K.R. MANGALAM UNIVERSITY
THE COMPLETE WORLD OF EDUCATION

School of Engineering and Technology

Programme Handbook
(Programme Study & Evaluation Scheme)

Bachelor in Science
(B.Sc. (Hons.) Cyber Security)

Programme Code: 83

THREE YEARS UNDERGRADUATE PROGRAMME

WITH EFFECTIVE FROM 2024-25 SESSION

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

Contents

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

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

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 premier universities, research centers, industries, and professional bodies.
- **Enhance** leadership qualities among the youth with an understanding of ethical values and environmental realities.

About School

The School of Engineering and Technology at K. R. Mangalam University started in 2013 to create a niche of imparting quality education, innovation, entrepreneurship, skill development and creativity. It has 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 a holistic and contemporary learning experience.

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

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

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

We are dedicated to fostering a culture of innovation and entrepreneurship, recognizing these as essential pillars for the success of our students in the rapidly evolving world of technology. We inspire innovation and entrepreneurship through our dynamic Entrepreneurship and Incubation Center, engaging contests like 'MindBenders' ,'HackKRMU,' 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 infrastruc-

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

School Vision & Mission

Vision

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

Mission

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

About the Programme

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 they are programme-specific. There are two to four PSOs for a programme.
- **Programme Educational Objectives (PEOs):** Programme Educational Objectives of a degree programme are the statements that describe the expected achievements of graduates in their career, and what the graduates are expected to perform and achieve during the first few years after graduation.
- **Credit:** Credit refers to a unit 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.

Programme Educational Objectives (PEO)

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

Programme Outcomes (PO)

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 understanding the impact of engineering solutions on the environment and advocating for sustainable development.
- **PO4 Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the 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.

Programme Specific Outcomes (PSO)

- **PSO1:** Understanding the core concepts, theories, tools, techniques, and methodologies of Cyber Security.
- **PSO2:** Applying Cyber Security principles to solve real-world security challenges.
- **PSO3:** Analysing security threats, vulnerabilities, and issues related to Cyber Security.
- **PSO4:** Evaluating alternative security solutions and making informed decisions to mitigate risks.
- **PSO5:** Designing and developing innovative solutions to address complex Cyber Security problems.

Career Avenues

Graduates of the B.Sc (Hons.) Cyber Security program have diverse career opportunities, including but not limited to the following:

1. **Cybersecurity Analyst:** Professionals in this role are tasked with safeguarding an organization's data and systems. They monitor networks for security breaches, develop security policies, conduct risk assessments, and implement robust security measures to defend against cyber threats.
2. **Security Consultant:** As security consultants, graduates advise organizations on how to protect their information systems and data. They evaluate existing security protocols, recommend improvements, and assist in the implementation of security solutions tailored to the organization's needs.

3. **Penetration Tester (Ethical Hacker):** Penetration testers are responsible for simulating cyberattacks on an organization's systems to identify vulnerabilities before malicious hackers can exploit them. They document findings, provide recommendations, and help strengthen the organization's security posture.

4. **Security Architect:** Security architects design, build, and maintain secure network infrastructures. They are responsible for anticipating potential security threats and creating solutions that protect the integrity and confidentiality of the organization's data.

5. **Incident Response Analyst:** Incident response analysts are the first line of defense when a security breach occurs. They investigate and respond to security incidents, mitigate threats, and develop strategies to prevent future attacks. Their role is crucial in minimizing the damage from cyber incidents.

6. **orensic Computer Analyst:** Forensic analysts recover, analyze, and preserve digital evidence in the aftermath of cybercrimes. They work closely with law enforcement agencies and legal teams to investigate cyberattacks and ensure that evidence is admissible in court.

7. **Security Operations Center (SOC) Analyst:** SOC analysts monitor and manage security systems within a Security Operations Center. They respond to alerts, investigate suspicious activities, and work to continuously improve the organization's security defenses.

8. **Cryptographer:** Cryptographers develop algorithms and encryption techniques to secure sensitive data. They work on creating new cryptographic methods and enhancing existing ones to protect information from unauthorized access.

9. **Compliance Analyst:** Compliance analysts ensure that an organization adheres to relevant laws, regulations, and standards related to cybersecurity. They conduct audits, assess risks, and implement compliance programs to ensure legal and regulatory requirements are met.

10. **Risk Management Specialist:** Risk management specialists identify potential threats to an organization's information assets and develop strategies to mitigate these risks. They conduct risk assessments, develop risk management plans, and ensure that the organization is prepared to handle potential security threats.

Duration

3 Years (6 Semesters) - Full-Time Program

Eligibility Criteria

Passed 10+2 or equivalent equivalent from any recognized board/university in any stream with Mathematics/Statistics/Computer/Information Science as one subject with minimum 50% aggregate marks.

Student's Structured Learning Experience in the Programme

University Education Objective

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

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.

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, Cybersecurity, Full Stack, and 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.

Academic Journey

Curriculum Structure and Degree Requirements

The B.Sc (Hons.) Cyber Security 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 eight semesters, balancing foundational knowledge, advanced topics, practical skills, and interdisciplinary learning. The total credit requirement for the B.Sc (Hons.) Cyber Security program is 135 credits.

Semester-wise Progression

Semester	Total Credits	Key Components
I (Odd)	25	• **Major Courses:** Fundamentals of Web Technologies, MATLAB Programming, Fundamentals of Software Engineering • **Lab Courses (Major):** Fundamentals of Web Technologies Lab, MATLAB Programming Lab, Clean Coding with Python Lab • **Skill Enhancement Courses (SEC):** Linux Environment Lab • **Value-Added Courses (VAC):** Environmental Studies & Disaster Management • **Minor Courses:** Essentials of Computer Science (MOOC) • **MOOC Courses:** Essentials of Computer Science (MOOC)

Semester	Total Credits	Key Components
II (Even)	24	• **Major Courses:** Introduction to Discrete Structures, Basics of Operating Systems, Concepts of Object-Oriented Programming Using C++ • **Lab Courses (Major):** Basics of Operating Systems Lab, Concepts of Object-Oriented Programming Using C++ Lab, Introduction to R Programming Lab • **Skill Enhancement Courses (SEC):** Introduction to R Programming • **Value-Added Courses (VAC):** Community Service • **Projects:** Minor Project-I • **Open Electives:** Open Elective-I • **Audit Courses:** Competitive Coding-I
III (Odd)	27	• **Major Courses:** Introduction to Data Structures, Basics of Probability & Statistics, Introduction to Java Programming • **Lab Courses (Major):** Introduction to Java Programming Lab, Introduction to Data Structures Lab, Machine Learning Lab • **Ability Enhancement Courses (AEC):** Verbal Ability • **Skill Enhancement Courses (SEC):** Introduction to Data Structures Lab • **Minor Courses:** Fundamentals of Machine Learning • **Value-Added Courses (VAC):** VAC-III (Innovations for Engineers, AWS Cloud Fundamentals, etc.) • **Internships (INT):** Summer Internship-I • **Audit Courses:** Competitive Coding-II • **Communication Skills (CS):** Club/Society Participation

Semester	Total Credits	Key Components
IV (Even)	24	• **Major Courses:** Fundamentals of Algorithm Design & Analysis, Introduction to Database Management Systems, Introduction to Computer Networks • **Lab Courses (Major):** Introduction to Database Management Systems Lab, Fundamentals of Algorithm Design & Analysis Lab, Introduction to Computer Networks Lab • **Ability Enhancement Courses (AEC):** Communication & Personality Development • **Projects:** Minor Project-II • **Open Electives:** Open Elective-II • **Audit Courses:** Competitive Coding-III • **Skill Enhancement Courses (SEC):** Fundamentals of Algorithm Design & Analysis Lab
V (Odd)	23	• **Major Courses:** Computer Organization and Architecture • **Discipline Specific Electives (DSE):** Cloud Computing or Full Stack Development • **Internships (INT):** Summer Internship-II • **Ability Enhancement Courses (AEC):** Arithmetic and Reasoning Skills • **Value-Added Courses (VAC):** Applied Programming and Problem-Solving Skills for Campus Interviews (Infosys Connect Program) • **Projects:** Discipline Specific Elective Lab
VI (Even)	12	• **Projects:** Industry Project / Research Project

Table 1: Semester-wise Academic Journey for B.Sc (Hons.) Cyber Security Program

Key Milestones and Experiences

- **Internships:**
  - **Summer Internship-I (Semester III):** Provides students with six weeks of industry exposure, allowing them to apply theoretical concepts in real-world scenarios.
  - **Summer Internship-II (Semester V):** Offers an extended internship experience, encouraging deeper engagement with industry projects, research & development, or startup initiatives.
- **Projects:**
  - **Minor Projects (Semester II and IV):** Focused on specific areas of study, these projects enable students to delve into practical applications of their coursework.
  - **Major Project (Semester VI):** A capstone project that integrates knowledge from the entire program, fostering innovation and problem-solving skills.
- **Discipline Specific Electives (DSE):**
  - **Cloud Computing (Discipline Specific Elective I):** Covers computational services in the cloud, including hands-on labs with Microsoft Azure.
  - **Full Stack Development (Discipline Specific Elective II):** Focuses on mobile application development, DevOps, and modern programming frameworks.
- **Skill Enhancement Courses (SEC):** Practical labs and courses designed to bolster technical skills in areas such as Linux environments, data structures, and computer science essentials.
- **Value-Added Courses (VAC):** Programs aimed at developing soft skills, ethical understanding, and career readiness, including boot camps and community engagement services.
- **Audit Courses:** Optional courses like Competitive Coding Bootcamps that allow students to enhance their coding proficiency through practice and challenges.
- **Ability Enhancement Courses (AEC):** Courses designed to enhance students' communication skills, critical thinking, and other essential abilities necessary for academic and professional success.
- **Communication Skills (CS):** Participation in clubs and societies to foster teamwork, leadership, and extracurricular engagement.

Overall Credit Distribution

The program ensures a balanced distribution of credits across various course categories to promote a well-rounded education:

Course Category	Total Credits
Major Courses	61
Skill Enhancement Courses (SEC)	4
Information and Data Communication (IDC)	5
Communication Skills (CS)	2

Course Category	Total Credits
Discipline Specific Electives (DSE)	15
Value-Added Courses (VAC)	8
MOOC Courses	2
Open Electives	6
Projects (PROJ.)	16
Audit Courses	0
Ability Enhancement Courses (AEC)	9
Internships (INT)	4
Total	**135**

Table 2: Overall Credit Distribution for B.Sc (Hons.) Cyber Security Program

Program Credits Summary

The academic journey is supported by a structured credit distribution across six semesters, ensuring that students meet the total requirement of 135 credits upon graduation. Below is a summary of the credit allocation per semester:

Semester	Credits
Semester I	25
Semester II	24
Semester III	27
Semester IV	24
Semester V	23
Semester VI	12
Total	**135**

Table 3: Credit Allocation per Semester

Discipline Specific Electives

To cater to diverse interests and emerging industry trends, students can choose from specialized electives in their final semesters:

Discipline Specific Elective I: Cloud Computing

- Computational Services in The Cloud
- Microsoft Azure Cloud Fundamentals
- Storage and Databases on Cloud
- Application Development and DevOps on Cloud

Discipline Specific Elective II: Full Stack Development

- Mobile Application Development using iOS
- DevOps & Automation
- .Net Framework
- New Age Programming Languages

Academic Journey

Curriculum Structure and Degree Requirements

The B.Sc (Hons.) Cyber Security program at the School of Engineering & Technology offers a comprehensive, structured curriculum designed to meet the demands of the rapidly evolving field of cyber security. 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.) Cyber Security program is 135 credits.

Course Categories and Credit Distribution

1. **Ability Enhancement Courses (AEC) – 9 credits:** Courses designed to enhance students' communication skills, critical thinking, and other essential abilities necessary for academic and professional success.

2. **Audit Courses – 0 credits:** Non-credit 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) – 4 credits:** Practical industry experience through internships, allowing students to apply theoretical knowledge in real-world settings and gain valuable professional exposure.

4. **Major Courses – 61 credits:** Core courses in cyber security covering key topics such as programming, data structures, algorithms, databases, software engineering, computer networks, cryptography, and cyber crime investigation, forming the foundation of the degree.

5. **Minor Courses – 27 credits:** Elective courses that allow students to specialize in areas such as Information and Data Communication (IDC), Communication Skills (CS), and Discipline Specific Electives (DSE), fostering interdisciplinary learning and providing flexibility in their academic journey.

