 978-1119629191

Software / Tools Required

- **Figma** (Free for students)
- **Balsamiq** (or similar lo-fi wireframing tool)
- **Miro / Jamboard** for collaboration
- **Canva / Notion / Google Docs** for documentation

Learning Experience for Design Thinking & Prototyping

- **Exploratory Lectures:** Students are introduced to the five stages of design thinking—Empathize, Define, Ideate, Prototype, and Test—through real-world examples and case studies across industries. Emphasis is placed on human-centered approaches to problem-solving and innovation.
- **Empathy Exercises:** Through user interviews, observations, and persona development, learners build deep understanding of user needs, behaviors, and pain points. These activities help shape problem statements grounded in real-world insights.
- **Collaborative Ideation:** Students engage in brainstorming sessions, mind mapping, and "How Might We" challenges to generate creative solutions. Visual thinking and divergent-convergent techniques encourage out-of-the-box thinking and teamwork.
- **Hands-On Prototyping:** Learners develop low- and high-fidelity prototypes using paper sketches, wireframes, and digital tools (e.g., Figma, Adobe XD). Emphasis is placed on iterative design and translating abstract ideas into tangible solutions.

- **Testing & Feedback:** Students test prototypes with peers or target users, collect feedback, and refine designs. Reflection and iteration cycles are integral to improving functionality, usability, and desirability.
- **Mini Design Sprints:** Teams work on real or simulated problems to deliver prototypes in a short timeframe, simulating professional design environments and fostering time management and collaboration.
- **Peer Learning & Showcases:** Group reviews, critique sessions, and final project showcases encourage constructive feedback, confidence in presentation, and exposure to diverse design approaches.
- **Assessments & Reflection:** Students are assessed through design journals, process documentation, presentations, and peer evaluations. Reflective practice is encouraged to deepen understanding of creative confidence, user focus, and iterative improvement.

Design Thinking & Prototyping Lab Equipment Requirements

Equipment Name	**Specifications / Description**	**Qty (for 60 students)**
Whiteboards	Mobile whiteboards for ideation, sketching, and mapping user journeys	6 large panels
Sticky Notes (Post-its)	Assorted colors and sizes for brainstorming and affinity mapping	100 pads
Dot Stickers / Markers	For dot voting and ideation ranking	60 sheets
Empathy Mapping Templates	A3 printed templates for group empathy mapping exercises	30 templates
Persona Canvas Sheets	Pre-designed persona sheets for design research	30 templates
User Journey Mapping Sheets	Pre-formatted A2/A1 paper or foam boards	30 sheets or boards
A3/A2 Sketching Pads	For wireframing and paper prototyping	60 pads
Marker Sets (Fine & Bold)	For sketching, mapping, and whiteboard work	30 sets
Scissors / Paper Cutters	For physical prototyping exercises	20 pairs
Glue Sticks / Tape	Standard adhesive tools for low-fidelity physical mockups	30 sets
Prototyping Stationery Kit	Paper, foam board, cardboard sheets, popsicle sticks, straws, etc.	5 shared kits
Digital Drawing Tablets (Optional)	Wacom or XP Pen for students wanting to do digital sketching	5 units (shared)
Laptops / Desktops	With internet access, and pre-installed software (Figma, Balsamiq, Canva, Google tools)	30 systems (or BYOD)
High-Speed Internet	Wi-Fi with access to cloud tools (Figma, Miro, Canva, Google Docs)	Lab-wide
Projector & Screen	For design presentations, walkthroughs, tutorials	1 set
Color Printers (A4/A3)	For printing personas, user journey maps, wireframes	2 printers
Storage Lockers /	For safely storing project materials and supplies	10–12 lockers

Cabinets		
Collaborative Software Licenses	Free/Edu plans for: Figma, Canva, Notion, Miro, Jamboard	As needed
Audio Recorders / Smartphones	For recording user interviews and feedback	10 shared devices or BYOD
Video Recording Setup	For recording final project walkthroughs / pitching sessions	1 basic camera or tripod
Tabletop Presentation Boards	For showcasing final team projects during review	15–20 boards
Flexible Furniture	Movable desks, modular seating, idea zones for team collaboration	Configurable for 10 teams
LED Desk Lamps (Optional)	For close-up design work or usability testing ambiance	10 shared

Software Requirements

Software / Platform	**Purpose**	**License Type**
Figma	UI/UX design and prototyping	Free for students/teams
Balsamiq	Lo-fi wireframing	Free trial / Edu license
Canva	Graphic design, reports, presentation	Free / Edu version
Miro / Jamboard	Real-time collaborative whiteboarding	Free tier for education
Notion	Project documentation, templates	Free for students
Google Docs/Slides	Collaborative writing and presentation	Free

Semester: 2

Basics of Data Structures

Program Name:	B. Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Basics of Data Structures	ETCCDS201	3-0-2	4
Type of Course:	Major		
Pre-requisite(s):	Fundamentals of Programming (preferably in Python, C)		

Course Description: This course delivers a deep theoretical grounding in essential data structures and algorithm design, critical for building efficient computational solutions. Students will explore linear and non-linear structures, algorithmic complexity, and foundational paradigms. Python will be used to illustrate concepts, keeping the emphasis on logic over syntax. The course also supports analytical and placement preparation objectives.

The Course Outcomes (COs)

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

COs	Statements
CO 1	Evaluate time and space complexity for various algorithms.
CO 2	Apply linear data structures in problem-solving.
CO 3	Implement and compare sorting algorithms based on input patterns.
CO 4	Construct and traverse hierarchical and graph-based data structures.

A student is expected to have learned concepts and demonstrated abilities or skills related to programming at the end of the course.

Course Outline

Unit Number: 1	Title: Algorithm Fundamentals & Analysis	No. of hours:15
Content:		
• Introduction to Data Structures & Abstract Data Types (ADTs) • Complexity Analysis: Big O, Theta, Omega; Best, Average, Worst Case • Algorithmic Paradigms: Overview of Divide & Conquer, Greedy, DP • Recursion Principles and the Call Stack		
Unit Number: 2	**Title: Linear Structures**	**No. of hours: 15**
Content:		
• Arrays (1D & 2D), Static vs Dynamic • Linked Lists: Singly, Doubly, Circular • Stack: Concept, Real-world use cases • Queue: Types, Implementation & Applications		
Unit Number: 3	**Title: Sorting Techniques**	**No. of hours:15**
Content:		
• Sorting Categories: Comparison vs Non-Comparison • Elementary Sorts: Bubble, Insertion, Selection • Advanced Sorts: Merge, Quick, Heap • Conceptual: Counting Sort, Radix Sort		
Unit Number: 4	**Title: Trees, Graphs & Hashing**	**No. of hours: 15**
Content:		
• Tree Structures: Binary, BSTs, Traversals • Heap Structures, Priority Queues • Balanced Trees: AVL, Red-Black, B-Trees (Intro only) • Graphs: Representation and Traversals (BFS, DFS) • Hashing: Concepts, Collision Handling • Prefix Trees (Tries), Advanced DS Overview (Bloom Filter, Segment Tree, etc.)		

Learning Experience for Basics of Data Structures

• **Interactive Lectures**: Introduce students to core data structure concepts such as arrays, linked lists, stacks, queues, trees, and graphs through engaging and interactive lectures. These sessions include visual demonstrations, real-life analogies, and short coding snippets in C/C++ or Python to strengthen understanding and maintain interest.

• **Hands-On Labs**: Organize regular lab sessions where students implement and manipulate various data structures manually. These labs help reinforce theoretical knowledge by guiding students through practical tasks such as inserting, deleting, traversing, and searching in different data structures using structured code.

• **Group Projects:** Encourage peer learning through small team-based projects that involve building simple applications or solving algorithmic problems using appropriate data structures. These projects foster collaboration and help students understand the importance of choosing the right structure for a given task.

• **Assignments & Case Studies:** Assign exercises and real-life scenarios that challenge students to apply their knowledge in practical settings—such as managing a waiting list using queues or designing a family tree using linked structures—thus bridging the gap between theory and application.

Textbooks and Reference Books

1. Data Structures & Algorithms in Python; Author: Fabio Neves; Publisher: Packt Publishing; Publication Year: 2021; ISBN-13: 978-1801815170
2. Ellis Horowitz & Sartaj Sahni – Fundamentals of Data Structures

Introduction to Basics of Data Structures Lab

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: Python, CLI, VS Code

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 a 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: Python, CLI tools

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

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: Python, CLI or Jupyter Notebooks

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: Python, CLI, SQLite

Cloud Computing Foundations and Practices

Program Name:	B. Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Cloud Computing Foundations and Practices	ETBCCC202	3-0-2	4
Type of Course:	DSE		
Pre-requisite(s):			

Course Description: Cloud Computing Foundations and Practices introduces core concepts, architecture, service models (IaaS, PaaS, SaaS), and deployment strategies of cloud computing. The course covers virtualization, scalability, security, and hands-on experience with leading cloud platforms, enabling students to design, deploy, and manage cloud-based solutions effectively in real-world environments.

Course Outcomes
After completing this course, students will be able to:

COs	**Statements**
CO 1	Understanding the fundamental principles and service models of cloud computing.
CO 2	Applying virtualization and containerization techniques in practical cloud setups.
CO 3	Deploying and manage applications using IaaS and PaaS services on leading cloud platforms.

CO4	Analyzing cloud security, cost management, and scalability strategies.

Course Outline

Unit Number: 1	Title: Introduction to Cloud Computing	No. of hours: 15
Content Summary: • Cloud Computing Evolution and Characteristics • Cloud Service Models: IaaS, PaaS, SaaS • Cloud Deployment Models: Public, Private, Hybrid • Benefits and Challenges of Cloud Adoption **Real-World Applications**: • Analyze how startups use SaaS products (e.g., Google Workspace) • Case studies of public cloud migrations (e.g., Netflix, Dropbox)		
Unit Number: 2	**Title: Virtualization and Containerization**	**No. of hours: 15**
Content Summary: • Virtual Machines, Hypervisors, and Containers • Comparison: VMs vs Containers • Tools: VMware, VirtualBox, Docker • Container Orchestration: Introduction to Kubernetes **Real-World Applications**: • Use Docker to containerize a legacy app • Simulate multi-tenant applications in Kubernetes		
Unit Number: 3	**Title: Cloud Platforms and Deployment Models**	**No. of hours: 15**
Content Summary: • Introduction to AWS, Azure, GCP • Compute, Storage, Networking Services • Auto-scaling, Load Balancing, Monitoring • CI/CD Pipelines in the Cloud **Real-World Applications**: • Deploy static websites on AWS S3 • Configure virtual machines and app services on Azure		
Unit Number: 4	**Title: Cloud Security and Cost Optimization**	**No. of hours: 15**
Content Summary: • Identity & Access Management (IAM) • Data Encryption and Secure Communication • Cloud Cost Estimation Tools and Budget Alerts • Compliance: GDPR, HIPAA basics **Real-World Applications**: • Set up IAM roles for secure resource access • Estimate costs for a production-ready app on AWS		

Textbooks & References

- Rajkumar Buyya et al. – *Cloud Computing: Principles and Paradigms*
- Thomas Erl – *Cloud Computing: Concepts, Technology & Architecture*
- Amazon Web Services – *AWS Documentation & Whitepapers*
- Kubernetes Official Docs – *Kubernetes by Example*
- Pethuru Raj – *Cloud Computing Strategies*

Learning Experience for Cloud Computing Foundations and Practices

- **Interactive Lectures:** Students are introduced to the fundamentals of cloud computing, including service models (IaaS, PaaS, SaaS), deployment models (public, private, hybrid), and enabling technologies such as virtualization and containerization. Conceptual topics are supported with real-world use cases and discussions on leading cloud platforms like AWS, Azure, and Google Cloud.