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

7. **Skill Enhancement Courses (SEC) – 4 credits:** Practical, hands-on courses focused on developing specific technical skills, including programming languages, software tools, and other industry-relevant proficiencies.

8. **Projects (PROJ.) – 16 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) – 8 credits:** Courses aimed at fostering holistic development, emphasizing ethics, social responsibility, entrepreneurship, and leadership capabilities.

Course Category and Credits Summary

Course Category	Credits
Ability Enhancement Courses (AEC)	9
Audit Courses	0
Internships (INT)	4
Major Courses	61
Minor Courses	27
Open Electives	6
Skill Enhancement Courses (SEC)	4
Projects (PROJ)	16
Value-Added Courses (VAC)	8
Total	**135**

Table 4: Course Category and Credits Distribution for B.Sc (Hons.) Cyber Security Program

Program Name	Semester						Total Credits
	I	II	III	IV	V	VI	
B.Sc (H) Cyber Security	25	24	27	24	23	12	**135**

Course Registration and Scheduling

Effective course registration and scheduling are pivotal to the academic success of students in the B.Sc (Hons.) Cyber Security program. This section outlines the procedures and guidelines for selecting Majors and Minors, as well as the integration of internships and projects into the curriculum.

Major and Minor Selection

In the B.Sc (Hons.) Cyber Security program, students have the opportunity to specialize in their field through the selection of Major and Minor courses. This structure allows students to tailor their academic experience based on their interests and career goals.

Major Selection

The Major component comprises **61 credits** of core Cyber Security and related courses, providing a robust foundation in fundamental concepts such as algorithms, data

structures, software engineering, computer networks, cryptography, and cyber crime investigation. These courses are essential for developing the technical expertise required for advanced studies and professional practice in the field.

Minor Selection

The Minor component offers **27 credits** of department-specific electives across several cutting-edge domains:

1. **Information and Data Communication (IDC)**
2. **Communication Skills (CS)**
3. **Discipline Specific Electives (DSE)**

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. Selecting a Minor allows students to complement their Major coursework with focused expertise, thereby preparing them for dynamic roles in the evolving tech industry.

Furthermore, the program includes Value Added Courses (VACs), which provide students with practical skills and knowledge in areas like Design Thinking, AWS Cloud Fundamentals, Web Development, and Software Testing, among others. These courses are designed to enhance students' employability by equipping them with hands-on experience and industry-relevant skills.

Open Electives and Summer Internships further enrich the curriculum, giving students the opportunity to broaden their knowledge base and gain real-world experience in various professional settings.

Overall, the B.Sc (Hons.) Cyber Security program is structured to provide a balanced and comprehensive education, enabling students to tailor their learning journey to suit their individual aspirations while ensuring they are well-prepared for the challenges of the modern cyber security landscape.

Internships, Projects, and Experiential Learning

In alignment with our commitment to providing practical, hands-on experiences that complement academic learning, the 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 cyber security 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 internships are structured as follows:
   - **SIBC251: Summer Internship I (2 Credits)**
     - **Objective:** To provide students with their first exposure to a professional work environment, enabling the application of fundamental concepts learned in coursework.
     - **Duration:** Summer term, typically spanning 6-8 weeks.

  - **Focus:** Gaining initial industry experience, understanding professional practices, and developing workplace skills.
- **SIBC351: 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. **Projects:** Projects offer students the opportunity to engage in focused, project-based learning, emphasizing the application of core concepts in practical scenarios.
   - **ENSI152: Minor Project-I (2 Credits)**
     - **Objective:** To introduce students to project-based learning, where they work on small-scale projects requiring the application of fundamental principles.
     - **Focus:** Developing project planning, execution, and presentation skills.
   - **SIBC252: 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.
   - **SIBC352: Major Project/Industrial Training/Start-up Project (12 Credits)**
     - **Objective:** To provide a comprehensive, in-depth project experience in collaboration with industry partners, research institutions, or start-ups.
     - **Focus:** Addressing complex cyber security 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 cyber security profession and excel in their future careers.

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 $\leq$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 $\geq$80%, they are categorized as advanced learners.
  - They are provided with opportunities for career counseling, participation in technical events, and advanced courses.

Student Support Services

- **Mentor-Mentee:** Personalized guidance from experienced mentors to support academic and professional growth.
- **Counselling and Wellness Services:** Mental health and wellness support to ensure students maintain a healthy balance between academics and personal life.
- **Career Services and Training:** Comprehensive career services, including resume building, interview preparation, and job placement assistance.

Learning and Development Opportunities

- **Laboratories and Practical Learning:** State-of-the-art labs equipped with the latest technologies to provide hands-on experience.
- **Experiential Learning:** Real-world projects and internships to apply theoretical knowledge in practical settings.
- **Case-Based Learning/Problem-Based Learning/Project Based Learning:** Interactive learning methodologies to enhance critical thinking and problem-solving skills.
- **Workshops, Seminars, Guest Lectures:** Regular workshops and seminars conducted by industry experts to keep students updated with the latest trends.
- **Inside & Outside Classroom Learning:** A blend of classroom learning and external experiences to provide a holistic education.
- **Holistic Education:** Emphasis on overall development, including technical, soft, and ethical skills.

Assessment and Evaluation

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.

Projects: Projects are assessed based on the depth of research, problem-solving abilities, innovation, and the quality of the final report and presentation.

Internships: Internships are evaluated based on the completion of assigned tasks, professional conduct, and the submission of a comprehensive internship report.

Dissertation: The dissertation is assessed on the originality of research, methodology, analysis, and the clarity of presentation.

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

Feedback and Continuous Improvement Mechanisms

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:** Driven by analyzing feedback, implementing necessary changes, and reviewing the effectiveness of modifications.

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.

Examination and Evaluation Methods

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.) Cyber Security
Total Credits	135
Total Semesters	6

Semester I

SN	Category	Course Code	Course Title	L	T	P	C
1	Major-1	ENBC101	Fundamentals of Web Technologies	4	-	-	4
2	Major-2	ENBC103	MATLAB Programming	4	-	-	4
3	SEC-1	SEC050	Linux Environment Lab	-	-	4	2
4	SEC-2	ENSP107	Introduction to Programming in Python	4	0	-	4
5	Major-3	ENBC151	Fundamentals of Web Technologies Lab	-	-	2	1
6	Major-4	ENBC153	MATLAB Programming Lab	-	-	2	1
7	SEC-3	ENSP155	Programming in Python Lab	-	-	2	1
8	VAC-1	VAC-151	Environmental Studies & Disaster Management	2	-	-	2
9	Major-5	ENBC105	Fundamentals of Software Engineering	4	-	-	4
10	MOOC-1	SEC067	Essentials of Computer Science[1]	-	-	-	2
Total				**18**	**0**	**10**	**25**

[1]Course "**Essentials of Computer Science**" will be offered in an online self-paced mode. Students will be required to complete the suggested online module and produce the certification. Marks shall be allocated based on internal evaluation of 100 marks.

Semester II

SN	Category	Course Code	Course Title	L	T	P	C
1	SEC-4	ENSP114	Network Defence Essentials	4	-	-	4
2	Major-6	ENBC102	Introduction to Discrete Structures	3	1	-	4
3	Major-7	ENBC104	Basics of Operating Systems	3	1	-	4
4	Major-8	ENBC106	Concepts of Object Oriented Programming Using C++	3	1	-	4
5	SEC-5	ENSP166	Network Defence Essentials Lab	-	-	2	1
6	Major-9	ENBC152	Basics of Operating Systems Lab	-	-	2	1
7	Major-10	ENBC154	Concepts of Object Oriented Programming Using C++ Lab	-	-	2	1
8	Open Elective-1	-	Choose any one elective from the open pool of the university	3	-	-	3
9	Proj-1	ENSI152	Minor Project-I[2]	-	-	-	2
10	AUDIT-1	-	Competitive Coding- I	2	-	-	0
Total				**18**	**3**	**6**	**24**

[2]Marks for "**Minor Project-I**" shall be allocated based on internal evaluation of 100 marks. No End term evaluation is required.

Semester III

SN	Category	Course Code	Course Title	L	T	P	C
1	Major-11	ENBC201	Introduction to Data Structures	3	1	-	4
2	SEC-6	ENSP207	Fundamentals of Cryptography	4	-	-	4
3	Major-12	ENBC203	Basics of Probability & Statistics	4	-	-	4
4	Major-13	ENBC205	Introduction to Java Programming	3	1	-	4
5	AEC-1	AEC006	Verbal Ability	3	-	-	3
6	Major-14	ENBC251	Introduction to Java Programming Lab	-	-	2	1
7	Major-15	ENBC253	Introduction to Data Structures Lab	-	-	2	1
8	SEC-7	ENSP259	Fundamentals of Cryptography Lab	-	-	2	1
9	VAC-2	-	Students can choose any one elective from the pool of the VAC Courses offered by School	-	-	-	2
10	INT-1	SIBC251	Summer Internship-I	-	-	-	2
11	AUDIT-2	-	Competitive Coding - II	2	-	-	0
12	CS-1	CS001	Club/Society	1	-	-	1
Total				**20**	**2**	**6**	**27**

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

Semester IV

SN	Category	Course Code	Course Title	L	T	P	C
1	Major-16	ENBC202	Fundamentals of Algorithm Design & Analysis	3	1	-	4
2	Major-17	ENBC204	Introduction to Database Management Systems	3	1	-	4
3	Major-18	ENBC206	Introduction to Computer Networks	3	1	-	4
4	Major-19	ENBC252	Introduction to Database Management Systems Lab	-	-	2	1
5	Major-20	ENBC254	Fundamentals of Algorithm Design & Analysis Lab	-	-	2	1
6	Major-21	ENBC256	Introduction to Computer Networks Lab	-	-	2	1
7	AEC-2	AEC007	Communication & Personality Development	3	-	-	3
8	Proj-2	SIBC252	Minor Project-II	-	-	-	2
9	Open Elective-2	-	Choose any one elective from the open pool of the university	3	-	-	3
10	AUDIT-3	-	Competitive Coding- III	2	-	-	0
11	CS-2	CS002	Community Service	1	-	-	1
Total				**18**	**3**	**6**	**24**

Semester V

SN	Category	Course Code	Course Title	L	T	P	C
1	Major-22	ENBC301	Computer Organization and Architecture	3	1	-	4
2	DSE-1		Discipline Specific Elective -I	4	-	-	4
3	DSE-2		Discipline Specific Elective -I Lab	-	-	2	1
4	DSE-3		Discipline Specific Elective -II	4	-	-	4
5	DSE-4		Discipline Specific Elective -II Lab	-	-	2	1
6	SEC-8	ENSP367	Ethical Hacking Lab	-	-	4	2
7	INT-2	SIBC351	Summer Internship-II[3]	-	-	-	2
8	AEC-3	AEC008	Arithmetic and Reasoning Skills	3	-	-	3
9	MOOC-2		Applied Programming and Problem-Solving Skills for Campus Interviews (Infosys Connect Program)	-	-	-	2
Total				**14**	**1**	**8**	**23**

Semester VI

SN	Category	Course Code	Course Title	L	T	P	C
1	Project	SIBC352	Industry Project /Research Project	-	-	-	12
Total				-	-	-	**12**

[3]For "**Summer Internship-II**" students have to complete 6 weeks internship during the summers and submit a completion certificate. Students will be evaluated on a scale of 100 during the 5th semester for allocation of marks for internship.

Discipline Specific Elective I (Cyber Security)

SN	Category	Course Code	Course Title	L	T	P	C
(i)	DSE	ENSP301	Secure Coding and Vulnerabilities	4	-	-	4
(ii)	DSE	ENSP351	Secure Coding and Vulnerabilities Lab	-	-	2	1
(iii)	DSE	ENSP303	Cyber Crime Investigation & Digital Forensics	4	-	-	4
(iv)	DSE	ENSP353	Cyber Crime Investigation & Digital Forensics Lab	-	-	2	1
(v)	DSE	ENSP305	AI in Cyber Security	4	-	-	4
(vi)	DSE	ENSP355	AI in Cyber Security Lab	-	-	2	1
(vii)	DSE	ENSP307	Social Media Security	4	-	-	4
(viii)	DSE	ENSP357	Social Media Security Lab	-	-	2	1

Discipline Specific Elective II (Full Stack Development)

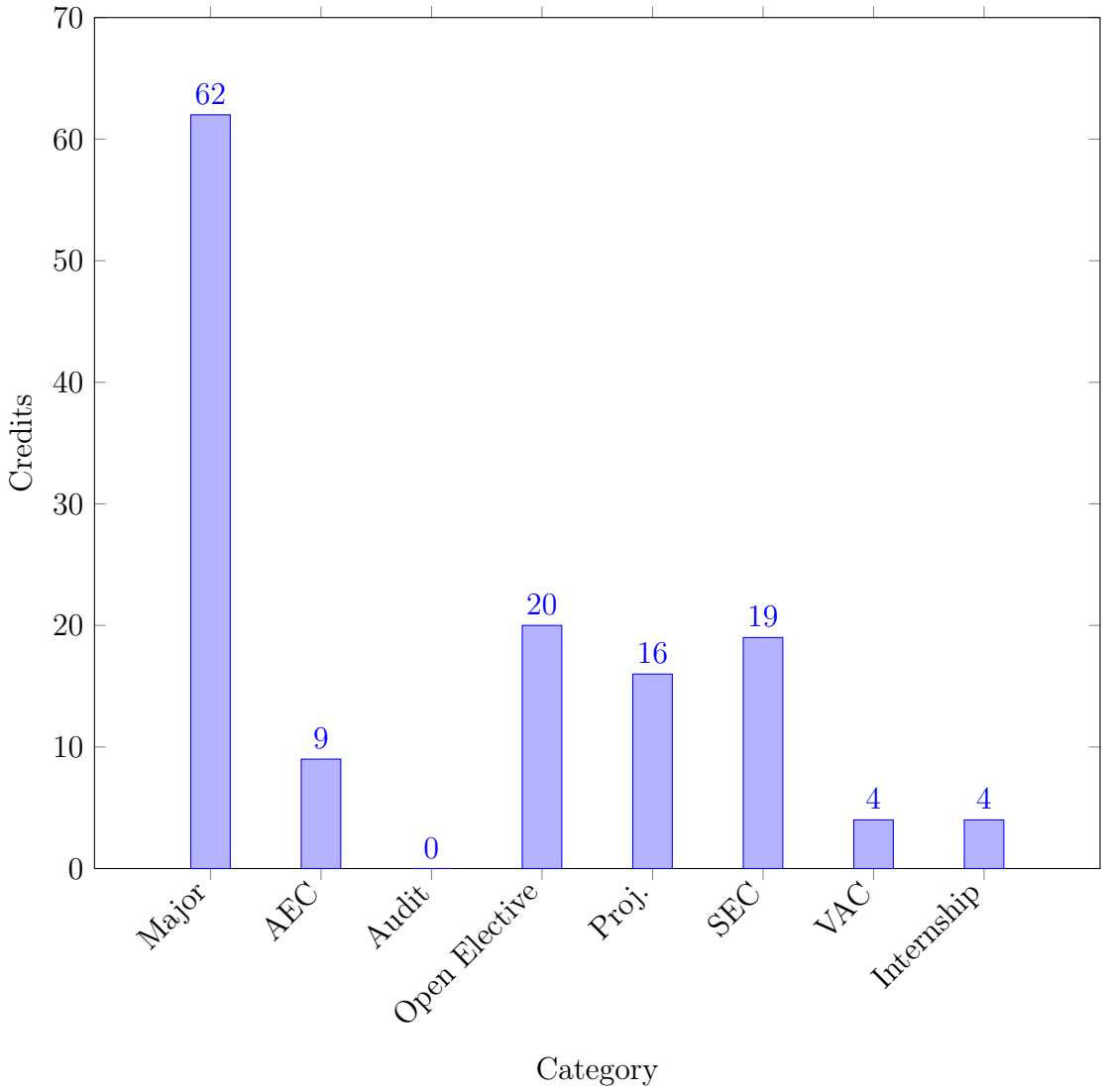
SN	Category	Course Code	Course Title	L	T	P	C
(i)	DSE	ENSP409	Mobile Application Development using iOS	4	-	-	4
(ii)	DSE	ENSP459	Mobile Application Development using iOS Lab	-	-	2	1
(iii)	DSE	ENSP411	DevOps & Automation	4	-	-	4
(iv)	DSE	ENSP461	DevOps & Automation Lab	-	-	2	1
(v)	DSE	ENSP413	.Net Framework	4	-	-	4
(vi)	DSE	ENSP463	.Net Framework Lab	-	-	2	1
(vii)	DSE	ENSP415	New Age Programming Languages	4	-	-	4
(viii)	DSE	ENSP465	New Age Programming Languages Lab	-	-	2	1

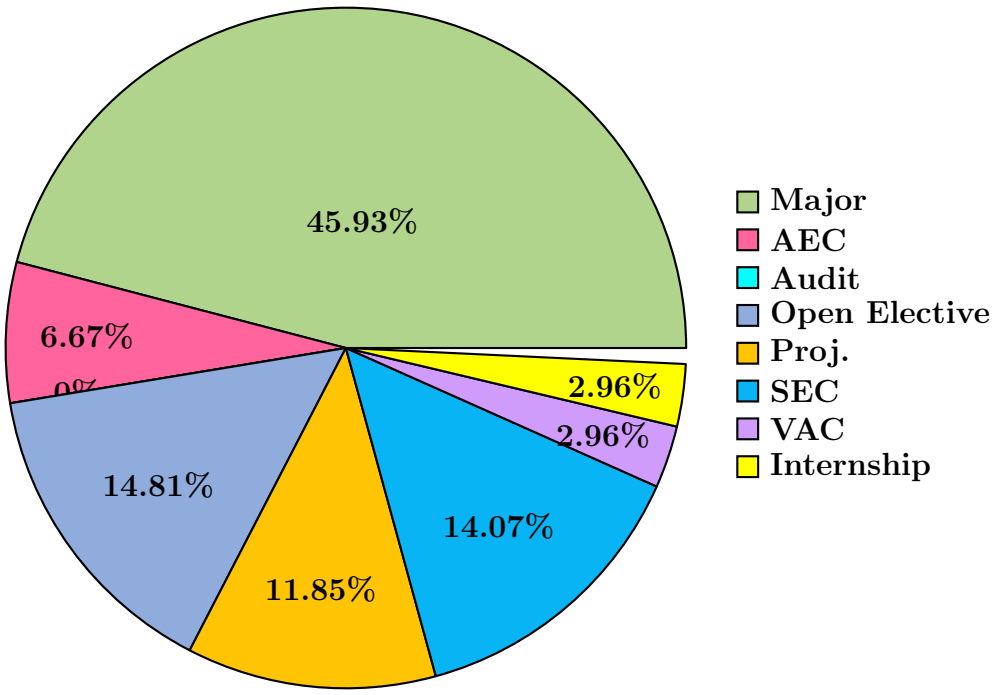
Program Credits

Program Name	Semester						Total Credits
	I	II	III	IV	V	VI	
B.Sc (H) Cyber Security	25	24	27	24	23	12	**135**

Total Credits: *135*

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/ Participation/ 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

Fundamentals of Web Technologies

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Fundamentals of Web Technologies	ENBC101	4-0-0	4
Type of Course:	Major		
Pre-requisite(s):	Basic knowledge of computer systems		

Course Perspective: This course introduces the foundational concepts and technologies of the World Wide Web (WWW). It covers the architecture of web systems, client-side scripting, web design principles, and advanced web technologies like XML and AJAX. The course aims to equip students with the skills to create, design, and manage effective web systems. The course is divided into 4 units:

1. Introduction to Web Technology
2. Client-side Scripting
3. Concepts of Effective Web Design
4. XML and Advanced Web Technologies

The Course Outcomes (COs)

On completion of the course the participants will be:

COs	**Statements**
CO 1	Applying foundational concepts of web technology, including WWW, OSI and TCP/IP models, HTTP, and HTML5, to understand and create web systems.
CO 2	Developing client-side scripting skills using JavaScript and CSS3 to enhance web page interactivity and styling.
CO 3	Implementing effective web design principles to address design issues, user-centric design, and website navigation.
CO 4	Utilizing XML, web services, and AJAX for advanced web technologies to create dynamic and efficient web applications.

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

Course Outline

Unit Number: 1	Title: Introduction to Web Technology	No. of hours: 8

Content:

- Concept of WWW, Internet and WWW
- OSI Reference Model
- Understanding Web System Architecture
- Understanding 3-Tier Web Architecture
- **Layers in the TCP/IP Model**: Physical, Link, Internet, Transport, Application
- Web Browsers
- **Retrieving Documents on the Web**: URL and Domain Name System
- **Overview of HTTP**: Request and Response
- **HTML5**: Introduction, Document structure tags, comments, Text formatting, inserting special characters, anchor tag, adding images and sound, lists, tables, frames, forms, Image maps, Meta tags, Character entities

Unit Number: 2	Title: Client-side Scripting	No. of hours: 12

Content:

- **JavaScript**: Data Types, Control Statements, Operators, Built-in and User Defined Functions, Objects in JavaScript, Handling Events
- HTML Document Object Model
- **Page Styling**: Separation of content and presentation in HTML, CSS3 - Types of Style Sheets – Internal, inline, and External style sheets, customizing common HTML elements, types of CSS selectors
- **Introduction to Forms and HTML Controls**: Creating Forms, Using HTML Controls

Unit Number: 3	Title: Concepts of Effective Web Design	No. of hours: 12

Content:

- Concepts of effective web design
- Web design issues including Browser, Bandwidth and Cache, Display resolution, Look and Feel of the Website, Page Layout and linking, User-centric design, Sitemap, Planning and publishing website, Designing effective navigation, Browser architecture, and Website structure
- Introduction to DHTML

Unit Number: 4	Title: XML and Advanced Web Technologies	No. of hours: 8

Content:

- **Introduction to XML**: Markup languages, XML Syntax, XML Declaration, Elements, Attributes, Valid XML Documents, Viewing XML, XML Parser
- **Introduction to Web Services**: UDDI, SOAP, WSDL, Web Service Architecture
- **AJAX**: Introduction, AJAX programming, improving web page performance using AJAX

Learning Experience

Classroom Learning Experience

- **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 performance, 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.

Outside Classroom Learning Experience

- **Assignments:** Extend classroom concepts through take-home tasks, requiring students to design web pages, implement JavaScript functions, and explore advanced HTML and CSS techniques.
- **Online Forums:** Facilitate discussions on complex web development topics and troubleshooting, encouraging collaborative learning outside the classroom.
- **Self-Study:** Encourage students to independently explore web development tools, frameworks, and design practices to enhance their understanding of the subject.
- **Group Work:** Promote collaboration on larger projects that involve the application of web development techniques to create fully functioning websites.
- **Additional Resources:** Provide access to coding tutorials, videos, and documentation for deeper exploration of web technologies.
- **Peer Feedback:** Encourage students to review each other's work, share feedback, and suggest improvements through collaborative online platforms.

Text Books

- "Web Technologies: HTML, JavaScript, PHP, Java, JSP, XML and AJAX, Black Book" by Kogent Learning Solutions Inc

Reference Books

- Web Technologies, Uttam K. Roy, Oxford University Press
- HTML Black Book, Stephen Holzner, Wiley Dreamtech.
- Web Technology, Rajkamal, Tata McGraw-Hill.
- Web Technologies: A Computer Science Perspective, Jeffrey C. Jackson, Pearson.
- XML: How to Program, Deitel & Deitel Nieto

Additional Readings

Self-Learning Components:

1. Link to W3Schools HTML tutorial: https://www.w3schools.com/html/
2. Link to Mozilla Developer Network (MDN) JavaScript documentation: https://developer.mozilla.org/en-US/docs/Web/JavaScript
3. Link to CSS-Tricks for CSS resources and guides: https://css-tricks.com/
4. Link to XML tutorial by W3Schools: https://www.w3schools.com/xml/
5. Link to AJAX tutorial by W3Schools: https://www.w3schools.com/xml/ajax_intro.asp
6. Link to NPTEL Web Technologies course: https://onlinecourses.nptel.ac.in/noc21_cs30/

MATLAB Programming

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
MATLAB Programming	ENBC103	4-0-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course aims to introduce students to MATLAB programming, covering fundamental concepts, programming constructs, data visualization, file I/O operations, and advanced applications. The course is divided into 4 units:

1. Introduction to MATLAB
2. Programming Constructs
3. Data Visualization and File I/O
4. Introductory Applications in MATLAB

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding and applying basic concepts and environment of MATLAB, including its history, features, syntax, scripts, and functions.
CO 2	Implementing programming constructs in MATLAB, such as control statements, loops, vectors, matrices, built-in functions, and user-defined functions.
CO 3	Utilizing MATLAB for data visualization and file I/O operations, including 2D/3D plotting, advanced plotting techniques, and handling various file formats.
CO 4	Exploring advanced MATLAB topics and applications, such as image processing, signal processing, and Simulink for solving real-world problems.