- **Hands-On Labs:** Through guided lab sessions, students gain practical experience in configuring virtual machines, deploying applications on the cloud, setting up storage solutions, and managing network resources. Tools such as AWS Educate, Microsoft Azure Lab Services, or GCP Free Tier accounts are used for immersive, cloud-native experimentation.
- **Case Studies & Industry Scenarios:** Students analyze case studies to understand how different industries use cloud computing for scalability, reliability, and cost-efficiency. Scenarios such as disaster recovery, auto-scaling, and cloud migration provide practical context to theoretical concepts.
- **Mini Projects:** Learners work in teams to design, implement, and present cloud-based solutions. Projects might include deploying a web app using PaaS, setting up a CI/CD pipeline, or configuring a cloud-based database and analytics workflow.
- **Group Activities & Peer Learning:** Interactive activities like cloud architecture design challenges, role-play as cloud service providers, and peer reviews foster collaboration, communication, and problem-solving in real-world cloud environments.
- **Assessments & Reflection:** Quizzes, hands-on practical tests, and reflective summaries ensure conceptual clarity and application skills. Continuous feedback is provided to guide student progress and deepen learning.

Cloud Computing Foundations and Practices Lab

LAB TASK 1: Virtualization Lab – Multi-OS Setup
Objective: To create and manage multiple virtual machines and understand isolation.
Real-World Scenario: A developer needs to test an app in different OS environments.
Activities:

- Install VirtualBox and create Ubuntu/Windows VMs
- Configure shared folders, snapshots
- Simulate host-guest networking

Tools: VirtualBox, Vagrant, Ubuntu/Windows ISOs

LAB TASK 2: Containerization Lab – Docker for Web Apps
Objective: To containerize and run a web application using Docker.
Real-World Scenario: A microservice is packaged for portability and CI/CD workflows.
Activities:

- Write Dockerfiles for frontend and backend
- Run containers using Docker Compose
- Push image to DockerHub

Tools: Docker, Docker Compose, GitHub

LAB TASK 3: Cloud Platform Deployment – Static Site on AWS
Objective: To host a responsive website on a public cloud.
Real-World Scenario: A company launches a marketing website hosted globally.
Activities:

- Create AWS S3 buckets
- Upload static files with correct permissions
- Enable CloudFront CDN and custom domains

Tools: AWS Console, CLI, S3, Route53

LAB TASK 4: Secure Cloud App with IAM & Budgeting

Objective: To apply IAM roles and monitor cloud costs for a secure deployment.
Real-World Scenario: A project needs to manage access for multiple teams and control costs.
Activities:

- Create IAM users, roles, and policies
- Configure CloudWatch alerts for CPU usage
- Set billing alerts and budget constraints

Tools: AWS IAM, CloudWatch, AWS Budgets

Capstone Project: Cloud-Based Learning Management System (LMS)
Objective: To design, deploy, and monitor a cloud-native application that supports content delivery, student logins, and assignment submissions.
Real-World Scenario: An institute needs a scalable and secure LMS hosted on the cloud.
Activities:

- Use Docker to containerize LMS services (e.g., content, auth, DB)
- Deploy on AWS EC2 or Azure App Services
- Use S3/GCS for media content, RDS for backend storage
- Configure IAM roles, security groups, auto-scaling, and backup

Tools: Docker, AWS EC2/S3/RDS or Azure App Services, GitHub, Postman, Cloud Monitoring

Responsive and Dynamic Web Design

Program Name:	B. Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Responsive and Dynamic Web Design	ETCCWD203	3-0-2	4
Type of Course:	Major		
Pre-requisite(s):	Basic knowledge of HTML, CSS, and JavaScript		

Course Description: This course guides learners from core web development into modern front-end engineering using ES6+ JavaScript and React. It focuses on building responsive, interactive single-page applications (SPAs) using component-driven architecture. Students will explore hooks, asynchronous patterns, routing, state management, and deployment workflows. The course blends practical labs, project sprints, and a capstone to ensure industry-readiness.

Course Outcomes

After completing this course, students will be able to:

COs	Statements
CO 1	Use advanced JavaScript (ES6+) features including async patterns and modules in real-world applications.
CO 2	Develop responsive UIs using React functional components and hooks.
CO 3	Implement routing, shared state, and API-based workflows in SPAs.
CO 4	Optimize, test, and deploy React apps using modern CI/CD tools.

CO 5	Build and present a professional-grade SPA with best practices in performance and accessibility.

Course Outline

Unit Number: 1	Title: *Modern JavaScript Essentials*	No. of hours: 15
Content:		
• ES6+ syntax: let, const, destructuring, template literals, spread/rest operators. • Functional programming: arrow functions, closures, currying, higher-order functions. • Asynchronous programming: Promises, async/await, fetch, AbortController. • Modules and tooling: ES Modules, Vite basics, ESLint, Prettier. • JS Testing: Introduction to Jest, mocking techniques.		
Unit Number: 2	**Title:** ***React Core Concepts***	**No. of hours: 15**
Content:		
• Setting up projects with Vite or Create-React-App. • JSX syntax, component rendering, props, and state. • Core React Hooks: useState, useEffect, useRef, useMemo. • Styling strategies: CSS Modules, Styled Components, Tailwind CSS. • Component composition and prop drilling alternatives.		
Unit Number: 3	**Title:** ***Routing, State, and API Integration***	**No. of hours: 15**
Content:		
• Routing with React Router v6: nested, dynamic, and protected routes. • Global state: Context API with useReducer. • Creating and using custom hooks. • API integration using React Query or SWR. • Authentication basics using JWT and react-hook-form.		
Unit Number: 4	**Title:** ***Optimization, Testing & Deployment***	**No. of hours: 15**
Content:		
• Profiling with React DevTools and performance insights. • Memoization: React.memo, useMemo, useCallback. • Code-splitting and lazy loading with React.lazy. • Accessibility and testing: ARIA roles, Lighthouse, React Testing Library. • CI/CD: GitHub Actions, deployment with Netlify/Vercel, basic PWA setup.		

Learning Experience for Responsive and Dynamic Web Design

- **Interactive Lectures:** Introduce students to the foundational principles of responsive and dynamic web design through interactive lectures. Topics include HTML5, CSS3, JavaScript, and modern frameworks such as Bootstrap. Live demonstrations and browser-based examples help students visualize how responsive layouts adapt to different devices and resolutions.

- **Hands-On Labs**: Conduct intensive lab sessions where students design and develop responsive web pages. Activities include creating flexible grid layouts, applying media queries, and integrating interactive JavaScript components. These labs reinforce theoretical knowledge through step-by-step coding exercises in real-time.

- **Group Projects:** Encourage collaboration by assigning group projects where students design and develop fully responsive and dynamic websites. Projects involve user interface (UI) planning, mobile-first design, interactivity using JavaScript, and the use of frontend libraries. These projects promote teamwork, creativity, and practical problem-solving.

- **Assignments & Case Studies**: Provide students with real-world scenarios that involve analyzing poorly designed websites and improving their responsiveness and user experience. Assignments may include redesign tasks, responsive layout creation, and performance testing on different devices.

Textbooks & References

Learning React: Modern Patterns for Developing React Apps; Alex Banks & Eve Porcello; O'Reilly Media; 5th Edition, 2024 ; ISBN-13: 978-1098132924

Responsive and Dynamic Web Design Lab

Lab Experiments

Lab 1 – ES6+ Portfolio Generator CLI

- Build a Node.js CLI tool to generate a static HTML resume using JSON.
- Use async/await, modules, and command-line arguments for themes.
- Write unit tests using Jest and push to GitHub.

Lab 2 – Personal Finance Tracker (React Core UI)

- Set up a new React project with Vite.
- Design dashboard, modals, and transaction lists with functional components.
- Use useState for UI logic and persist data with localStorage.
- Style with Tailwind CSS or CSS Modules.

Lab 3 – Real-Time Chat App with Routing & Context

- Implement multi-page routing with React Router v6.
- Use Context API and useReducer to manage user and message state.
- Simulate chat with a custom WebSocket hook (polling).
- Secure routes with React-Hook-Form-based login.

Lab 4 – E-Commerce SPA Performance Audit

- Improve performance of an e-commerce SPA using lazy loading and memoization.
- Add accessibility tests and core unit tests for cart logic.
- Set up GitHub Actions to run automated tests and deploy to Netlify.

Lab 5 – Capstone Project: Smart-City Events Dashboard

- Use GitHub Projects to manage tasks and plan features.
- Fetch public events data using API and cache using React Query.
- Build filters, map view, and save favorites with Context.
- Optimize app with PWA support and deploy to Vercel.
- Ensure Lighthouse scores ≥ 90 for performance and accessibility

Verbal Ability

Program Name:	B. Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Verbal Ability		2-0-0	2
Type of Course:	AEC		

Course Description: 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.

Course Outcomes

After completing this course, students will be able to:

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

Course Outline

Unit Number: 1	**Title:** ***Vocabulary Development and Application***	**No. of hours: 8**
Content:		
• Understanding the concept of root words, Prefix and suffix, Ways to enhance Vocabulary, Crosswords and word quizzes, Confusing words, One word substitution, Odd one out, Synonyms and Antonyms, Commonly misspelt words, Idioms and Phrases.		
Unit Number: 2	**Title:** ***Fundamentals of Grammar and Sentence Structure***	**No. of hours: 8**
Content:		
Introduction to Parts of Speech, Tenses and its 'rules, Sentences (Simple, Compound and Complex), Subject Verb Agreement, Pronoun Antecedent agreement, Phrases and Clauses.		
Unit Number: 3	**Title:** ***Mastering Sentence Accuracy and Completion Skills***	**No. of hours: 8**
Content:		
• Spot the error (grammatical errors in a sentence), Sentence Correction (Improvement of sentences based on Grammar rules), Sentence Completion, Cloze Tests		
Unit Number: 4	**Title: Enhancing Sentence Structure and Reading Comprehension**	**No. of hours: 6**
Content:		
• Logical Arrangement of Sentences, Comprehending passages, Contextual questions, Anagrams, Analogies		

Learning Experience for Verbal Ability

- **Interactive Lectures**: Each unit begins with interactive lectures to explain core concepts such as vocabulary building, grammar rules, and sentence structures. Real-life examples, storytelling techniques, and classroom discussions are used to make abstract language concepts more relatable and engaging.
- **Hands-On Activities**: Students participate in vocabulary enrichment exercises like word-building games, crosswords, and quizzes. These activities help learners understand the application of root words, prefixes, suffixes, and commonly confused words in a fun and memorable way.
- **Practice Worksheets & Online Tools:** Regular worksheets and digital tools are provided for grammar drills, sentence corrections, and cloze tests. These enable consistent practice

and reinforce learning of sentence construction, error spotting, and grammar rules.

- **Reading Comprehension Labs**: Special sessions focus on comprehension strategies where students practice reading passages followed by analytical and contextual questions. Logical arrangement and analogy-based tasks are included to sharpen critical thinking.
- **Peer Review & Group Activities:** Group discussions and collaborative editing tasks are introduced for mastering sentence correction and error identification. Peer feedback encourages active learning and improves sentence precision.
- **Assessments & Feedback**: Unit-wise quizzes, online tests, and verbal reasoning assignments are conducted to assess vocabulary development, grammar accuracy, and reading comprehension. Immediate feedback is provided to support continuous improvement.

Textbooks & References

1. Norman Lewis – Word Power Made Easy
2. Wren & Martin – High School English Grammar & Composition
3. R.S. Agarwal & Vikas Agarwal – Quick Learning Objective General English
4. S.P. Bakshi - Objective General English

.

Operating Systems Fundamentals

Program Name:	B. Sc. Computer Science and Engineering		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Basics of Operating Systems	ETCCOS204	3-0-0	3
Type of Course:	Major		
Pre-requisite(s):	None		

Course Description: This course introduces operating systems, covering fundamental concepts such as process management, CPU scheduling, memory management, and file systems. The course is divided into 4 units:

1. Introduction to Operating Systems, Process and CPU Scheduling
2. Threads, Synchronization, Deadlock and Memory Management
3. Virtual Memory, Device Management and Secondary Storage Structure
4. File-System Interface, Implementation and Security

The Course Outcomes (COs)

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

COs	Statements
CO 1	Understand the role and structure of an operating system.
CO 2	Analyze process management and synchronization mechanisms.
CO 3	Evaluate memory management techniques including paging and segmentation.
CO 4	Examine file systems and input/output management in modern OS.

A student is expected to have learned concepts and demonstrated abilities or skills related to operating systems at the end of the course.

Course Outline

Unit Number: 1	Title: Introduction to Operating Systems	No. of hours: 12
Content:		
• Functions and types of Operating Systems • OS structure: Monolithic, Microkernel, Layered • System calls and OS interface • Boot process and OS services **Hands-On Activity**: • Trace Linux/Windows system boot process using documentation and system logs • Write a short report on types of OS used in various devices (e.g., smartphone, router, PC)		
Unit Number: 2	**Title: Process and Thread Management**	**No. of hours: 11**
Content:		
• Process concepts: PCB, states, transitions • Thread models: user-level vs kernel-level • CPU scheduling algorithms: FCFS, SJF, RR, Priority • Inter-process communication, deadlocks, and handling strategies **Hands-On Activity**: • Simulate Round Robin scheduling using a Python/C++ program • Analyze process states using ps, top, or Task Manager in real systems		
Unit Number: 3	**Title: Memory Management**	**No. of hours: 10**
Content:		
• Contiguous and non-contiguous allocation • Paging and segmentation • Virtual memory, demand paging, page replacement algorithms (FIFO, LRU) • TLB, Thrashing, and memory hierarchy **Hands-On Activity**: • Create a simulation for page replacement strategies • Compare real vs simulated memory usage using OS monitoring tools		
Unit Number: 4	**Title: File Systems and I/O Management**	**No. of hours: 12**
Content:		

- File concepts, directory structures, access methods
- File allocation strategies: contiguous, linked, indexed
- Disk scheduling algorithms: FCFS, SSTF, SCAN
- I/O hardware, buffering, device drivers
  Hands-On Activity:
- Use terminal commands to explore file metadata and file permissions
- Design a disk scheduling visualizer using FCFS and SCAN algorithms

Learning Experience for Operating System Fundamentals

- **Interactive Lectures:** Introduce students to core operating system concepts, such as process management, memory management, and file systems, through interactive lectures. These sessions are designed to facilitate understanding by including real- world examples, case studies, and problem-solving activities.
- **Hands-On Labs:** Provide practical lab sessions where students implement and experiment with operating system concepts like CPU scheduling, memory management techniques, and file-system operations. These labs help reinforce theoretical knowledge through hands-on experience with operating system simulators or Linux- based environments.
- **Group Projects:** Encourage collaborative learning by assigning group projects that require students to analyze, design, and implement operating system components, such as a simple scheduler or memory manager. These projects simulate real-world scenarios and promote teamwork and problem-solving skills.
- **Assignments & Case Studies:** Assign regular homework and case studies that challenge students to apply operating system concepts to solve complex problems, such as optimizing CPU scheduling or improving file-system security. Case studies focus on practical applications and help students understand the impact of different operating system strategies.
- **Support & Feedback:** Offer continuous support through office hours, online fo- rums, and peer-assisted learning groups. Provide detailed feedback on assignments, lab exercises, and projects to help students refine their understanding and skills related to operating systems.
- **Assessments:** Include quizzes, mid-term exams, and final exams that evaluate students' comprehension of operating system concepts, their ability to apply these concepts to practical scenarios, and their proficiency in analyzing and optimizing operating system performance.

Textbooks

- "Operating System Concepts" by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne
- "Modern Operating Systems" by Andrew S. Tanenbaum

Reference Books

- "Operating Systems: Internals and Design Principles" by William Stallings
- "Operating Systems: A Design-Oriented Approach" by Charles Crowley

Operating Systems Fundamentals Lab

LAB TASK 1: Process Scheduling Simulator – Classroom Time Allocation

Objective: Simulate CPU scheduling techniques using real-life class time allocation. **Scenario**: A school administrator wants to test scheduling strategies for assigning lecture slots.

Activities:

- Implement FCFS, SJF, and RR in code or spreadsheet.
- Compare average turnaround and waiting times.
- Present results graphically and interpret.

Learning Focus: CPU scheduling, turnaround time, algorithm efficiency **Tools**: Python/C++/Excel

LAB TASK 2: Page Replacement Algorithms – Memory Paging Demonstrator

Objective: To model memory paging and simulate page replacement policies. **Scenario**: A memory management unit decides which page to swap during execution.

Activities:

- Simulate FIFO, LRU, Optimal replacement.
- Count and compare page faults.
- Use visual tables to show page frame status.

Learning Focus: Virtual memory, paging, performance optimization **Tools**: Python/C++ or Jupyter Notebooks

LAB TASK 3: File Organization Visualization – File Explorer Clone

Objective: To explore file systems and implement directory hierarchy. **Scenario**: Students organize academic resources into a virtual file structure.

Activities:

- Create tree-based representation of files and directories.
- Implement file add/delete/search and path resolution.
- Visualize with indentation or text-based UI.

Learning Focus: File system design, hierarchy, path resolution
Tools: Python/C++, command-line interface

LAB TASK 4: Disk Scheduling Visualizer – Elevator Algorithm Simulation

Objective: To simulate disk scheduling for optimizing seek time.
Scenario: An elevator (disk arm) must serve floor requests efficiently.

Activities:

- Implement FCFS, SSTF, and SCAN.
- Plot seek sequence and head movement.
- Compare total seek distances and efficiency.

Learning Focus: Disk I/O performance, algorithm analysis
Tools: Python/C++, plotting libraries like matplotlib

CAPSTONE PROJECT: Mini Operating System – Command Line Shell with Process & Memory Management

Objective:
To design and implement a simplified command-line operating environment that supports basic process management, memory simulation, and file handling operations.

Real-World Scenario:
An academic or training OS simulator for beginners, where students build a working shell that mimics fundamental OS tasks like executing commands, managing pseudo-processes, handling memory, and accessing files.

Activities:

- Design a command-line shell supporting built-in commands (e.g., cd, ls, exit, history).
- Implement process spawning using system calls (e.g., fork(), exec(), wait() in C/C++ or Python subprocess).
- Simulate memory allocation using paging or segmentation with basic visualization.
- Include a basic file navigation and manipulation module (open, read, write, delete).
- Create logs for memory usage, process list, and system calls executed.
- Optional: Include a GUI dashboard to visualize processes and memory blocks.

Learning Focus:

- Shell programming, system calls, and process lifecycle
- Virtual memory models and allocation techniques
- File system navigation and I/O handling
- OS modular design and CLI interface development

Tools:

- **Languages**: C/C++, Python
- **Environment**: Linux Terminal, GCC/G++, Python3, Bash
- **Optional**: Tkinter/Flask (for dashboard), VS Code, GitHub for versioning

Minor Project-I

COURSE NAME: Minor Project-I	COURSE CODE	L-T-P	CREDITS
	ETCCPR205	0-0-0	2
TYPE OF COURSE:	Project		
PRE-REQUISITE(S):			

Introduction:
The objective of Minor Project-I for the BSc.(CS)- (Bachelor of Science in Computer Science) 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:

Standard Operating Procedure (SOP)
Minor Project – I (2 Credits, 12-14 Weeks)
BSc.(CS) - Bachelor of Science in computer Science
2 Purpose
Minor Project-I 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 BSc. students registered for Minor Project-I.**
- **Excludes: Implementation-heavy Minor Project-II (covered by a separate SOP).**
- **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-I

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, manage change-requests	Admin dashboard, deadline override
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-	Inline feedback

		Matrix	
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

Principles of Algorithm Design and Analysis

Program Name:	B.Tech Computer Science and Engineering		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Principles of Algorithm Design and Analysis	ETCCDA301	3-0-2	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Description: This course introduces foundational techniques for designing and analyzing efficient algorithms. It covers time and space complexity, paradigms such as divide and conquer, greedy, dynamic programming, and graph algorithms. Students will learn to write optimized algorithms and assess their performance using Big O notation.

The Course Outcomes (COs)

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

COs	Statements
CO 1	Analyze time and space complexity using Big O, Ω, and Θ notations.
CO 2	Apply algorithmic paradigms such as divide-and-conquer, greedy, and dynamic programming.
CO 3	Solve real-world problems using graph algorithms and optimization techniques.
CO 4	Implement, compare, and validate algorithmic solutions through practical tasks.