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

Course Outline

Unit Number: 1	Title: Introduction to MATLAB	No. of hours: 8
Content:		
• Overview of MATLAB: History, Features, and Applications • MATLAB Environment: Command Window, Workspace, and Command History • Basic Syntax: Variables, Data Types, and Operators • Scripts and Functions: Creating, Saving, and Running Scripts • Input and Output: Getting User Input, Displaying Output		
Unit Number: 2	**Title: Programming Constructs**	**No. of hours: 12**
Content:		
• Control Statements: Conditional Statements (if, else, switch) • Loops: for, while, and nested loops • Vectors and Matrices: Creating, Indexing, and Manipulating • Built-in Functions: Using common mathematical and statistical functions • User-Defined Functions: Writing and calling functions, function handles		
Unit Number: 3	**Title: Data Visualization and File I/O**	**No. of hours: 12**
Content:		
• Plotting: 2D and 3D plots, Customizing Plots (labels, titles, legends) • Advanced Plotting Techniques: Subplots, Logarithmic plots, Bar and Pie charts • File I/O: Reading from and writing to files, File formats (txt, csv, xls) • Data Import and Export: Importing data from external sources, Exporting results		
Unit Number: 4	**Title: Introductory Applications in MATLAB**	**No. of hours: 8**
Content:		

- Basic Numerical Methods: Solving simple linear equations, Numerical integration, and differentiation
- Basic Data Analysis: Statistical analysis, Curve fitting, and interpolation
- Introduction to Image Processing: Simple image manipulation and analysis
- Simulink: Introduction to Simulink, Creating simple models
- Basic Algorithm Implementation: Implementing simple algorithms like Fibonacci series, factorial calculation, and basic sorting techniques

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** Engage students with detailed explanations of MATLAB concepts and programming constructs using live coding sessions and interactive demonstrations of key features, such as plotting and data analysis.
- **Hands-On Labs:** Provide extensive lab sessions where students work directly in the MATLAB environment, writing scripts, creating functions, and manipulating data. These labs help students reinforce theoretical knowledge through practical application.
- **Group Projects:** Encourage collaborative learning by assigning group projects where students develop complex MATLAB programs to solve real-world problems, such as data visualization tasks or basic algorithm implementation.
- **In-Class Problem Solving:** Conduct problem-solving sessions where students work on MATLAB challenges related to engineering, data science, and mathematical modeling.
- **Immediate Feedback:** Provide in-class feedback through quizzes and lab exercises, ensuring students' understanding of key MATLAB programming concepts.

Outside Classroom Learning Experience

- **Assignments & Case Studies:** Assign practical tasks and case studies that require students to apply MATLAB programming to real-world scenarios, such as engineering simulations, image processing, and data analysis.
- **Online Forums:** Facilitate discussions and troubleshooting sessions on MATLAB programming, encouraging peer-to-peer learning and collaboration outside the classroom.
- **Self-Study:** Encourage students to explore additional MATLAB features, toolboxes, and documentation to enhance their understanding of advanced topics.
- **Collaborative Projects:** Promote group work outside of class where students can apply MATLAB programming to develop real-world solutions and share their progress.

- **Practice Problems:** Provide access to additional practice exercises, problem sets, and example scripts that students can work on to improve their programming skills.
- **Peer Feedback:** Encourage students to review each other's MATLAB code, providing constructive feedback to help them improve their coding techniques.

Text Books

- Amos Gilat, *MATLAB: An Introduction with Applications*, 5th Edition, Wiley, 2014.
- Steven C. Chapra, *Applied Numerical Methods with MATLAB for Engineers and Scientists*, 3rd Edition, McGraw-Hill, 2012.
- Holly Moore, *MATLAB for Engineers*, 4th Edition, Pearson, 2017.

Additional Readings

Self-Learning Components:

1. Link to MATLAB documentation: https://www.mathworks.com/help/MATLAB/
2. Link to MATLAB tutorials on MathWorks: https://www.mathworks.com/learn/tutorials/MATLAB-onramp.html
3. Link to MATLAB Central for user-contributed content and discussions: https://www.mathworks.com/MATLABcentral/
4. Link to NPTEL MATLAB course: https://onlinecourses.nptel.ac.in/noc18_cs06/preview

MATLAB Programming Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
MATLAB Programming Lab	ENBC153	0-0-2	1
Type of Course:	Major		

Defined Course Outcomes

COs	**Statements**
CO 1	Implementing basic programming constructs and algorithms to solve computational problems.
CO 2	Applying numerical methods and techniques for solving mathematical problems.
CO 3	Analyzing data using visualization and reporting solutions.
CO 4	Creating advanced Matlab applications for image processing, signal processing, and simulation.

Proposed Lab Experiments

S.N	**Lab Task**	**Mapped CO/COs**
1	Write a MATLAB script to display "Hello, World!" and get user input to display a personalized message.	CO1
2	Create a MATLAB script that performs basic arithmetic operations (addition, subtraction, multiplication, division) on two user-input numbers and displays the results.	CO1
3	Write a MATLAB function to convert temperatures from Celsius to Fahrenheit and vice versa.	CO1
4	Develop a MATLAB script to solve a quadratic equation ($ax^2 + bx + c = 0$) and display the roots.	CO2
5	Implement a MATLAB script that uses `if-else` statements to categorize a given number as positive, negative, or zero.	CO2
6	Write a MATLAB script that demonstrates the use of `for` and `while` loops to compute the factorial of a number.	CO2
7	Create a MATLAB function that takes a matrix as input and returns the transpose of the matrix.	CO2
8	Write a MATLAB script to perform matrix addition, subtraction, and multiplication using user-defined matrices.	CO2

S.N	Lab Task	Mapped CO/COs
9	Develop a MATLAB script to plot a sine wave and a cosine wave on the same graph with appropriate labels and legends.	CO3
10	Implement a MATLAB script to read data from a CSV file and plot it using a bar chart.	CO3
11	Write a MATLAB script to create subplots of various mathematical functions (sine, cosine, exponential) in a single figure.	CO3
12	Create a MATLAB script to read and display an image, then convert it to grayscale.	CO4
13	Develop a MATLAB script to solve a system of linear equations using matrix inversion.	CO4
14	Implement a MATLAB script for numerical integration of a given function using the trapezoidal rule.	CO4
15	Write a MATLAB script to fit a polynomial to a set of data points and plot the result.	CO4
16	Create a MATLAB script that performs statistical analysis (mean, median, standard deviation) on a dataset.	CO4
17	Develop a MATLAB script to implement the Fibonacci series using a loop.	CO2, CO4
18	Write a MATLAB function to calculate and plot the factorial of a number using recursion.	CO2, CO4
19	Create a MATLAB script to import data from an Excel file and perform basic data analysis (sum, average).	CO3, CO4
20	Develop a MATLAB script to perform simple image processing operations like edge detection and histogram equalization.	CO4
21	**Personal Expense Tracker:** Track daily expenses and generate summary reports. Create a MATLAB application to allow users to input daily expenses under various categories, store the data, and generate weekly and monthly summary reports with graphical visualizations of spending patterns.	CO3, CO4
22	**Temperature Converter:** Convert temperatures between Celsius, Fahrenheit, and Kelvin. Develop a MATLAB program that takes user input for temperature values and converts them between Celsius, Fahrenheit, and Kelvin. The application should provide a user-friendly interface and display the conversion results.	CO1, CO4
23	**Simple Calculator:** Perform basic arithmetic operations. Create a MATLAB application that functions as a simple calculator, supporting addition, subtraction, multiplication, and division operations. Implement error handling for invalid inputs and division by zero.	CO2, CO4

S.N	Lab Task	Mapped CO/COs
24	**Data Visualization of Student Scores:** Generate charts and graphs. Develop a MATLAB program to read student score data from a file, perform basic statistical analysis, and generate visualizations such as bar charts, histograms, and scatter plots to represent the data.	CO3, CO4
25	**Loan Amortization Schedule:** Calculate loan payment schedules. Write a MATLAB application that calculates and displays the loan amortization schedule based on user input for loan amount, interest rate, and loan term. The application should provide a detailed breakdown of each payment.	CO3, CO4
26	**Image Processing – Edge Detection:** Implement edge detection using MATLAB functions. Develop a MATLAB script to read an image file, apply edge detection algorithms (such as Sobel or Canny), and display the original and processed images side by side.	CO4
27	**Simulation of Projectile Motion:** Visualize projectile trajectories. Create a MATLAB application to simulate the motion of a projectile given initial velocity and angle. The application should plot the trajectory and allow users to adjust parameters to see the effect on the motion.	CO1, CO4
28	**Weather Data Analysis:** Analyze and visualize weather data trends. Write a MATLAB program to read weather data from a file, perform analysis such as calculating averages and trends, and generate visualizations like line plots and histograms to represent temperature, humidity, and precipitation data over time.	CO3, CO4

Online Learning Resources

- **Codecademy:** Interactive coding platform that offers hands-on MATLAB courses, teaching both the basics and more advanced topics. Ideal for practicing specific programming tasks.
  https://www.codecademy.com/learn/learn-MATLAB
- **HackerRank:** Provides a vast range of programming problems across various domains of computer science, along with a dedicated MATLAB domain. Great for practicing coding skills and understanding algorithms.
  https://www.hackerrank.com/domains/tutorials/10-days-of-MATLAB
- **LeetCode:** Known for its extensive array of programming challenges that can help improve your understanding of data structures and algorithms. It's particularly good for preparing for technical job interviews.
  https://leetcode.com/
- **GitHub:** Not just a code repository, GitHub offers collaborative features and a wealth of open-source projects where students can engage in real-world software development and contribute to ongoing projects.
  https://github.com/

Linux Environment Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Linux Environment Lab	SEC050	0-0-4	2
Type of Course:	SEC		

Defined Course Outcomes

COs	**Statements**
CO 1	Understanding the fundamental concepts and operations of the Linux operating system, including file management, directory structures, and basic shell commands.
CO 2	Applying basic scripting skills to automate routine tasks and simplify system management in a Linux environment.
CO 3	Analyzing system performance and security logs to identify potential issues and optimize Linux system operations.
CO 4	Creating and manage network configurations and security settings to ensure safe and efficient operation of Linux servers.