Course Outline

Unit Number: 1	Title: Algorithm Analysis & Recurrence Relations	No. of hours: 15
Content:		
• Introduction to Algorithms and Flow of Execution • Growth of Functions: Asymptotic Notations (Big-O, Ω, Θ) • Solving Recurrence Relations: Substitution, Recursion Tree, Master Theorem • Worst Case, Best Case, Average Case Analysis • **Hands-On**: • Analyze time complexity of basic sorting/searching algorithms • Implement and compare recursive vs iterative functions		
Unit Number: 2	**Title: Divide & Conquer and Greedy Algorithms**	**No. of hours: 15**
Content:		
• Divide and Conquer: Merge Sort, Quick Sort, Binary Search • Greedy Algorithms: Activity Selection, Huffman Encoding, Fractional Knapsack • Design vs Correctness: Greedy Choice Property, Optimal Substructure • **Hands-On**: • Implement Merge Sort and Quick Sort with comparison • Simulate greedy-based encoding like Huffman Tree		
Unit Number: 3	**Title: Dynamic Programming and Backtracking**	**No. of hours: 15**
Content:		

• Overlapping Subproblems & Memoization • 0/1 Knapsack, Matrix Chain Multiplication, Longest Common Subsequence • Backtracking: N-Queens, Sudoku Solver, Subset Sum **Hands-On**: • Apply bottom-up DP to solve LCS or Knapsack • Design a Sudoku Solver using backtracking		
Unit Number: 4	**Title: Graph Algorithms & NP-Completeness**	**No. of hours: 15**
Content:		
• Graph Representations (Adjacency Matrix/List) • BFS, DFS, Dijkstra's, Kruskal's, Prim's • Intro to NP, P vs NP, Polynomial Reduction, NP-Hard & NP-Complete **Hands-On**: • Implement BFS, DFS and Dijkstra on campus map • Simulate Kruskal's algorithm with edge weights input from a file •		

Learning Experience for Concepts of Principles of Algorithm Design and Analysis

- **Interactive Lectures**: Core concepts such as time and space complexity, asymptotic notations, and algorithm design paradigms (Divide and Conquer, Greedy, Dynamic Programming, etc.) are introduced through concept-driven lectures. Visual aids and step-by-step walkthroughs of classic algorithms (e.g., Merge Sort, Dijkstra's Algorithm) help students understand the logic behind each method.
- **Hands-On Coding Labs:** Practical implementation of algorithms is emphasized through lab sessions using programming languages like C++/Python. Students write, test, and optimize algorithms, developing a deeper understanding of performance trade-offs and computational efficiency.
- **Problem-Solving Workshops:** Regular workshops provide focused practice on solving algorithmic problems from competitive programming platforms and coding contests. Students are encouraged to think critically and select the most appropriate algorithmic approach for each scenario.
- **Mini Projects & Case Studies**: Students work on real-world inspired mini projects where they apply algorithm design principles to solve practical problems. These activities foster creativity and highlight the importance of algorithm selection in performance-critical applications.
- **Assignments & Peer Discussions**: Structured assignments and peer-led discussions promote collaborative learning and help clarify complex topics like NP-completeness, backtracking, and amortized analysis through peer explanation and debate.
- **Formative Assessments**: Quizzes, online coding tasks, and viva sessions are used to evaluate

student understanding continuously. Feedback is shared promptly to guide improvement and strengthen conceptual clarity.

Textbooks and Reference Books

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.
2. Skiena, S. (2020). *The Algorithm Design Manual* (3rd ed.). Springer.
3. Narasimha Karumanchi – Data Structures and Algorithms Made Easy

Principles of Algorithm Design and Analysis Lab

LAB TASK 1: Sorting Techniques – Performance Comparison Tool

Objective: To implement and benchmark various sorting algorithms.
Activities:

- Implement Bubble, Insertion, Merge, and Quick Sort
- Measure and graph time complexity on large datasets
- Analyze recursive depth and space usage
  Tools: Python/C++, Matplotlib

LAB TASK 2: Huffman Encoding Simulator – Greedy Compression Tool

Objective: To simulate text compression using Huffman Coding.
Activities:

- Create character frequency map
- Build min-heap and Huffman Tree
- Generate and decode binary representations

Tools: C++/Python, CLI or GUI

LAB TASK 3: Dynamic Programming Toolkit – Knapsack and LCS

Objective: To build reusable DP modules.
Activities:

- Implement 0/1 Knapsack using tabulation
- Solve Longest Common Subsequence using memoization
- Compare performance of top-down vs bottom-up

Tools: Python, Jupyter Notebook

LAB TASK 4: Graph Algorithm Visualizer – Campus Path Finder

Objective: To simulate pathfinding over real-world map inputs.
Activities:

- Parse adjacency list input from file
- Implement BFS/DFS for connectivity
- Apply Dijkstra's for optimal path
  Tools: Python (NetworkX), Graphviz

Capstone Project: Route Optimization Engine for Delivery Network

Objective: To design a complete system that computes optimized delivery routes using a mix of greedy, DP, and graph techniques.

Real-World Scenario: A logistics company requires a system that schedules routes for deliveries based on distance, time, and vehicle constraints.

Activities:

- Represent delivery network using graph structures
- Apply Dijkstra's and shortest path heuristics
- Optimize batch delivery using 0/1 Knapsack
- Visualize routes and report fuel/time savings

Learning Focus:

- Algorithmic problem modeling
- Cross-algorithm integration
- Performance vs accuracy trade-offs

Tools: Python/C++, Graph Libraries, Dashboards (Tkinter/Flask)

Building Scalable Web Backends

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Building Scalable Web Backends	ETCCWB302	3-0-2	4
Type of Course:	Major		

Defined Course Outcomes

COs	Statements
CO 1	Understand and apply RESTful API principles using Node.js and Express.
CO 2	Integrate databases (SQL/NoSQL) to support data-driven web backends.
CO 3	Design scalable systems with authentication, middleware, and modular routing.
CO 4	Deploy, monitor, and maintain backends on cloud platforms with CI/CD workflows.

Course Outline

Unit Number: 1	**Title:** ***Introduction to Web Backend Architecture***	**No. of hours: 15**
Content:		
• Monolithic vs Microservice Architectures • RESTful APIs and HTTP Methods • Overview of Node.js, Express.js Framework • Middleware, Routing, and Modularization • Hands-On: • Build a basic Express server with routing and middleware • Return JSON from a RESTful API endpoint		
Unit Number: 2	**Title:** ***Database Integration***	**No. of hours: 15**
Content:		
• SQL vs NoSQL databases • MongoDB Schema Design & Mongoose		

• Relational DB integration using PostgreSQL/Knex • CRUD operations and indexing 🛠 **Hands-On**: • Connect backend to MongoDB Atlas • Design models and handle database CRUD APIs		
Unit Number: 3	**Title:** ***Backend Security & Authentication***	**No. of hours: 15**
Content:		
• Authentication vs Authorization • JWT Tokens, Passport.js • Environment Variables and Config Management • Input Validation, Error Handling, Rate Limiting **Hands-On:** • Implement JWT-based authentication • Apply middleware for protected routes and validation		
Unit Number: 4	**Title: Scalability, Deployment & Monitoring**	**No. of hours: 15**
Content:		
• Load Balancing, Caching with Redis • Rate Limiting and Throttling • Deploying with Render/Vercel/Heroku • Logging, Monitoring with Postman/NewRelic **Hands-On:** • Set up backend CI/CD pipeline (GitHub + Render) • Add Redis cache and simulate high traffic conditions		

Text and Reference Books

1. **Ethan Brown** – *Web Development with Node and Express*
2. **Alex Banks & Eve Porcello** – *Learning Node.js*
3. **Brad Traversy** – *Backend Web Development: Modern JavaScript APIs*
4. **Kyle Simpson** – *You Don't Know JS*
5. **MongoDB University & PostgreSQL Docs**

Building Scalable Web Backends Lab

LAB TASK 1: REST API for Book Library
Objective: Design a modular Express.js backend to manage books.
Real-World Scenario: Library system manages book data, search, and CRUD operations.
Activities:

- Set up routes, controllers, and middleware
- Use MongoDB to persist book entries
- Implement query filters and error handling
  Tools: Node.js, Express, MongoDB, Postman

LAB TASK 2: User Authentication System with JWT
Objective: Build a secure login/register system using JSON Web Tokens.
Real-World Scenario: A multi-user blog requires protected user routes and session validation.
Activities:

- Create user model with password hashing
- Add login/logout/token refresh routes
- Apply access control to protected endpoints
  Tools: Node.js, MongoDB, bcrypt, JWT

LAB TASK 3: Task Manager with PostgreSQL
Objective: Integrate relational database in a task-tracking application.
Real-World Scenario: Teams use a dashboard to track and manage tasks with relational integrity.
Activities:

- Create tasks and user relations
- Implement status updates, deadlines, sorting
- Connect using Knex and apply SQL joins
  Tools: Express, PostgreSQL, Knex.js, pgAdmin

LAB TASK 4: Performance Optimization & Redis Caching
Objective: Optimize backend performance with caching and request limits.
Real-World Scenario: A news website needs to reduce redundant DB queries under traffic spikes.
Activities:

- Implement Redis caching for GET requests
- Add API rate limiting using middleware
- Compare response times before and after caching
  Tools: Redis, Express, Postman, Load Test Tools

Capstone Project: Scalable Event Management Backend
Objective: To build a production-grade backend for managing event registration, schedules, and attendees.
Real-World Scenario: A university tech fest platform allows users to browse events, register, receive notifications, and view schedules.
Activities:

- Create RESTful APIs for events, users, and schedules
- Implement JWT authentication and access roles (admin, participant)
- Use MongoDB or PostgreSQL for persistent storage
- Add caching, rate limiting, and deployment with CI/CD
- Integrate Swagger for API documentation

Learning Focus: REST API structure, multi-user handling, database integration, scalability
Tools: Node.js, Express.js, MongoDB/PostgreSQL, Redis, Render, GitHub Actions

Machine Learning: Core Concepts and Techniques

Program Name:	B. Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Machine Learning: Core Concepts and Techniques	ETBCML303	3-0-2	4
Type of Course:	DSE		
Pre-requisite(s):			

Course Description: This course introduces core concepts and techniques of Machine Learning, including supervised, unsupervised, and reinforcement learning. Students will explore algorithms such as decision trees, SVMs, neural networks, and clustering methods, gaining hands-on experience through real-world applications and projects. Emphasis is placed on model evaluation, tuning, and ethical considerations.

Course Outcomes

After completing this course, students will be able to:

COs	Statements
CO 1	Understanding the mathematical foundations and core principles of machine learning.
CO 2	Applying supervised and unsupervised algorithms to real-world problems.
CO 3	Evaluating model performance and improving accuracy using tuning techniques.
CO4	Implementing and deploying end-to-end ML pipelines using popular libraries.

Course Outline

Unit Number: 1	Title: Introduction and Foundations of Machine Learning	No. of hours: 15
Content Summary: • Overview of ML, AI, and Data Science • Types of ML: Supervised, Unsupervised, Reinforcement • Steps in a Machine Learning Workflow • Linear Algebra, Probability, and Statistics for ML **Real-World Applications**: • Analyze ML use in spam detection and recommendation engines • Discuss Netflix and Spotify personalization systems		
Unit Number: 2	**Title: Supervised Learning Techniques**	**No. of hours: 15**
Content Summary: • Linear Regression and Classification • Logistic Regression, k-NN • Decision Trees, Random Forests • Evaluation Metrics: Accuracy, Precision, Recall, F1 Score **Real-World Applications**: • Predict housing prices using linear regression • Build a medical diagnosis system using classification		
Unit Number: 3	**Title: Unsupervised Learning and Model Evaluation**	**No. of hours: 15**
Content Summary: • Clustering: k-Means, Hierarchical • Dimensionality Reduction: PCA • Anomaly Detection Techniques • Cross-Validation, Confusion Matrix, ROC-AUC **Real-World Applications**: • Segment customers using k-Means • Detect outliers in transaction data for fraud detection		
Unit Number: 4	**Title: Introduction to Neural Networks and Model Deployment**	**No. of hours: 15**
Content Summary: • Basics of Neural Networks: Perceptrons, Activation Functions • Introduction to Deep Learning and TensorFlow/PyTorch • Saving, exporting, and deploying ML models • Ethical considerations and bias in AI **Real-World Applications**: • Handwritten digit recognition with ANN • Deploying models using Flask APIs		

Textbooks & References

- Aurélien Géron – *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*
- Tom Mitchell – *Machine Learning*
- Ethem Alpaydin – *Introduction to Machine Learning*
- Trevor Hastie et al. – *The Elements of Statistical Learning*
- Sebastian Raschka – *Python Machine Learning*

Learning Experience for Machine Learning: Core Concepts and Techniques

- **Interactive Lectures:** Core ML concepts such as supervised and unsupervised learning, regression, classification, clustering, and evaluation metrics are introduced through concept-rich lectures. Real-world examples and visual demonstrations (e.g., decision boundaries, tree structures) are used to illustrate abstract ideas.
- **Hands-On Programming Labs:** Students use Python and libraries like Scikit-learn, Pandas, and Matplotlib to implement algorithms such as Linear Regression, KNN, Decision Trees, and K-Means. Lab exercises focus on applying theory to structured and unstructured datasets.
- **Mini Projects and Case Studies:** Students work on real-world datasets (e.g., predicting house prices, spam detection, image clustering) to build and evaluate machine learning models. These projects help bridge the gap between academic concepts and practical applications.
- **Model Evaluation Workshops:** Dedicated sessions help students understand concepts like overfitting, underfitting, cross-validation, confusion matrix, ROC-AUC, and performance tuning using hyperparameters. Students compare model performance and learn how to improve accuracy**.**
- **Peer Learning and Code Reviews:** Students collaborate in teams to debug, optimize, and document ML pipelines. Peer code reviews and group brainstorming foster collaborative learning and reinforce best practices.
- **Quizzes and Feedback Loops:** Formative assessments through weekly quizzes, coding challenges, and in-class discussions ensure continuous learning. Immediate feedback helps learners identify areas of improvement.

Machine Learning: Core Concepts and Techniques Lab

LAB TASK 1: Data Preprocessing and Visualization
Objective: To clean and explore a dataset before feeding it into ML models.
Real-World Scenario: A company wants to analyze customer data for trend prediction.
Activities:

- Handle missing data, outliers, and normalization
- Create plots using seaborn/matplotlib for EDA

Tools: Python, Pandas, Matplotlib, Seaborn, Jupyter Notebook

LAB TASK 2: Predictive Modeling with Supervised Learning
Objective: To build a prediction model using regression/classification algorithms.
Real-World Scenario: Predict student performance or loan approval.
Activities:

- Train/test split and model training
- Hyperparameter tuning using GridSearchCV

Tools: Scikit-learn, Pandas, Jupyter Notebook

LAB TASK 3: Customer Segmentation using Clustering
Objective: To group users based on behavioral attributes.
Real-World Scenario: An e-commerce platform wants to customize offers.
Activities:

- Apply k-Means and visualize clusters
- Elbow method to choose optimal k

Tools: Scikit-learn, Matplotlib, Jupyter

LAB TASK 4: Build and Deploy a Neural Network
Objective: To construct a simple ANN and deploy it using a web framework.
Real-World Scenario: A startup needs an OCR model accessible via API.
Activities:

- Train model on MNIST or CIFAR dataset
- Create Flask API to serve predictions

Tools: TensorFlow/Keras, Flask, Jupyter, Postman

Capstone Project: Machine Learning-Based Career Recommender System
Objective: To build a personalized career recommendation engine based on user skillsets and preferences.
Real-World Scenario: A university wants to guide students in selecting careers based on academic performance and interests.
Activities:

- Collect and preprocess career dataset
- Use classification and clustering to match student profiles
- Evaluate performance using accuracy/F1-score
- Deploy web interface for user interaction

Tools: Python, Scikit-learn, Flask/Streamlit, Jupyter Notebook, GitHub

Communication & Personality Development

Program Name:	B. Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Communication & Personality Development		2-0-0	2
Type of Course:	AEC		
Pre-requisite(s):			

Course Description: 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.

Course Outcomes

After completing this course, students will be able to:

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

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: 8**
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: 8**
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: 6**
Content Summary: Cover Letter, ATS friendly resume, Elevator Pitch, Video Resume (Visume), Networking, Group Discussion, Mock Interviews. Capstone Project		

Textbooks and Reference Books:

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

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

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

Summer Internship-I

Course Name:	Course Code	L-T-P	Credits
Summer Internship-I	ETCCIN304	0-0-0-	2
Type of Course:	INT		

Course Description: 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:

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

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

- **Improve Professional and Soft Skills:**
  - Exhibit improved professional and soft skills, including communication, 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.

COMPETITIVE CODING-I

Department	**School of Engineering Technology**		
Course Name: Competitive Coding-I	**Course Code**	**L-T-P**	**Credits**
		2-0-0	**NIL**
Type of Course:	**Audit/credit Course**		
Prerequisite(s), if any: Fundamentals of programming			

Course Description: 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, the participants will be able to:

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

SESSION WISE DETAILS

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.

Evaluation Criteria

Criteria	Internal (50 marks)
After 2 weeks Coding Test	No. of tests: 6 Marks per test: 5 Total marks: 30
Mid Term Coding Test	20 Marks

External (50 marks)
End Term Paper

Semester: 4

Fundamentals of Java and Object-Oriented Design

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Fundamentals of Java and Object-Oriented Design	ETCCJO401	3-1-0	4
Type of Course:	Major		
Pre-requisite(s):	Basic programming knowledge in any language		

Course Perspective: This course introduces students to the fundamental concepts of Java programming, focusing on object-oriented programming (OOP), inheritance, poly- morphism, exception handling, multithreading, and file handling. The course is divided into 4 units:

1. Introduction to Java and OOP
2. Inheritance and Polymorphism
3. Exception Handling and Multithreading
4. I/O Stream and Collections

The Course Outcomes (COs)

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

COs	Statements
CO 1	Understanding Java syntax, control structures, and core programming constructs.
CO 2	Applying object-oriented principles like encapsulation, inheritance, and polymorphism.
CO 3	Developing Java programs using collections, exception handling, and file I/O.
CO 4	Designing modular, maintainable applications with real-world OOP modeling.

A student is expected to have learned concepts and demonstrated abilities or skills related to Java programming at the end of the course.

Course Outline

Unit Number: 1	Title: Introduction to Java Programming	No. of hours: 15
Content:		
• Java Platform and IDE setup • Variables, Data Types, Operators • Conditional Statements and Loops • Functions and Static Methods **Real-World Applications**: • Build a command-line calculator and basic utility tools • Create pattern printers and simple interest calculators		
Unit Number: 2	**Title: Classes and Object-Oriented Principles**	**No. of hours: 15**
Content:		
• Classes and Objects • Constructors, Method Overloading • this keyword, Access Modifiers • Encapsulation and Abstraction **Real-World Applications**: • Model banking customers or library members using OOP • Design basic student information systems using encapsulation		
Unit Number: 3	**Title: Inheritance, Polymorphism, and Interfaces**	**No. of hours: 15**
Content:		

- Inheritance and super()
- Method Overriding and Dynamic Binding
- Abstract Classes and Interfaces
- Packages and import statements

Real-World Applications:

- Implement employee/payroll management systems with class hierarchies
- Build modular travel booking systems using interfaces

Unit Number: 4	Title: Collections, Exception Handling, and File I/O	No. of hours: 15
Content:		

- Java Collection Framework (List, Set, Map)
- Exception Types, Try-Catch, Custom Exceptions
- File Reading/Writing (BufferedReader/FileWriter)
- Intro to Streams and Lambda Expressions

Real-World Applications:

- Read student data from files and organize using collections
- Catch input validation errors and log them in report files

Learning Experience for Fundamentals of Java and Object-Oriented Design

- **Interactive Lectures:** Leverage visual aids, code demonstrations, and live coding sessions to explain Java concepts such as object-oriented programming (OOP), inheritance, and polymorphism, making abstract concepts more concrete and relatable.
- **Hands-On Labs:** Provide students with lab sessions where they can practice writing and debugging Java code. These labs will focus on implementing Java programs that use OOP principles, exception handling, and multithreading.
- **Group Projects:** Encourage students to work in groups on projects that require them to build small-scale Java applications. These projects will emphasize the practical application of Java concepts like file handling, collections, and GUI development.
- **Assignments & Case Studies:** Assign tasks that require students to solve real-world problems using Java. These assignments will cover various topics such

as inheritance, polymorphism, and data structures within Java.

- **Support & Feedback:** Offer ongoing support through office hours, online dis- cussion boards, and peer review sessions. Provide detailed feedback on coding assignments and projects to help students improve their programming skills.

- **Assessments:** Include quizzes, coding assignments, and exams to evaluate stu- dents' understanding of Java programming concepts, their ability to write efficient code, and their application of OOP principles.

Text and Reference Books

- **Herbert Schildt** – *Java: The Complete Reference*
- **Kathy Sierra & Bert Bates** – *Head First Java*
- **Paul Deitel & Harvey Deitel** – *Java: How to Program*
- **Bruce Eckel** – *Thinking in Java*
- **E. Balagurusamy** – *Programming with Java: A Primer*

Fundamentals of Java and Object-Oriented Design Lab

LAB TASK 1: OOP Model – Banking System
Objective: Apply object-oriented programming principles to simulate banking operations.
Real-World Scenario: A simplified banking software handles account creation, deposit, and withdrawal.
Activities:

- Create classes for Account, Customer
- Implement method overloading and encapsulation
- Use inheritance for different account types

Tools: Java, Eclipse/IntelliJ, CLI

LAB TASK 2: Exception Handling & File I/O – Student Grade Processor
Objective: Read, process, and validate student data from file input.
Real-World Scenario: A result management system processes grades and flags errors.
Activities:

- Parse CSV files and calculate grades
- Handle malformed data with custom exceptions
- Write results into new files

Tools: Java, BufferedReader/FileWriter, Exception Classes

LAB TASK 3: Collections – Inventory Management System
Objective: Use Java Collections to manage and search inventory items.
Real-World Scenario: A small store tracks product availability and pricing.
Activities:

- Use HashMap to store item details

- Search, add, and remove items dynamically
- Sort items using ArrayList and Comparator

Tools: Java Collections, CLI, Scanner

LAB TASK 4: Interface-Based Design – Transport Booking System
Objective: Use interfaces and polymorphism to build a modular booking system.
Real-World Scenario: An app supports multiple modes of transport bookings.
Activities:

- Define common interface Bookable
- Implement classes like Bus, Train, Flight with overridden methods
- Dynamically assign bookings using runtime polymorphism

Tools: Java, Interface Design, Eclipse/IntelliJ

Capstone Project: Online Course Registration System
Objective: Develop a complete object-oriented system for managing courses and registrations.
Real-World Scenario: A university platform allows students to register, drop, and view courses with admin support.
Activities:

- Design class diagrams for Student, Course, Admin
- Implement inheritance for different course types
- Use file I/O for persistent data storage
- Handle exceptions for invalid operations (e.g., duplicate registrations)
- Use ArrayList or HashMap for dynamic collections

Learning Focus: Real-world OOP application, file handling, robust design, modularity
Tools: Java, OOP Design, File I/O, Eclipse/IntelliJ

Introduction to Database Management Systems

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Database Management Systems	ETCCMM402	3-0-2	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Description: This course introduces the fundamentals of Database Management Systems (DBMS), including data models, relational databases, SQL, normalization, and transaction management. Students learn to design, implement, and query databases using SQL and understand how DBMS ensures data integrity, consistency, and security in real-world applications across various domains.