Proposed Lab Experiments

S.N	**Lab Task**	**Mapped CO/COs**
1	Navigate and manipulate files and directories using basic shell commands in Linux. Learn to manage file systems effectively.	CO1
2	Use grep and sed to perform text processing and data extraction from logs. Focus on extracting useful information from system logs for troubleshooting.	CO1
3	Write a shell script to automate the backup of files and directories, ensuring data integrity and recoverability.	CO2
4	Monitor system performance using commands like top, vmstat, and iotop. Identify and report on resource usage and potential bottlenecks.	CO3
5	Configure and manage user permissions and ownerships in a Linux environment, emphasizing security best practices.	CO1
6	Install and configure software packages using the package manager, understanding repository management and package dependencies.	CO1
7	Create and execute a shell script that automates system updates and cleanup processes, ensuring system efficiency and security.	CO2

S.N	Lab Task	Mapped CO/COs
8	Analyze security logs to detect potential unauthorized access or vulnerabilities, enhancing system security through log analysis.	CO3
9	Set up and manage network services such as SSH, FTP, and web servers, focusing on secure and efficient network operations.	CO4
10	Implement basic firewall settings using iptables or firewalld, securing the system against unauthorized network access.	CO4
11	Write advanced bash scripts incorporating loops, conditions, and functions to automate complex administrative tasks.	CO2
12	Utilize crontab to schedule and manage routine tasks across the system, ensuring regular maintenance and operations automation.	CO2
13	Configure and manage virtual hosts on an Apache or Nginx web server, focusing on hosting multiple websites on a single server.	CO4
14	Set up and secure a basic MySQL or PostgreSQL database server, ensuring data integrity and access control.	CO4
15	Use system monitoring tools to create performance reports and identify bottlenecks, optimizing server performance.	CO3
16	Automate the monitoring and alerting of system resources using custom scripts, enhancing proactive system management.	CO2
17	Configure and manage file sharing services using NFS or Samba, focusing on seamless file access within a network.	CO4
18	Implement a RAID array to manage disk redundancy and performance, ensuring data availability and fault tolerance.	CO4
19	Develop scripts to manage network configurations and troubleshoot common issues, enhancing network reliability and performance.	CO2
20	Secure Linux systems by configuring SELinux or AppArmor policies, focusing on enforcing strict security policies and controls.	CO3
1	**LAMP Stack Configuration:** Configure and deploy a full LAMP (Linux, Apache, MySQL, PHP) stack. Develop a comprehensive environment that allows for the hosting of dynamic websites and applications. The project should include setting up a virtual server, configuring Apache, installing MySQL, and deploying a sample PHP application.	CO4
2	**Dockerized Web Application:** Create a Docker container setup for deploying web applications efficiently. Students will develop Dockerfiles, manage Docker containers, and deploy a lightweight web application using Docker. This project should demonstrate the use of containers to streamline development and production workflows.	CO4
3	**Intrusion Detection System:** Develop a basic intrusion detection system (IDS) using open-source tools on a Linux system. The project involves setting up Snort or similar tools to monitor network traffic for suspicious activities, configuring alert systems, and analyzing intrusion attempts.	CO3

S.N	Lab Task	Mapped CO/COs
4	**Linux Server Backup Solution:** Implement a comprehensive backup solution for Linux servers. This project should cover the creation of backup scripts, scheduling of automatic backups, and restoration procedures. Students will explore different tools and methods for efficient data backup and recovery.	CO2
5	**Network Monitoring with Nagios:** Set up Nagios on a Linux server to monitor network health and performance. The project should include configuring Nagios to monitor critical network parameters, setting up alerts for system failures, and creating reports for network status. This provides practical experience in network management and monitoring.	CO4

Online Learning Resources

- **Codecademy:** Interactive platform offering courses on Linux command-line basics and shell scripting.
  https://www.codecademy.com/learn/learn-the-command-line
- **Linux Journey:** Comprehensive resource for learning Linux, from basic to advanced concepts.
  https://linuxjourney.com/
- **The Linux Documentation Project:** Offers extensive documentation and guides on various Linux topics.
  http://www.tldp.org/
- **GitHub:** Explore and contribute to open-source Linux projects and scripts.
  https://github.com/

Introduction to Programming in Python

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Programming in Python	ENSP107	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	None		

Course Perspective: This course introduces Python programming with a focus on cybersecurity applications. It covers Python fundamentals, secure coding practices, and key concepts relevant to cybersecurity.

The Course Outcomes (COs)

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

COs	Statements
CO 1	Interpreting the fundamental Python syntax and semantics and be fluent in the use of Python control flow statements.
CO 2	Expressing proficiency in the handling of strings and functions.
CO 3	Determining methods to create and manipulate Python programs utilizing data structures like lists, dictionaries, tuples, and sets.
CO 4	Identifying commonly used operations involving file systems and regular expressions.
CO 5	Articulating Object-Oriented Programming concepts such as encapsulation, inheritance, and polymorphism as used in Python.

Students are expected to demonstrate knowledge of Python programming concepts with a focus on cybersecurity by the end of the course.

Course Outline

Unit Number: 1	Title: Introduction to Python and Program Flow Control	No. of hours: 12
Content:		
• Python variables, basic operators, and Python blocks. • Python data types (int, float, etc.), numeric data type usage. • Conditional blocks: if, else, and elif. • Simple loops: for loop (ranges, strings, lists, dictionaries). • While loops, loop manipulation (pass, continue, break, and else). • Programming with conditional and loop blocks.		
Unit Number: 2	**Title: Python Data Structures**	**No. of hours: 8**
Content:		
• Strings and their operations, string manipulation methods. • Lists, list slicing, and manipulation. • Tuples and their uses. • Dictionaries and dictionary manipulation. • Defining functions and using them in Python programs. • Organizing code with functions.		
Unit Number: 3	**Title: File Operations and Regular Expressions**	**No. of hours: 8**
Content:		
• Reading and writing files: read(), readline(), readlines(), write(), writelines(). • Manipulating file pointers using seek(). • Introduction to regular expressions. • Pattern matching and text processing with regular expressions.		
Unit Number: 4	**Title: Object-Oriented Programming (OOP) Concepts in Python**	**No. of hours: 8**
Content:		

- Introduction to Object-Oriented Programming (OOP) in Python.
- Classes and Objects: Defining and using classes, creating objects from classes.
- Encapsulation: Understanding private and public attributes, getter and setter methods.
- Inheritance: Single and multiple inheritance, overriding methods.
- Polymorphism: Method overloading and overriding in Python.
- Constructors and Destructors: Understanding __init__() and __del__() methods.
- Special Methods in Python: __str__(), __repr__(), operator overloading.
- Case Studies and examples to demonstrate the practical use of OOP concepts in Python.

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** Engage students with in-depth explanations of Python syntax and clean coding practices, using real-world examples and interactive coding sessions to illustrate key concepts.
- **Hands-On Labs:** Provide practical lab sessions where students write Python code, focusing on clean code principles such as proper naming conventions, code organization, and error handling. These labs reinforce best practices in Python programming.
- **In-Class Code Reviews:** Conduct peer code reviews during lab sessions to provide immediate feedback on clean coding practices, helping students refine their Python skills.
- **Group Projects:** Encourage collaboration through group projects where students apply clean coding principles to develop Python programs, working together to refactor and improve code quality.
- **Immediate Feedback:** Provide feedback during lectures and lab sessions through quizzes and code challenges that test students' understanding of clean coding principles.

Outside Classroom Learning Experience

- **Assignments & Case Studies:** Assign tasks and case studies that challenge students to apply clean coding techniques in Python to solve complex problems, such as data handling and machine learning tasks.

- **Online Discussions:** Facilitate discussions on clean coding practices, common pitfalls, and best solutions through online forums, encouraging peer-to-peer support and knowledge sharing.
- **Self-Study:** Encourage students to independently explore advanced Python libraries and tools while practicing clean coding techniques.
- **Collaborative Coding Projects:** Promote teamwork on Python projects where students apply clean coding techniques outside the classroom, helping each other refine their code structure and readability.
- **Peer Code Review:** Set up peer code review activities where students critique and offer feedback on each other's code outside of class, fostering continuous improvement.
- **Practice Problems:** Provide additional clean coding practice exercises for students to complete on their own, reinforcing best practices in Python.

Textbooks

- T1: Wesley J. Chun, "Core Python Applications Programming", 3rd Edition, Pearson Education, 2016
- T2: Lambert, Fundamentals of Python: First Programs with MindTap, 2nd Edition, Cengage Learning publication
- T3: Charles Dierbach, "Introduction to Computer Science using Python", Wiley, 2015
- T4: Jeeva Jose & P. Sojanlal, "Introduction to Computing and Problem Solving with PYTHON", Khanna Publishers, New Delhi, 2016

Reference Books

- R1: Mark Lutz, "Learning Python", 5th Edition, O'Reilly Publication, 2013
- R2: John Zelle, "Python Programming: An Introduction to Computer Science", 2nd Edition, Course Technology Cengage Learning Publications, 2013
- R3: Michel Dawson, "Python Programming for Absolute Beginners", 3rd Edition, Course Technology Cengage Learning Publications, 2013
- R4: David Beazley, Brian Jones, "Python Cookbook", 3rd Edition, O'Reilly Publication, 2013

Online Learning Resources

The following online resources will help students further enhance their understanding of Python programming and cybersecurity applications:

- **Codecademy:** Interactive platform offering courses on Python programming, from basics to advanced topics, including applications in cybersecurity.
  https://www.codecademy.com/learn/learn-python-3

- **Coursera:** Courses on Python for cybersecurity offered by leading universities and institutions. Recommended course: "Python for Cybersecurity" specialization.
  https://www.coursera.org/specializations/python-for-cybersecurity
- **edX:** Offers various courses on Python and its applications, including data analysis and cybersecurity. Recommended course: "Introduction to Python for Data Science."
  https://www.edx.org/course/introduction-to-python-for-data-science
- **Kaggle:** Data science community with extensive Python tutorials and hands-on projects, including cybersecurity-related data science challenges.
  https://www.kaggle.com/learn/python
- **Real Python:** A comprehensive Python learning platform offering tutorials, coding exercises, and courses on Python programming, including security-related tasks.
  https://realpython.com/
- **GitHub:** An open-source platform to explore Python projects related to cybersecurity, contribute to projects, and learn from code written by experienced programmers.
  https://github.com/
- **Python Docs:** The official Python documentation is an excellent resource to learn Python, including detailed explanations of libraries and modules used for cybersecurity.
  https://docs.python.org/3/
- **Udemy:** Offers a variety of Python courses with practical projects, including topics on Python for cybersecurity.
  https://www.udemy.com/

Fundamentals of Web Technologies Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Fundamentals of Web Technologies Lab	ENBC151	0-0-2	1
Type of Course:	Major		

Defined Course Outcomes

COs	**Statements**
CO 1	Understanding the fundamentals of Web Technology, including the OSI model, TCP/IP layers, and web system architecture.
CO 2	Developing client-side scripts using JavaScript and style web pages effectively using CSS3.
CO 3	Applying principles of effective web design to create user-centric websites.
CO 4	Utilizing XML and AJAX to enhance web applications and develop basic web services.

S.N	**Lab Task**	**Mapped CO/COs**
1	Create a simple HTML page to understand basic HTML5 tags and structure including headings, paragraphs, and divs.	CO1
2	Utilize CSS3 to style a web page by applying different styles to HTML elements using class and id selectors.	CO2
3	Develop a multi-page website incorporating images, tables, and lists to demonstrate the usage of HTML5 structural elements.	CO1
4	Implement client-side form validation using JavaScript to enhance user interaction and data integrity.	CO2
5	Explore the Document Object Model (DOM) by dynamically modifying the content and style of a webpage with JavaScript.	CO2
6	Design a responsive web layout using CSS3 media queries to ensure the webpage is adaptable to different devices like tablets and smartphones.	CO2
7	Create a navigation menu using CSS to demonstrate the practical use of CSS styling and positioning.	CO2
8	Simulate a web shopping cart using JavaScript arrays and objects to handle dynamic data.	CO2
9	Construct a simple AJAX application to fetch data from the server without reloading the web page.	CO4

S.N	Lab Task	Mapped CO/COs
10	Introduce XML by creating a simple XML document that includes elements and attributes to store data about books or movies.	CO4
11	Use CSS Flexbox to design a flexible and efficient layout for a web page that adjusts content based on the screen size.	CO2
12	Develop a small project to parse XML data using JavaScript and display it in a structured format on a webpage.	CO4
13	Implement a basic web service using AJAX and SOAP to interact with external data sources.	CO4
14	Design and implement a website using all learned technologies to demonstrate effective web design principles and user-centric interfaces.	CO3
15	Enhance a web page by integrating interactive elements using DHTML to improve user experience.	CO3
16	Create a sitemap and a wireframe for a proposed website to illustrate the planning phase of web design.	CO3
17	Develop a user login system using HTML forms, JavaScript validation, and CSS styling.	CO2
18	Utilize JavaScript to create interactive sliders and content tabs that enhance the dynamic functionality of a web page.	CO2
19	Implement an interactive web-based calendar using JavaScript and AJAX to manage events and appointments.	CO2, CO4
20	Create an XML schema to validate the structure of an XML document used in a web application.	CO4
1	**Responsive Web Design Project:** Develop a fully responsive portfolio website that showcases a variety of media queries and CSS styles to ensure proper display on all devices.	CO2, CO3
2	**JavaScript Game:** Create an interactive web-based game using JavaScript and HTML5 canvas that includes event handling and real-time updates.	CO2
3	**Web Service Integration:** Design a web application that consumes multiple web services to provide a unified functionality, such as a weather forecast combined with local events.	CO4
4	**AJAX-driven Social Media Feed:** Implement a social media feed that uses AJAX to load content dynamically, simulating real-world social media platforms.	CO4
5	**XML Data Management:** Create a web application that uses XML to manage data (like a content management system), includes creating, editing, and deleting XML content.	CO4

Online Learning Resources

- **W3Schools:** Comprehensive tutorials and references on web development languages including HTML, CSS, JavaScript.
  https://www.w3schools.com/
- **MDN Web Docs:** Resources for developers, by developers, with documentation and tutorials on web technologies.
  https://developer.mozilla.org/en-US/
- **Codecademy:** Interactive platform offering web development courses in HTML, CSS, and JavaScript.
  https://www.codecademy.com/learn/paths/web-development
- **freeCodeCamp:** Learn to code for free with interactive lessons and build projects along the way.
  https://www.freecodecamp.org/

Programming in Python Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Programming in Python Lab	ENSP155	0-0-2	1
Type of Course:	Minor		

Defined Course Outcomes

COs	Statements
CO 1	Developing Python programs with a focus on secure coding practices and fundamental control structures.
CO 2	Implementing Python programs utilizing various data structures such as lists, dictionaries, tuples, and sets.
CO 3	Work withing file operations and regular expressions to securely handle and process data.
CO 4	Implementing Object-Oriented Programming concepts such as encapsulation, inheritance, and polymorphism in Python.

Proposed Lab Experiments

Ex. No	Experiment Title	Mapped CO/COs
1	Develop Python programs to demonstrate control structures, focusing on secure coding practices.	CO 1
2	Implement Python programs to work with lists, focusing on secure data handling and manipulation.	CO 2
3	Implement Python programs to work with dictionaries, focusing on key-value pair management.	CO 2
4	Develop Python programs using tuples, focusing on immutability and its applications.	CO 2
5	Implement functions in Python, emphasizing secure input validation and error handling.	CO 1
6	Work with file operations in Python, including reading and writing files, and manipulating file pointers.	CO 3
7	Develop Python programs using regular expressions for pattern matching and text processing.	CO 3
8	Create and implement classes and objects in Python, demonstrating OOP concepts like encapsulation and inheritance.	CO 4
9	Develop Python programs with polymorphism, method overloading, and method overriding.	CO 4
10	Create Python programs that implement exception handling and secure error management.	CO 1
11	Develop programs using constructors and destructors in Python to manage object lifecycles securely.	CO 4
12	Demonstrate practical use of operator overloading and special methods in Python (e.g., __str__(), __repr__()).	CO 4

Online Learning Resources

- **Codecademy:** Interactive platform offering Python programming exercises.
  https://www.codecademy.com/learn/learn-python-3
- **Coursera:** Courses on Python programming fundamentals.
  https://www.coursera.org/specializations/python-for-everybody
- **Kaggle:** Online platform for Python projects, tutorials, and challenges.
  https://www.kaggle.com/learn/python
- **GitHub:** Explore and contribute to open-source Python projects.
  https://github.com/

Fundamentals of Software Engineering

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Fundamentals of Software Engineering	ENBC105	4-0-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course aims to introduce students to the principles and practices of software engineering, covering fundamental concepts, requirement analysis, design, project management, UML, testing, and maintenance. The course is divided into 4 units:

1. Introduction to Software Engineering
2. Software Requirement Analysis, Design & Construction
3. Software Project Management and UML
4. Software Testing & Maintenance

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the fundamental concepts of software engineering, including software process models and quality concepts.
CO 2	Analyzing and document software requirements using various modeling techniques and design principles.
CO 3	Managing software projects using estimation techniques, quality management, and UML diagrams.
CO 4	Applying software testing strategies and maintain software using different testing techniques and maintenance models.

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

Course Outline

Unit Number: 1	Title: Introduction to Software Engineering	No. of hours: 10
Content:		
• Introduction to Software Engineering • Software Evolution • Software Characteristics • Software Crisis: Problem and Causes • Software process models: Waterfall, Incremental, Evolutionary, Agile • Software quality concepts and process improvement • Software process capability maturity models • Personal Software Process and Team Software Process • Overview of Agile Process		
Unit Number: 2	**Title: Software Requirement Analysis, Design & Construction**	**No. of hours: 10**
Content:		
• Problem Analysis • Requirement elicitation and Validation • Requirements modeling: Scenarios, Information and analysis classes, flow and behavioral modeling • Documenting Software Requirement Specification (SRS) • System design principles: levels of abstraction, separation of concerns, information hiding, coupling and cohesion • Structured design, object-oriented design, event driven design, component-level design, test driven design, aspect oriented design • Design patterns • Coding Practices: Techniques, Refactoring		
Unit Number: 3	**Title: Software Project Management and UML**	**No. of hours: 10**
Content:		

- Software Project Management: Scope, time, and cost estimation
- Quality Management
- Plan for software Quality Control and Assurance
- Earned Value Analysis
- UML: UML Structural Diagrams, UML Behavioural Diagrams

Unit Number: 4	**Title: Software Testing & Maintenance**	**No. of hours: 10**

Content:

- Testing: Levels of Testing, Functional Testing, Structural Testing
- Test Plan, Test Case Specification, Software Testing Strategies
- Verification & Validation
- Unit, Integration Testing
- Top Down and Bottom-Up Integration Testing
- Alpha & Beta Testing
- White box and black box testing techniques
- System Testing and Debugging
- Software Maintenance: Maintenance Process, Maintenance Models
- Reverse Engineering
- Software Re-engineering

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** Engage students through detailed explanations of software engineering principles, process models, and design techniques. Interactive sessions include real-world examples, case studies, and discussions on current industry practices.
- **Hands-On Labs:** Provide practical lab sessions where students apply software engineering concepts such as requirement analysis, UML modeling, and software testing. These labs help bridge the gap between theory and practice, enabling students to work on mini-projects and case studies.
- **Group Projects:** Encourage collaborative learning by assigning group projects that require students to manage software development from requirements gathering through design, implementation, and testing. These projects simulate real-world

software development environments, promoting teamwork and project management skills.

- **In-Class Problem Solving:** Facilitate problem-solving sessions where students tackle real-world software engineering challenges, applying concepts learned in class to practical scenarios.
- **Immediate Feedback:** Provide quizzes, code reviews, and in-class exercises to offer timely feedback on students' understanding of software engineering principles and methodologies.

Outside Classroom Learning Experience

- **Assignments & Case Studies:** Assign tasks that challenge students to apply software engineering methodologies to analyze and solve complex software development problems. Case studies focus on software process improvement, design patterns, and testing strategies.
- **Collaborative Projects:** Promote teamwork on larger projects outside the classroom, where students can simulate the complete software development lifecycle, from gathering requirements to testing.
- **Online Discussions:** Facilitate online forums where students can discuss software engineering concepts, share experiences, and troubleshoot project challenges.
- **Self-Study:** Encourage students to explore advanced software engineering topics and tools independently, further enhancing their understanding of real-world practices.
- **Peer Reviews:** Organize peer reviews where students assess each other's project deliverables, providing constructive feedback on software design, code quality, and adherence to engineering best practices.
- **Practice Problems:** Provide access to additional practice exercises focused on software design, modeling, and testing, allowing students to hone their skills outside the classroom.

Text Books

- "Software Engineering" by Ian Sommerville
- "Software Engineering: A Practitioner's Approach" by Roger S. Pressman

Reference Books

- "Fundamentals of Software Engineering" by Rajib Mall
- "Software Engineering" by K.K. Aggarwal and Yogesh Singh
- "Object-Oriented Software Engineering" by Ivar Jacobson
- "Software Engineering Concepts" by Richard Fairley
- "Software Engineering: Theory and Practice" by Shari Lawrence Pfleeger and Joanne M. Atlee

Additional Readings

Self-Learning Components:

1. Link to SEI Software Engineering Institute: https://www.sei.cmu.edu/
2. Link to IEEE Software Engineering Standards: https://standards.ieee.org/
3. Link to Agile Alliance resources: https://www.agilealliance.org/
4. Link to NPTEL Software Engineering course: https://onlinecourses.nptel.ac.in/noc21_cs24/
5. Link to Coursera Software Engineering courses: https://www.coursera.org/courses?query=software%20engineering

Essentials of Computer Science

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Essentials of Computer Science	—-	0-0-0	2
Type of Course:	SEC (Self-Paced Course)		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the Fundamentals of Computer Science, covering essential concepts and techniques used in computing and programming. The course emphasizes both theoretical understanding and practical application and is divided into 4 units:

1. Introduction to Computer Science and Programming Basics
2. Data Structures and Algorithms
3. Software Development and Engineering
4. Advanced Topics and Applications

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Developing a basic understanding of the foundational principles and history of computer science.
CO 2	Developing a basic understanding of fundamental programming skills using languages such as Python, C++, and JavaScript.
CO 3	Developing a basic understanding of essential data structures and algorithms.
CO 4	Developing a basic understanding of software development and engineering principles, including web development, databases, and cybersecurity basics.

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

Course Outline

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

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

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

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

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

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

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

- Web Development
  - HTML, CSS, and JavaScript
  - Client-server architecture
  - Introduction to web frameworks
- Databases
  - SQL and relational databases
  - NoSQL databases
  - Basic database design and querying
- Cybersecurity Basics
  - Principles of cybersecurity
  - Common threats and vulnerabilities
  - Basic encryption and security protocols
- Latest Technologies and Careers in Computer Science
  - Overview of latest technologies (e.g., AI, blockchain, IoT, cloud computing)
  - Emerging domains in computer science
  - Various careers in computer science
  - Skills and qualifications needed for different career paths
- Introduction to Machine Learning
  - Basic concepts of machine learning
  - Types of machine learning (supervised, unsupervised, reinforcement learning)
  - Introduction to neural networks

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** Engage students with foundational concepts in computer science through interactive discussions, real-world examples, and demonstrations of basic programming, data structures, and algorithms.
- **Hands-On Labs:** Provide practical sessions where students write and run simple programs, implement basic data structures, and solve algorithmic problems. These labs reinforce theoretical knowledge through hands-on experience with programming languages like Python, C, and JavaScript.
- **In-Class Problem Solving:** Facilitate coding exercises and problem-solving sessions to allow students to immediately apply theoretical concepts in the classroom setting.

- **Group Projects:** Encourage collaborative group projects where students design and implement solutions to real-world problems using the concepts of algorithms, data structures, and basic programming.
- **Immediate Feedback:** Provide quizzes and in-class coding challenges to offer real-time feedback on students' understanding of computer science concepts.

Outside Classroom Learning Experience

- **Self-Paced Learning:** Encourage students to explore course materials at their own pace, using self-paced modules that cover key concepts such as software development, networks, and operating systems. This approach allows for personalized learning experiences.
- **Assignments & Case Studies:** Assign practical tasks that challenge students to apply their knowledge to solve problems, such as implementing sorting algorithms or designing basic web applications. Case studies provide context and deepen understanding of computer science fundamentals.
- **Collaborative Projects:** Promote teamwork on larger, more complex projects outside the classroom where students apply foundational computer science concepts to solve problems.
- **Online Discussions:** Facilitate online forums where students can collaborate, discuss course topics, and troubleshoot programming challenges together.
- **Peer Review:** Set up peer review activities where students critique and provide feedback on each other's code and assignments, fostering improvement in programming practices.
- **Practice Problems:** Provide additional problem sets and programming challenges for students to practice independently and strengthen their grasp of essential computer science concepts.

Online Learning Modules

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

1. CS50's Introduction to Computer Science by Harvard University
   Platform: Harvard Online Learning
   Link: https://cs50.harvard.edu/
2. Computer Science 101 by Stanford University
   Platform: Stanford Online
   Link: https://online.stanford.edu/courses/sohs-ydkcs101-computer-science-101
3. Fundamentals of Computing by Rice University
   Platform: Coursera
   Link: https://www.coursera.org/specializations/computer-fundamentals
4. Introduction to Computer Science
   Platform: Udemy
   Link: https://www.udemy.com/course/introduction-to-computer-science/

5. Computer Science 101 - Computers & Programming for Beginners
   Platform: Udemy
   Link: https://www.udemy.com/course/computer-science-101-computers-programming-for

6. IT Fundamentals - Everything you need to know about IT
   Platform: Udemy
   Link: https://www.udemy.com/course/it-fundamentals-everything-you-need-to-know-al

7. Master Computer Fundamentals Course-Beginner to Intermediate
   Platform: Udemy
   Link: https://www.udemy.com/course/master-computer-fundamentals-skills-beginner-t

Please Note:

1. **Enrollment:** Students must enroll in any one of the above specified courses on their respective platforms.

2. **Certification:** Students will be required to submit the course completion certificate as an outcome.