The Course Outcomes (COs)

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

COs	**Statements**

CO 1	Understanding database concepts, architecture, and data models.
CO 2	Designing relational schemas using E-R modeling and normalization techniques.
CO 3	Applying SQL for data definition, manipulation, and transaction control.
CO 4	Implementing database applications with integrity constraints and indexing.

Course Outline

Unit Number: 1	Title: Database Design & Relational Modeling	No. of hours: 15
Content:		
• Introduction to DBMS: Database vs. File Systems, Characteristics, Applications • DBMS vs. RDBMS: E. F. Codd's Rules and their importance in real systems • ER Model: Entities, Attributes, Relationships, Generalization, Specialization, Aggregation • Relational Model: Schema, Primary Key, Foreign Key, Candidate Key, Integrity Constraints • Functional Dependencies and Normalization: 1NF, 2NF, 3NF, BCNF • Star and Snowflake Schemas for analytics and reporting • Tool Demo: ER modeling using Lucidchart or dbdiagram.io, schema creation in MySQL **Industry Use Cases & Practical Focus:** • Use Lucidchart/dbdiagram.io to design ER models • Design a Library Management System and convert ERD into relational schema using MySQL/PostgreSQL • Understand relational data modeling in real-world systems like banking and healthcare		
Unit Number: 2	Title: SQL & Query Operations	No. of hours: 15
Content:		
• SQL Fundamentals: DDL, DML, DCL, TCL Commands • Complex Queries: Joins (Inner, Outer, Self), Subqueries, Set Operations • Views and Indexing: Creation, Modification, Query Optimization using Indexes • Advanced SQL: Window Functions, Common Table Expressions (CTEs) • Tool Demo: Query development using MySQL Workbench or pgAdmin **Industry Use Cases & Practical Focus:** • Execute SQL queries and indexing operations using standard tools • Analyze customer data using advanced SQL from an e-commerce platform • Learn how SQL powers dashboards and data reporting in retail and finance sectors		
Unit Number: 3	Title: Transactions, PL/SQL & Security	No. of hours: 15
Content:		
• **Topics Covered:** • Transactions and Concurrency: ACID Properties, Commit, Rollback, Savepoints • Locking Mechanisms: Isolation Levels, Deadlocks, Optimistic vs. Pessimistic Locking • PL/SQL Programming: Stored Procedures, Functions, Cursors, Triggers • Database Security: User Roles, Privileges, Role-Based Access Control (RBAC), SQL Injection Protection		

• Tool Demo: Programming and testing stored logic in MySQL/PostgreSQL • **Industry Use Cases & Practical Focus:** • Write stored procedures and triggers for secure data operations • Develop a Banking System that handles transactions and fund transfers • Explore how transactional integrity is maintained in e-commerce and stock systems		
Unit Number: 4	**Title: Performance, NoSQL & Big Data**	**No. of hours: 15**
Content:		
• Query Optimization: Execution Plans, B-Tree and Hash Indexing Techniques • NoSQL Databases: Key-Value Stores, Document Stores (e.g., MongoDB, Firebase) • Cloud Databases: AWS RDS, Google BigQuery, Azure SQL Database • Big Data Ecosystem: ETL Processes, Data Lakes vs. Warehouses • Tool Demo: NoSQL schema design using MongoDB Atlas or Firebase **Industry Use Cases & Practical Focus:** • Design and manage NoSQL databases for unstructured data • Build a social media database with user posts and interactions using MongoDB • Understand how NoSQL is used in systems like Netflix, Spotify, and real-time analytics engines		

Learning Experience for Introduction to Database Management Systems

- **Interactive Lectures**: Students are introduced to foundational DBMS concepts such as data models, schema architecture, relational databases, and the role of DBMS in modern applications. Concepts are explained using real-world analogies, diagrams, and whiteboard simulations to simplify abstract topics.
-
- **Hands-On Lab Sessions**: Practical lab work involves using SQL to create and manipulate databases. Students write queries for data definition (DDL), data manipulation (DML), and data control (DCL), and work on joins, subqueries, and constraints to reinforce classroom learning through real-time execution in environments like MySQL or Oracle.
-
- **Conceptual Design Projects**: Students apply ER modeling techniques to design relational schemas from case studies. These mini-projects bridge the gap between theory and practical implementation, promoting an understanding of normalization, integrity constraints, and database design principles.
-
- **Assignments & Case Studies**: Regular assignments include designing ER diagrams, converting them to relational models, and writing optimized SQL queries. Case studies of real-world database applications help students appreciate how DBMS supports business and technical systems.
-
- **Peer Learning & Group Activities**: Group discussions and peer reviews on database design choices help foster collaboration and critical thinking. Activities like query optimization contests or schema critique sessions promote engagement and deeper insight.
-
- **Assessments & Feedback**: Quizzes, lab assessments, and periodic tests ensure continuous evaluation. Prompt feedback is provided to help students correct misconceptions and solidify understanding.

Text and Reference Books

- **Abraham Silberschatz, Henry F. Korth** – *Database System Concepts*
- **Ramez Elmasri, Shamkant Navathe** – *Fundamentals of Database Systems*
- **C. J. Date** – *An Introduction to Database Systems*
- **Raghu Ramakrishnan** – *Database Management Systems*
- **James Groff, Paul Weinberg** – *SQL: The Complete Reference*

Introduction to Database Management Systems Lab

LAB TASK 1: University Record Management using SQL

Objective: Use SQL to manage a relational database for student, faculty, and course records.
Real-World Scenario: A university admin panel manages course assignments, enrollments, and grading.
Activities:

- Create tables with constraints
- Insert and update student/faculty/course data
- Write joins and nested queries

Tools: MySQL/PostgreSQL, DBeaver/PhpMyAdmin

LAB TASK 2: Hospital Database with E-R to Relational Mapping

Objective: Design and map a healthcare E-R model to relational schema and populate it.
Real-World Scenario: A hospital manages patients, doctors, treatments, and billing.
Activities:

- Draw E-R diagram
- Convert to tables and create schema in SQL
- Use JOINs and subqueries to retrieve diagnostic reports

Tools: ERDPlus, SQL, CLI Tools

LAB TASK 3: Transaction and Concurrency Simulation

Objective: Simulate concurrent transactions and integrity control.
Real-World Scenario: Multiple bank users access the same account, causing transaction overlap.
Activities:

- Use SQL transactions with COMMIT, ROLLBACK
- Simulate dirty reads and lost updates
- Demonstrate isolation with locking levels

Tools: MySQL/PostgreSQL, SQL CLI

LAB TASK 4: Normalization and Indexing – Retail Store Database

Objective: Normalize a poorly structured database and create indexes for optimization.
Real-World Scenario: A retail system with redundant customer/order/product data is optimized.
Activities:

- Analyze the unnormalized schema
- Apply 1NF to BCNF
- Add indexing and measure query performance

Tools: SQL DB, Explain Plan, CLI

Capstone Project: Online Bookstore Database System

Objective: Design and implement a complete DBMS backend for an online bookstore.
Real-World Scenario: A bookstore website handles customer signups, book listings, orders, and payments.
Activities:

- Design an E-R model with Customer, Book, Order, Payment entities
- Normalize the schema
- Implement using SQL and connect with a sample front-end
- Test transactions and backup functionality

Learning Focus: Real-world schema design, SQL proficiency, transaction safety, optimization
Tools: MySQL/PostgreSQL, Workbench/DBeaver, Git

Principles of Information Security and Ethical Hacking

Program Name:	B. Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Principles of Information Security and Ethical Hacking	ETCCEH403	3-0-2	4
Type of Course:	DSE		
Pre-requisite(s):			

Course Description: This course introduces the fundamentals of information security, including confidentiality, integrity, and availability. It covers risk management, cryptography, network security, and ethical hacking techniques. Students learn to identify vulnerabilities, assess threats, and implement security measures ethically to protect digital assets and ensure secure computing environments across various platforms.

Course Outcomes

After completing this course, students will be able to:

COs	Statements
CO 1	Understanding the core concepts of cybersecurity, threats, and countermeasures.
CO 2	Analyzing vulnerabilities in systems, networks, and applications.
CO 3	Applying ethical hacking techniques using modern tools in controlled environments.
CO4	Demonstrating security best practices in system hardening and incident response.

Course Outline

Unit Number: 1	Title: Introduction to Information Security	No. of hours: 15
Content Summary: • Information security principles: CIA triad (Confidentiality, Integrity, Availability) • Types of security attacks: Passive, Active, Insider, Outsider • Security policies and risk management • Introduction to cryptography **Real-World Context**: Explore recent security breaches and analyze the root causes and their impact on businesses.		
Unit Number: 2	**Title: Network and System Security**	**No. of hours: 15**
Content Summary: • Network vulnerabilities and protection mechanisms • Firewalls, IDS/IPS, VPNs • Authentication protocols (Kerberos, SSL/TLS) • Secure system configuration and patch management **Real-World Context**: Study how organizations secure enterprise networks using firewall policies and endpoint protection.		
Unit Number: 3	**Title: Ethical Hacking and Penetration Testing**	**No. of hours: 15**
Content Summary: • Phases of ethical hacking: Reconnaissance, Scanning, Gaining Access, Maintaining Access, Covering Tracks • Footprinting, scanning networks, vulnerability assessment • Penetration testing methodology and tools • Legal and ethical considerations **Real-World Context**: Simulate penetration testing against a vulnerable virtual machine in a legal sandbox.		
Unit Number: 4	**Title: Cybercrime, Forensics and Response**	**No. of hours: 15**
Content Summary: • Types of cybercrimes and cyber laws (Indian IT Act, GDPR overview) • Digital forensics: data acquisition, imaging, and evidence handling • Incident response and business continuity planning • Basics of security auditing **Real-World Context**: Examine forensic investigation of a phishing attack that resulted in a data breach.		

Textbooks & References

- William Stallings – *Cryptography and Network Security*
- Charles P. Pfleeger & Shari Lawrence Pfleeger – *Security in Computing*
- Michael T. Goodrich – *Introduction to Computer Security*
- Kevin Mitnick – *The Art of Invisibility*

Learning Experience for Principles of Information Security and Ethical Hacking

- **Interactive Lectures:** Students are introduced to core concepts such as confidentiality, integrity, availability (CIA triad), cryptographic techniques, network vulnerabilities, and the fundamentals of ethical hacking. Real-life security incidents and case discussions are integrated to enhance understanding of risks and defenses.
- **Hands-On Labs:** Practical sessions are conducted in a controlled lab environment where students work with tools like Kali Linux, Wireshark, Metasploit, and Nmap. They simulate attacks (e.g., SQL injection, password cracking, DoS) and practice defensive strategies, reinforcing theoretical knowledge with real-time application.
- **Capture the Flag (CTF) Challenges**: CTF-style activities are used to engage students in problem-solving under time constraints. These challenges develop skills in reconnaissance, exploitation, privilege escalation, and vulnerability analysis.
- **Group Discussions & Ethical Debates:** Students participate in guided discussions on ethical dilemmas, legal issues, cyber law, and responsible disclosure. Debates on gray areas in hacking practices promote critical thinking and ethical reasoning.
- **Case Studies & Projects:** Analysis of famous breaches (e.g., Equifax, SolarWinds) and security frameworks (e.g., NIST, ISO 27001) allows students to connect classroom concepts with real-world implications. Mini projects include creating security policies, penetration testing reports, or simulated threat models.
- **Assessments & Feedback:** Quizzes, lab evaluations, and scenario-based exams assess student learning. Constructive feedback is provided to help them refine both technical and ethical decision-making skills.

Principles of Information Security and Ethical Hacking Lab

LAB TASK 1: Password Security and Hash Cracking
Objective: To explore password protection methods and their vulnerabilities.
Real-World Scenario: An internal audit reveals weak passwords in company systems.
Activities:

- Generate password hashes using MD5, SHA-256
- Perform dictionary and brute-force attacks with tools like John the Ripper
- Analyze password strength policies

Tools: Kali Linux, Hashcat, John the Ripper

LAB TASK 2: Network Scanning and Vulnerability Detection
Objective: To identify active devices and vulnerabilities on a simulated network.
Real-World Scenario: Security team prepares a vulnerability report for an organization.
Activities:

- Perform host discovery using Nmap
- Identify open ports, services, and OS fingerprinting
- Generate vulnerability scan reports

Tools: Nmap, Wireshark, Nessus

LAB TASK 3: Exploitation and Web Application Attacks

Objective: To simulate attacks on web applications and test defenses.
Real-World Scenario: A company tests its web portal for security flaws.
Activities:

- Launch SQL Injection, XSS, CSRF attacks on test apps
- Use Burp Suite for web vulnerability analysis
- Demonstrate mitigation techniques

Tools: OWASP Juice Shop, DVWA, Burp Suite

LAB TASK 4: Digital Forensics and Incident Handling
Objective: To simulate investigation and response to a cybersecurity incident.
Real-World Scenario: Investigating an insider attack in a corporate environment.
Activities:

- Acquire disk image and recover deleted files
- Analyze logs and system registry for evidence
- Write an incident report

Tools: Autopsy, FTK Imager, Log analyzers

Capstone Project: Secure Voting Portal – Design and Audit
Objective: To design and audit a secure online voting system with logging and encryption.
Real-World Scenario: An academic institution needs a secure digital platform for student elections.
Activities:

- Design architecture with secure authentication and session control
- Implement encryption for vote transmission and storage
- Perform penetration testing and generate audit logs
- Simulate vote tampering scenarios and validate detection mechanisms

Tools: Flask/Django, SQLCipher, OpenSSL, Burp Suite, GitHub

Arithmetic and Reasoning

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Arithmetic and Reasoning		2-0-0	2
Type of Course:	AEC		

Course Description: 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 academic and various practical fields.

Defined Course Outcomes

COs	Statements
CO 1	Understanding arithmetic algorithms required for solving mathematical problems.
CO 2	Applying arithmetic algorithms to improve proficiency in calculations.
CO 3	Analyzing cases, scenarios, contexts and variables, and understanding their inter-

	connections in a given problem.
CO 4	Evaluating & deciding approaches and algorithms to solve mathematical & reasoning problems.

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

Course Outline:

Unit Number: 1	**Title: Mathematical Essentials**	**No. of hours: 8**
Content Summary: Unit I: Vedic Maths, Classification of Numbers and Divisibility Rule, Percentage, Ratio and Proportion		
Unit Number: 2	**Title: Fundamentals of Logical Reasoning**	**No. of hours: 8**
Content Summary: Blood Relations, Direction Sense, Coding Decoding		
Unit Number: 3	**Title: Elementary Quantitative Skills**	**No. of hours: 8**
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: Textbooks/Web resources / MOOCs / Magazines/ Journals/ Videos /Podcast etc.

- **https://www.indiabix.com/online-test/aptitude-test/**
- **https://www.geeksforgeeks.org/aptitude-questions-and-answers/**
- **https://www.hitbullseye.com/**

COMPETITIVE CODING -II

Department: SOET	School of Engineering technology		
Course Name: Competitive Coding-II	Course Code	L-T-P	Credits
		2-0-0	NIL
Type of Course:	Audit/credit Course		
Prerequisite(s), if any: Competitive Coding-I, Fundamentals of programming & data structure			

Course Outcomes:

COs	Statements
CO 1	Apply advanced string algorithms to solve complex problems.

CO 2	Analyze and implement efficient linked list operations and complex problem solutions.
CO 3	Evaluate and apply various tree traversal techniques to solve traversal and view-related problems.

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

Evaluation Criteria

Criteria	Internal (50 marks)
After 2 weeks Coding Test	No. of tests: 6 Marks per test: 5 Total marks: 30
Mid Term Coding Test	20 Marks

•

External (50 marks)
End Term Paper

.

Semester -V

Essentials of Software Engineering

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Essentials of Software Engineering	ETCCGA502	3-0-2	4
Type of Course:	Major		
Prerequisites:	Basic programming knowledge and project exposure		

Defined Course Outcomes

COs	**Statements**
CO 1	Understand the software development lifecycle and software engineering principles.
CO 2	Apply modeling techniques for requirements gathering and design.
CO 3	Evaluate and implement software development methodologies.
CO 4	Utilize testing, configuration, and maintenance practices in real-world projects.

Essentials of Software Engineering

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Essentials of Software Engineering	ETCCSE502	3-0-2	4
Type of Course:	Major		
Prerequisites:	Basic programming knowledge and project exposure		

Course Outcomes (COs):

CO	**Description**
CO1	Understanding the software development lifecycle and software engineering principles.
CO2	Applying modelling techniques for requirements gathering and design.
CO3	Evaluating and implement software development methodologies.
CO4	Utilizing testing, configuration, and maintenance practices in real-world projects.

Course Outline

Unit No.	**Title: Software Engineering Overview**	**15**
1	• Introduction to Software Engineering • Software Characteristics, Myths, and Process • Software Development Life Cycle (SDLC) Models • Agile and Scrum Overview Real-World Applications: • Analyze software development practices used in startups vs enterprise systems • Study how SDLC and agile are adopted by companies like Google, Infosys	

Unit No.	**Title: Requirements Engineering and Analysis**	**15**
2	• Requirement Types: Functional and Non-Functional • Requirement Engineering Process • Use Case Diagrams and Scenarios • Software Requirement Specification (SRS) Document Real-World Applications: • Capture and document requirements for an e-commerce or student portal • Model use cases for a mobile banking app	
3	• Design Principles and Modeling • Data Design and Architectural Styles • UML Diagrams: Class, Sequence, Activity • Component and Interface Design Real-World Applications: • Create UML models for healthcare, HR, or ERP systems • Design layered architecture for social networking platforms	15
4	**Title: Testing, Deployment, and Maintenance** • Types of Testing: Unit, Integration, System, Acceptance • Test Case Design and Automation Basics • Configuration Management and Version Control • Software Deployment and Maintenance Practices • Real-World Applications: • Simulate regression testing and bug tracking in ticketing systems • Manage code versions and releases using Git in real-world scenarios	15

Text and Reference Books

- **Roger S. Pressman** – *Software Engineering: A Practitioner's Approach*
- **Ian Sommerville** – *Software Engineering*
- **Pankaj Jalote** – *An Integrated Approach to Software Engineering*
- **Sudarshan S. & Malhotra D.** – *Essentials of Software Engineering*
- **McConnell Steve** – *Code Complete*

Essentials of Software Engineering Lab

LAB TASK 1: Use Case and Requirements Engineering for a Library Management System
Objective: To gather, analyze, and document functional and non-functional requirements.
Real-World Scenario: A college library needs an automated system for cataloging and issuing books.
Activities:

- Conduct stakeholder interviews (simulation)
- Create use case diagrams and SRS
- Model scenarios and constraints

Tools: Draw.io, MS Word/Google Docs, Lucidchart

LAB TASK 2: UML Design Modeling for a Healthcare System
Objective: To visualize and document the design phase of a software system.
Real-World Scenario: A hospital requires software to manage patient records, appointments, and billing.
Activities:

- Create class diagrams, sequence diagrams, and activity diagrams
- Apply design patterns (optional)
- Use CRC cards for role modeling

Tools: StarUML, Visual Paradigm, Lucidchart

LAB TASK 3: Testing and Bug Tracking for a Retail Web Application
Objective: To perform software testing with documentation and defect tracking.
Real-World Scenario: A retail app has been deployed and needs QA testing before production.
Activities:

- Write test cases and test plans
- Perform black-box and white-box testing
- Report bugs using tools

Tools: Selenium (optional), JUnit/PyTest, Bugzilla or GitHub Issues

LAB TASK 4: Version Control and Agile Task Boards
Objective: To simulate agile team collaboration with Git and Kanban tools.
Real-World Scenario: A dev team manages tasks, features, and bugs collaboratively.
Activities:

- Use GitHub/GitLab for version control
- Create branches and merge pull requests
- Use Trello/Jira for task tracking

Tools: Git, GitHub, Trello, VS Code

Capstone Project: Project Management Dashboard (Mini-Jira Clone)
Objective: To design and develop a software system that helps teams manage projects, tasks, and sprints.
Real-World Scenario: A startup needs a custom dashboard for project tracking and developer collaboration.