3. **Self-paced:** Students will be required to complete any of the certifications on their own. No physical classes shall be conducted.

4. **Assignments:** Complete all assignments, quizzes, and problem sets as required by each course.

Semester: 2

Network Defence Essentials

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Network Defence Essentials	ENSP114	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	Basic Networking and Introduction to Cybersecurity		

Course Perspective: This course aims to provide students with a comprehensive understanding of network defense mechanisms. It covers fundamental concepts of network security, defense strategies, tools, and best practices essential for safeguarding network infrastructures. The course is divided into 4 units:

1. Fundamentals of Network Security
2. Network Defense Techniques and Tools
3. Intrusion Detection and Prevention Systems (IDS/IPS)
4. Advanced Network Defense and Incident Response

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the principles of network security and various defense mechanisms.
CO 2	Implementing and configure network defense tools such as firewalls, IDS/IPS, and honeypots.
CO 3	Analyzing network traffic and identify potential security threats using various tools and techniques.
CO 4	Developing and implement network defense strategies and incident response plans.

A student is expected to have learned concepts and demonstrated/developed abilities or skills related to network defense essentials by the end of the course.

Course Outline

Unit Number: 1	Title: Fundamentals of Network Security	No. of hours: 10
Content:		
• Introduction to Network Security: Key Concepts and Principles • Types of Network Attacks: Overview and Case Studies • Network Security Protocols: SSL/TLS, IPsec, SSH • Firewalls: Types, Configuration, and Best Practices • Virtual Private Networks (VPNs) and Secure Communication • Security Policies and Access Control Mechanisms		
Unit Number: 2	**Title: Network Defense Techniques and Tools**	**No. of hours: 10**
Content:		
• Network Defense Architectures: Defense in Depth, Zero Trust • Firewalls and Their Role in Network Defense • Implementing and Configuring Intrusion Detection Systems (IDS) • Honeypots and Honeynets: Deception Techniques for Network Defense • Network Monitoring Tools: Wireshark, Snort, Nagios • Security Information and Event Management (SIEM) Systems		
Unit Number: 3	**Title: Intrusion Detection and Prevention Systems (IDS/IPS)**	**No. of hours: 10**
Content:		
• Overview of IDS/IPS: Concepts, Types, and Deployment Strategies • Signature-Based vs. Anomaly-Based Detection • Implementing IDS/IPS in a Network Environment • Analyzing IDS/IPS Alerts and Logs • Case Studies: Successful Implementation of IDS/IPS • Challenges in IDS/IPS Implementation and How to Overcome Them		
Unit Number: 4	**Title: Advanced Network Defense and Incident Response**	**No. of hours: 10**
Content:		

- Advanced Threat Detection Techniques: Machine Learning in Network Defense
- Incident Response Planning and Execution
- Forensics in Network Defense: Tools and Techniques
- Case Studies: Responding to Network Breaches
- Developing a Comprehensive Network Defense Strategy
- Best Practices for Continuous Network Defense and Monitoring

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** The course incorporates interactive lectures on network security fundamentals, defense techniques, and advanced strategies. These lectures encourage active participation and discussion, helping students gain a solid understanding of the core concepts in network defense.
- **Hands-On Lab Sessions:** Students will engage in practical lab sessions that involve configuring firewalls, setting up IDS/IPS systems, and working with network defense tools like Wireshark, Snort, and Nagios. These sessions provide hands-on experience in implementing network defense mechanisms.
- **Case Study Analysis:** Through case studies of real-world network breaches and defense strategies, students will explore the effectiveness of various tools and techniques. This analysis helps students connect theory to practical applications.
- **Guest Lectures:** Industry experts will deliver guest lectures on the latest trends in network defense, including the use of machine learning and AI in threat detection. These sessions provide insights into cutting-edge network defense strategies.

Outside Classroom Learning Experience

- **Project-Based Learning:** Students will work on projects that simulate real-world network defense scenarios, such as setting up secure networks, configuring firewalls, and deploying IDS/IPS systems. These projects allow students to apply their knowledge in practical, real-world situations.
- **Collaborative Group Work:** Team projects will encourage students to collaborate and share ideas while solving complex network security challenges. This fosters teamwork and improves problem-solving skills.
- **Self-Study and Research:** Students are encouraged to engage in self-study and research on advanced network defense topics such as machine learning in threat detection and forensics in incident response. This independent learning helps students stay updated with the latest developments in network security.

- **Continuous Feedback and Iterative Learning:** Students will receive continuous feedback on assignments and lab work, allowing them to improve their skills. Regular peer reviews and instructor feedback ensure that students refine their understanding and practical abilities in network defense.

Text Books

- T1: "Network Security Essentials: Applications and Standards" by William Stallings
- T2: "Firewalls and Internet Security: Repelling the Wily Hacker" by William R. Cheswick, Steven M. Bellovin, and Aviel D. Rubin

Reference Books

- R1: "Security Information and Event Management (SIEM) Implementation" by David Miller
- R2: "Applied Network Security Monitoring: Collection, Detection, and Analysis" by Chris Sanders and Jason Smith

Online Learning Resources

- **Cybrary:** Online platform offering courses on network defense, IDS/IPS, and SIEM.
  https://www.cybrary.it/course/network-defense/
- **Coursera:** Specializations in network security and defense techniques offered by top universities.
  https://www.coursera.org/specializations/network-security
- **SANS Institute:** Comprehensive cybersecurity courses including network defense essentials.
  https://www.sans.org/courses/
- **Cisco Networking Academy:** Courses on network security, defense, and monitoring.
  https://www.netacad.com/

Introduction to Discrete Structures

Program Name:	B.Tech Computer Science and Engineering		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Discrete Structures	ENBC102	3-1-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the fundamental concepts of discrete structures, focusing on set theory, logic, relations, graph theory, combinatorics, and number theory. The course is divided into 4 units:

1. Set Theory and Logic
2. Relations and Graph Theory
3. Combinatorics and Discrete Structures
4. Number Theory and Cryptography

The ing (COs)

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

COs	**Statements**
CO 1	Understanding the basic concepts of set theory and logic, including set operations and logical connectives.
CO 2	Applying relations and basic graph theory concepts to solve problems.
CO 3	Utilizing combinatorial principles and discrete structures to analyze problems.
CO 4	Understanding basic number theory and cryptography concepts and their applications.

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

Course Outline

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

Content:

- **Set Theory**
  - Basic Concepts: Notations and terminology (union, intersection, complement), Types of sets (finite, infinite, empty, universal), Multisets (elements with multiplicity)
  - Ordered Pairs and Cartesian Product: Definition of ordered pairs, Properties of Cartesian product
  - Set Algebra and Proofs: Set operations (union, intersection, difference), Proofs of set identities (De Morgan's laws, distributive properties)
- **Logic**
  - Propositional Logic: Syntax and semantics, Truth tables for logical connectives (AND, OR, NOT), Tautologies and contradictions
  - Predicate Logic: Quantifiers (universal and existential), Predicate calculus, Proofs using mathematical induction

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

Content:

- **Relations**
  - Representation and Properties: Matrices, graphs, and directed graphs, Reflexive, symmetric, and transitive relations
  - Equivalence Relations and Partitions: Equivalence classes, Equivalence partitions
- **Graph Theory Basics**
  - Definitions: Vertices, edges, degree
  - Types of graphs: Simple, directed, weighted
  - Graph representations: Adjacency matrix, adjacency list
- **Graph Algorithms**
  - Depth-First Search (DFS), Breadth-First Search (BFS), Shortest path algorithms (Dijkstra's)

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

Content:

- **Combinatorics**
  - Counting Principles: Product rule, sum rule, Permutations and combinations, Binomial coefficients
  - Inclusion-Exclusion Principle: Solving problems with overlapping sets
- **Discrete Structures**
  - Trees and Recurrence Relations: Tree properties (rooted, binary), Solving linear recurrence relations
  - Finite State Machines and Regular Languages: Deterministic Finite Automata (DFA), Regular expressions, Regular languages
  - Formal Languages and Grammars: Context-free grammars

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

Content:

- **Number Theory**
  - Divisibility and Modular Arithmetic: Greatest common divisor (GCD), Modular inverses
  - Congruences: Solving congruences
- **Cryptography**
  - Symmetric-Key Cryptography: Basic concepts
  - Public-Key Cryptography: Basic concepts

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** Engage students with fundamental concepts of discrete structures, including set theory, logic, and graph theory, through interactive lectures. These sessions include problem-solving activities and discussions that help students grasp abstract concepts.
- **Hands-On Problem Solving:** Provide students with ample opportunities to solve problems in set theory, combinatorics, and graph theory during class and in dedicated problem-solving sessions. These exercises are designed to reinforce theoretical understanding through practical application.
- **Group Projects:** Encourage collaboration by assigning group projects that require students to apply concepts from discrete structures to real-world scenarios. Projects may include designing algorithms, working on graph-related problems, or exploring combinatorial designs.

- **Immediate Feedback:** Conduct quizzes and in-class exercises that provide students with immediate feedback on their grasp of discrete structures concepts and their problem-solving approaches.

Outside Classroom Learning Experience

- **Assignments & Case Studies:** Assign regular homework and case studies that challenge students to apply discrete mathematics concepts to solve complex problems. Case studies focus on applications in computer science, such as algorithm design and cryptography.
- **Collaborative Projects:** Encourage students to work together on larger projects outside the classroom, where they can apply discrete structures concepts to solve practical problems in fields like network theory and cryptographic algorithms.
- **Online Discussions:** Facilitate online forums where students can discuss discrete structures problems, share ideas, and collaborate on assignments and projects.
- **Peer Review:** Organize peer review activities where students critique and provide feedback on each other's solutions to complex problems, promoting deeper understanding and refining problem-solving skills.
- **Self-Study:** Encourage self-paced learning with additional problem sets and study materials that allow students to explore advanced topics in discrete structures, such as Boolean algebra and advanced graph theory.
- **Practice Problems:** Provide additional exercises for students to work on independently, reinforcing their understanding of key concepts in discrete structures.

Text Books

- "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
- "Discrete Mathematics" by Seymour Lipschutz and Marc Lipson

Reference Books

- "Discrete Mathematics with Applications" by Susanna S. Epp
- "Discrete Mathematics" by Richard Johnsonbaugh

Additional Readings

Self-Learning Components:

1. Link to Discrete Mathematics course on NPTEL: https://nptel.ac.in/courses/106/106/106106094/
2. Link to Discrete Mathematics on Coursera: https://www.coursera.org/courses?query=discrete%20mathematics
3. Link to Graph Theory resources: https://www.graphclasses.org/

4. Link to Set Theory tutorials: https://www.tutorialspoint.com/discrete_mathematics/discrete_mathematics_set_theory.htm

5. Link to Combinatorics lectures: https://www.math.cmu.edu/~bkell/21110-2010s/lectures.shtml

Basics of Operating Systems

Program Name:	B.Tech Computer Science and Engineering		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Basics of Operating Systems	ENBC104	3-1-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course provides an introduction to 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 basic concepts and structure of operating systems.
CO 2	Managing processes, threads, and CPU scheduling in operating systems.
CO 3	Applying synchronization techniques and handle deadlocks.
CO 4	Managing memory, virtual memory, and file systems in operating systems.

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, Process and CPU Scheduling	No. of hours: 10
Content:		
• Introduction: Definition, Role, Types of Operating Systems, Batch Systems, Multiprogramming, Time–sharing • Operating system structure, components, and services • Processes: Concept, Scheduling, Operations, Cooperating Processes, Threads • CPU Scheduling: Basic Concepts, Scheduling Criteria, Scheduling Algorithms		
Unit Number: 2	**Title: Threads, Synchronization, Deadlock and Memory Management**	**No. of hours: 10**
Content:		
• Threads: Overview, Benefits, User and Kernel Threads, Multithreaded Models • Synchronization: Critical-Section Problem, Semaphores, Classical Problems, Monitors • Deadlocks: Characterization, Handling Deadlocks, Prevention, Avoidance, Detection, Recovery • Memory Management: Logical vs. Physical Address space, Swapping, Contiguous allocation, Paging, Segmentation		
Unit Number: 3	**Title: Virtual Memory, Device Management and Secondary-Storage Structure**	**No. of hours: 10**
Content:		
• Virtual Memory: Demand Paging, Page-replacement Algorithms, Thrashing • Device Management: Techniques, Dedicated Devices, Shared Devices, Buffering, Device Allocation • Secondary-Storage Structure: Disk Structure, Disk Scheduling, Disk Management, Swap Space Management		
Unit Number: 4	**Title: File-System Interface, Implementation and Security**	**No. of hours: 10**
Content:		

- File-System Interface: File Concept, Access Methods, Directory Structure
- File-System Implementation: Structure, Basic File System, Allocation Methods, Free-Space Management
- Security: Problems, Goals, Access matrix, Authentication, Program threats, System threats

Learning Experience

Classroom Learning Experience

- **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.
- **Immediate Feedback:** Provide in-class quizzes, coding exercises, and interactive problem-solving sessions to give students timely feedback on their understanding of operating system concepts.