Activities:

- Define user roles and requirements
- Model system architecture using UML
- Implement user stories and sprints
- Integrate versioning, task status, and reporting

Learning Focus: SDLC implementation, team collaboration, UML design, testing
Tools: HTML/CSS/JS, Node.js or Django/Flask, GitHub, Figma, SQL/SQLite

Practical Applications of Generative AI

Program Name:	B. Sc (Hons.) Computer Science)		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Practical Applications of Generative AI	ETCCSE502	3-0-2	4
Type of Course:	DSE		
Pre-requisite(s):			

Course Description: This course explores the practical applications of Generative AI across domains like text, image, audio, and video generation. Students will gain hands-on experience with leading tools and models, learning to build creative and intelligent systems while addressing ethical, legal, and societal implications of AI-generated content.

Course Outcomes

After completing this course, students will be able to:

COs	**Statements**
CO 1	Understanding the foundational principles and models behind Generative AI.
CO 2	Exploring real-world applications of generative models in media, design, and business.
CO 3	Implementing generative models using modern frameworks and tools.
CO4	Evaluating ethical implications, performance, and limitations of generative AI solutions.

Course Outline

Unit Number: 1	Title: Introduction to Generative AI	No. of hours: 15
Content Summary: • Overview of Generative AI: History, motivation, and growth • Core concepts: Latent space, sampling, creativity through AI • Introduction to generative models: VAEs, GANs, Diffusion Models • Applications in art, design, text, and image generation **Real-World Context**: Explore how brands use AI to generate product prototypes and marketing creatives.		
Unit Number: 2	**Title: Generative AI with Text and Language**	**No. of hours: 15**
Content Summary: • Language Models: GPT, BERT, and Transformer architecture • Text generation, summarization, and translation • Prompt engineering and fine-tuning LLMs • Use cases: Email drafts, chatbot generation, creative writing **Real-World Context**: Deploy an AI writing assistant trained on organizational style guides.		
Unit Number: 3	**Title: Generative AI with Images and Media**	**No. of hours: 15**
Content Summary: • Generative Adversarial Networks (GANs): Concepts and architecture • Diffusion models and image generation tools (e.g., DALL·E, Midjourney) • AI-based audio and video generation • Use cases: Visual storyboarding, synthetic avatars, audio dubbing **Real-World Context**: Design synthetic ad banners and product demo videos using generative AI.		
Unit Number: 4	**Title: Responsible AI and Evaluation**	**No. of hours: 15**
Content Summary: • Evaluation metrics for generative models: BLEU, FID, Inception Score • Ethical concerns: Deepfakes, misinformation, copyright • Bias and fairness in generated content • Tools for detection and watermarking of AI-generated content **Real-World Context**: Study how platforms detect AI-generated misinformation and enforce policy compliance.		

Textbooks & References

- *Deep Learning with Python* – François Chollet
- *Generative Deep Learning* – David Foster
- OpenAI, Google DeepMind, Hugging Face official documentation
- Stanford CS231n, CS50 AI: Lecture videos
- Responsible AI Guidelines – Microsoft, OpenAI, and OECD

Learning Experience for Practical Applications of Generative AI

- **Interactive Lectures**: Students are introduced to the fundamentals of generative AI, including deep learning, GANs (Generative Adversarial Networks), diffusion models, and transformer architectures like GPT. These lectures include real-world use cases such as image generation, text synthesis, music composition, and code generation, promoting curiosity and contextual understanding.
- **Hands-On Lab Sessions**: Learners engage in practical sessions using platforms like Google Colab and tools like Hugging Face, OpenAI APIs, and Stability AI. They experiment with fine-tuning pre-trained models, generating creative content, and building simple generative applications.
- **Mini Projects**: Students undertake projects such as creating AI-generated art, building a chatbot, summarizing long documents, or generating synthetic datasets. These projects promote creativity, technical skill, and real-world application.
- **Case Studies & Ethical Discussions**: Real-world case studies (e.g., DALL·E, ChatGPT, Deepfake detection) are discussed alongside ethical implications such as bias, misinformation, and intellectual property. This encourages critical thinking and responsible AI usage.
- **Collaborative Learning**: Group activities like prompt engineering challenges, hackathons, or model comparisons enable peer learning and enhance understanding of different generative approaches.
- **Continuous Assessment & Feedback**: Learning is reinforced through quizzes, lab reviews, and reflective assignments. Personalized feedback ensures continuous growth and understanding of both theoretical and applied aspects of generative AI.

Practical Applications of Generative AI Lab

LAB TASK 1: Text Generation using GPT APIs
Objective: Build an AI assistant that generates short responses based on user queries.
Real-World Scenario: A customer support tool needs AI-generated draft replies.
Activities:

- Use OpenAI/Gemini API to generate context-based responses
- Experiment with prompt engineering and fine-tuning
- Add sentiment analysis to screen outputs

Tools: Python, OpenAI API, Hugging Face, Streamlit

LAB TASK 2: Image Generation using Diffusion/GAN Models
Objective: Create AI-generated marketing posters for a fictional brand.
Real-World Scenario: A startup uses generative AI for affordable design prototyping.
Activities:

- Generate images using Stable Diffusion or Midjourney
- Use ControlNet or custom prompts for fine-tuned outputs
- Overlay generated images with text and branding

Tools: Python, Hugging Face, Canva, Diffusion Web Tools

LAB TASK 3: Code & Audio Generation with LLMs

Objective: Generate snippets of code and podcast scripts using GenAI tools.
Real-World Scenario: Automate blog post narration and code generation for tutorials.
Activities:

- Use Code LLMs to generate Python/JS functions
- Convert text to audio using ElevenLabs or Google TTS
- Evaluate tone, clarity, and syntactic correctness

Tools: GitHub Copilot, ElevenLabs, OpenAI Codex

LAB TASK 4: Deepfake Detection and Ethics Simulation
Objective: Identify and analyze AI-generated deepfakes.
Real-World Scenario: A news agency wants to ensure authenticity of incoming media.
Activities:

- Compare real vs AI-generated images and videos
- Explore detection tools for deepfakes and watermarking
- Simulate a newsroom policy for AI usage and disclosure

Tools: Deepware, Hugging Face Detector, NewsGuard policies
Capstone Project: GenAI Business Assistant for SMEs
Objective: Build a domain-specific GenAI assistant that helps small businesses with content generation, basic analytics, and branding support.
Real-World Scenario: A small business owner needs an AI tool to create blog posts, product descriptions, and social media content on demand.
Activities:

- Integrate LLM APIs with a simple UI (e.g., Flask or React)
- Use GenAI to generate branded text, visuals, and summaries
- Add analytics module using Python and visual libraries
- Ensure ethical use, branding consistency, and content reliability

Tools: OpenAI/Gemini APIs, Flask/Streamlit, Canva API, MongoDB, Hugging Face

Comprehensive Placement Preparation

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Comprehensive Placement Preparation		2-0-0	2
Type of Course:	AEC		

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Enhance aptitude and reasoning skills to meet industry placement standards.
CO 2	Develop effective communication and interview readiness.
CO 3	Build technical problem-solving ability across core CS subjects.
CO 4	Simulate real-world placement scenarios through mock drills and assessments.

Course Outline

Unit Number: 1	Title: Aptitude and Logical Reasoning for Tech Interviews	No. of hours: NA
Content:		
• Number systems, profit & loss, ratios, percentages • Logical reasoning: puzzles, arrangements, syllogisms • Real-World Activity: *Timed aptitude quizzes simulating company hiring rounds like TCS NQT, Wipro Elite.*		

Unit Number: 2	Title: Verbal Ability and Resume Writing	No. of hours: NA
Content:		
• Grammar, reading comprehension, sentence correction • E-mail and resume writing for industry roles • Real-World Activity: *Peer-reviewed resume creation and Grammarly/Jobscan optimization.* •		

Unit Number: 3	Title: Technical Skill Drills – Data Structures, OS, DBMS, OOP	No. of hours: NA
Content:		
• MCQs and code snippets based on core subjects • Conceptual and application-based revision sessions • Real-World Activity: *Leetcode/GeeksforGeeks contests focused on DSA/OS/DBMS.*		

Unit Number: 4	Title: Interview Readiness and Group Discussions	No. of hours: NA
Content:		
• Personal Interview strategies: HR + Tech round • Group discussions and communication techniques • Real-World Activity: *Mock interviews and G.D. rounds evaluated by alumni/HR professionals.*		

Tools and Platforms

- HackerRank, LeetCode, CodeStudio for technical practice
- Google Docs, Grammarly for resume and email drafting
- MS Teams/Zoom for mock interviews
- Mettl or AMCAT-like simulators for test environments

COMPETITIVE CODING -III

Department: SOET	School of Engineering technology		
Course Name: Competitive Coding-II	Course Code	L-T-P	Credits
		2-0-0	NIL
Type of Course:	Audit/credit Course		
Prerequisite(s), if any: Competitive Coding-I, Fundamentals of programming & data structure			

Course Outcomes:

COs	Statements
CO 1	Apply bit manipulation, number theory, and string algorithms to solve computational problems.
CO 2	Analyze and implement advanced backtracking and recursion techniques for combinatorial problems.
CO 3	Evaluate sliding window techniques and two-pointer algorithms for efficient solutions.
CO4	Solve 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	2
Content Summary: assign cookies, array partition, can place flower, lemonade change		
Session: 10	Greedy-II.	2
Content Summary: Activity selection, minimum platform, coin change		
Session : 11	Greedy-III.	2
Content Summary: max chunk to make sorted, max chunk to make sorted-2, 0/1 knapsack.		
Session: 12	Graph Introduction and representation.	2
Content Summary: Introduction, Representation using adjacency matrix and adjacency list		
Session: 13	2	2
Content Summary: Graph Traversal BFS(Breadth first search) and DFS(Depth first search)		
Session: 14	Graph-III	2
Content Summary : Connected Components, Detecting Cycles in Graphs		
Session: 15	Graph Problems-IV.	2
Content summary: find if path exist(has path), print all path from source to destination, Number of Island		
Session: 16	Advanced Graph.	2
Content summary: Number of Provinces, Flood Fill, Number of closed islands.		
Session: 17	Minimum Spanning Tree algorithms.	2
Content summary: Prim's Algorithm, Kruskal's algorithm.		
Session: 18	Shortest Path Algorithm.	2
Content summary: Dijkstra algorithm, Bellman ford algorithm.		
Session: 19	Summarizing the Semester 5.	2
Content summary: Company specific problems on Graphs, sliding window and recursion.		
Session: 20	Summarizing the Semester 5.	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
  Best for coding interview prep with 189 questions, concepts, and system design.

Evaluation Criteria

Criteria	Internal (50 marks)
After 2 weeks Coding Test	No. of tests: 6 Marks per test: 5 Total marks: 30
Mid Term Coding Test	20 Marks

-

External (50 marks)
End Term Paper

SEMESTER VI

Course_Code	**Category**	**Course Title**	**L**	**T**	**P**	**Credits**
ETCCIN601	INT	Industry Internship	0	0	16	8
ETCCPR602	PROJ	Major Project-II	0	0	8	4
MOOC	MOOC	MOOC (Swayam/ NPTEL/AICTE’s ELIS)	2	0	0	2
			2	0	24	14
	Total Credits=125					