Outside Classroom Learning Experience

- **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.
- **Collaborative Projects:** Encourage students to work together on larger projects outside the classroom, where they can apply operating system concepts to design and implement components such as memory managers or file systems.
- **Online Discussions:** Facilitate online forums where students can discuss operating system topics, share solutions to challenging problems, and collaborate on project ideas.
- **Peer Review:** Organize peer review activities where students critique and provide feedback on each other's work on operating system projects, fostering improvement in design and implementation skills.

- **Self-Study:** Encourage self-paced learning by providing additional reading materials, practice problems, and exercises to explore advanced operating system topics, such as virtual memory and deadlock management.
- **Practice Problems:** Provide supplementary problem sets and challenges for students to complete independently, helping reinforce core concepts in operating systems.

Text Books

- "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

Additional Readings

Self-Learning Components:

1. Link to Operating System course on NPTEL: https://nptel.ac.in/courses/106/106/106106144/
2. Link to Operating Systems on Coursera: https://www.coursera.org/courses?query=operating%20systems
3. Link to Operating Systems resources: https://www.os-book.com/
4. Link to Operating Systems tutorials: https://www.tutorialspoint.com/operating_system/index.htm
5. Link to Operating Systems lectures: https://ocw.mit.edu/courses/electrical-engineering-6-828-operating-system-engineering-fall-2012/

Concepts of Object Oriented Programming Using C++

Program Name:	B.Tech Computer Science and Engineering		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Concepts of Object Oriented Programming Using C++	ENBC106	3-1-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the fundamental concepts of object-oriented programming using C++, focusing on classes and objects, inheritance, polymorphism, and advanced features of C++. The course is divided into 4 units:

1. Foundations of Object-Oriented Programming
2. Classes and Objects
3. Inheritance and Polymorphism
4. Advanced C++ Features

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the basic concepts and principles of object-oriented programming.
CO 2	Implementing classes and objects in C++ and manage memory using constructors and destructors.
CO 3	Applying inheritance and polymorphism concepts to enhance code reusability and flexibility.
CO 4	Utilizing advanced C++ features such as templates and exception handling.

A student is expected to have learned concepts and demonstrated abilities or skills related to object-oriented programming using C++ at the end of the course.

Course Outline

Unit Number: 1	Title: Foundations of Object-Oriented Programming	No. of hours: 10

Content:

- Programming Approaches: Procedure-Oriented Approach vs. Object-Oriented Approach
- Introduction to C++: Basic syntax and structure of a C++ program, Data Types and Variables, Operators and Expressions, Control Structures, Functions, Arrays and Strings, Pointers
- Basic Concepts of Object-Oriented Programming: Objects and Classes, Principles of OOP: Abstraction, Encapsulation, Inheritance, Polymorphism, Dynamic Binding, and Message Passing
- Characteristics of Object-Oriented Languages: Benefits and features of OOP languages

Unit Number: 2	Title: Classes and Objects	No. of hours: 10

Content:

- Abstract Data Types and Classes: Concept of abstract data types, Objects and classes, attributes, and methods
- C++ Class Declaration: Declaring classes in C++, State, identity, and behaviour of objects
- Objects: Local Objects and Global Objects, Scope resolution operator
- Functions in C++: Friend Functions, Inline Functions
- Constructors and Destructors: Instantiation of objects, Types of constructors (default, parameterized, copy), Static Class Data, Array of Objects, Constant member functions and objects
- Memory Management Operators: New and delete operators for dynamic memory allocation

Unit Number: 3	Title: Inheritance and Polymorphism	No. of hours: 10

Content:

- Inheritance: Types of inheritance (single, multiple, hierarchical), Access specifiers: public, private, and protected, Abstract Classes
- Advanced Inheritance Concepts: Aggregation and composition vs. classification hierarchy
- Polymorphism: Types of Polymorphism (compile-time and run-time), Function Overloading, Operator Overloading
- Pointers and Virtual Functions: Pointer to objects, this pointer, Virtual Functions

Unit Number: 4	**Title: Advanced C++ Features**	**No. of hours: 10**

Content:

- Strings and Streams: Manipulating strings, Streams and file handling
- Operators and Error Handling: Overloading operators, Error handling during file operations, Formatted I/O
- Generic Programming: Function templates, Class templates
- Exception Handling: Throwing an exception, The try block, Catching an exception

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** Introduce students to the fundamental concepts of object-oriented programming through engaging lectures. These sessions will cover key topics such as classes, objects, inheritance, and polymorphism, supported by real-world examples and problem-solving activities to enhance understanding.
- **Hands-On Programming Labs:** Provide practical lab sessions where students write and debug C++ programs that implement object-oriented concepts. Labs will focus on creating classes, utilizing inheritance, and applying polymorphism, allowing students to gain hands-on experience with C++ programming.
- **Group Projects:** Encourage collaborative learning by assigning group projects that require students to design and implement a complete object-oriented system in C++. Projects will simulate real-world scenarios, fostering teamwork and problem-solving skills.
- **Immediate Feedback:** Provide quizzes, coding challenges, and in-class exercises to give real-time feedback on students' understanding of object-oriented programming concepts.

Outside Classroom Learning Experience

- **Assignments & Case Studies:** Assign regular programming assignments and case studies that challenge students to apply object-oriented principles to solve complex problems. Case studies will include the design and implementation of software systems using C++, focusing on code reusability, maintainability, and efficiency.
- **Collaborative Projects:** Promote teamwork on larger projects outside the classroom, where students work together to design and implement object-oriented systems, applying C++ concepts to real-world challenges.
- **Online Discussions:** Facilitate online discussion forums where students can collaborate on coding problems, share insights on object-oriented design, and troubleshoot project challenges together.
- **Peer Review:** Organize peer code review sessions where students critique and provide feedback on each other's C++ programs, improving code quality and deepening their understanding of object-oriented principles.
- **Self-Study:** Encourage self-paced exploration of advanced object-oriented concepts and C++ libraries through additional reading materials, tutorials, and exercises.
- **Practice Problems:** Provide supplementary coding challenges and problem sets for students to work on independently, reinforcing their understanding of key object-oriented programming concepts in C++.

Text Books

- "C++ Programming Language" by Bjarne Stroustrup
- "Object-Oriented Programming in C++" by Robert Lafore

Reference Books

- "C++ Primer" by Stanley B. Lippman, Josée Lajoie, and Barbara E. Moo
- "Effective C++" by Scott Meyers

Additional Readings

Self-Learning Components:

1. Link to C++ Language Reference: https://en.cppreference.com/w/
2. Link to C++ Tutorials on GeeksforGeeks: https://www.geeksforgeeks.org/c-plus-plus/
3. Link to C++ Programming course on NPTEL: https://nptel.ac.in/courses/106/106/106106093/

4. Link to Object-Oriented Programming in C++ on Coursera: https://www.coursera.org/courses?query=object%20oriented%20programming%20in%20c++

5. Link to C++ lectures: https://ocw.mit.edu/courses/electrical-engineering-and-compute 6-096-introduction-to-c-january-iap-2011/

Network Defence Essentials Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Network Defence Essentials Lab	ENSP166	0-0-2	1
Type of Course:	Minor		

Defined Course Outcomes

COs	Statements
CO 1	Implementing and configure network defense tools such as firewalls, IDS/IPS, and honeypots.
CO 2	Analyzing network traffic using tools like Wireshark and Snort to detect potential security threats.
CO 3	Developing and deploy secure communication channels using VPNs and other network security protocols.
CO 4	Implementing advanced network defense techniques and develop incident response strategies.

Proposed Lab Experiments

Ex. No	Experiment Title	Mapped CO/COs
1	Configuring a basic firewall to filter network traffic based on IP address and port numbers.	CO 1
2	Setting up and configuring an Intrusion Detection System (IDS) using Snort to monitor network traffic.	CO 1, CO 2
3	Deploying a honeypot using open-source tools like Honeyd to attract and analyze potential attackers.	CO 1
4	Using Wireshark to capture and analyze network packets, identifying security threats and anomalies.	CO 2
5	Implementing a Virtual Private Network (VPN) to secure communication between two network segments.	CO 3
6	Configuring SSL/TLS on a web server to secure HTTP traffic and prevent Man-in-the-Middle (MITM) attacks.	CO 3
7	Setting up IPsec for secure data transmission over an unsecured network.	CO 3
8	Developing and testing a network defense strategy using multiple layers of security (Defense in Depth).	CO 4
9	Conducting a vulnerability assessment and penetration testing on a simulated network environment.	CO 4
10	Creating an incident response plan and executing it in response to a simulated network breach.	CO 4

Online Learning Resources

- **Cybrary:** Online platform offering courses on network defense, IDS/IPS, and SIEM.
  https://www.cybrary.it/course/network-defense/
- **Coursera:** Specializations in network security and defense techniques offered by top universities.
  https://www.coursera.org/specializations/network-security
- **SANS Institute:** Comprehensive cybersecurity courses including network defense essentials.
  https://www.sans.org/courses/
- **Cisco Networking Academy:** Courses on network security, defense, and monitoring.
  https://www.netacad.com/

Basics of Operating Systems Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Basics of Operating Systems Lab	ENBC152	0-0-2	1
Type of Course:	Major		

Defined Course Outcomes

COs	**Statements**
CO 1	Understanding the fundamental concepts and components of operating systems.
CO 2	Implementing process scheduling algorithms and understand process synchronization techniques.
CO 3	Managing memory allocation, including paging and segmentation.
CO 4	Handling device management and understand secondary storage structures.
CO 5	Implementing file system interfaces and basic security measures.

S.N	**Lab Task**	**Mapped CO/COs**
1	Install and configure an operating system (e.g., Linux, Windows).	CO1
2	Explore the structure and components of the installed operating system.	CO1
3	Implement a simple process scheduler using different scheduling algorithms (FCFS, SJF, Round Robin).	CO2
4	Simulate process synchronization using semaphores and monitor solutions.	CO2
5	Implement a program to demonstrate the critical section problem.	CO2
6	Develop a program to handle deadlock detection and recovery.	CO2
7	Create a memory management simulation to handle paging and segmentation.	CO3
8	Implement a virtual memory system with page replacement algorithms (FIFO, LRU).	CO3
9	Simulate disk scheduling algorithms (FCFS, SSTF, SCAN, C-SCAN).	CO4
10	Develop a program for file system operations including creation, deletion, and traversal of directories.	CO5
11	Implement file allocation methods (contiguous, linked, indexed).	CO5

S.N	Lab Task	Mapped CO/COs
12	Create a simulation of buffer management in device allocation.	CO4
13	Develop a program for swap space management.	CO4
14	Implement user authentication and access control mechanisms.	CO5
15	Simulate a basic file system with free-space management techniques.	CO5
16	Implement a program to handle system threats and security measures.	CO5
17	Develop a simple thread management program demonstrating multithreading.	CO2
18	Simulate a device management system including buffering and spooling.	CO4
19	Implement a program for inter-process communication using pipes and message queues.	CO2
20	Create a simple command-line interpreter (shell) that can execute basic commands.	CO1
1	**Process Scheduling Project:** Develop a comprehensive scheduler that implements and compares multiple scheduling algorithms, demonstrating their efficiency in different scenarios.	CO2
2	**Memory Management Project:** Create a memory management simulator that handles paging, segmentation, and virtual memory, providing detailed visualization and analysis.	CO3
3	**File System Project:** Design a basic file system that includes file creation, deletion, access control, and directory management with user authentication.	CO5
4	**Device Management Project:** Implement a device management module that handles multiple devices, their allocation, and management using different techniques.	CO4
5	**Security Project:** Develop a security module for an operating system that addresses authentication, access control, and threat detection, implementing various security protocols.	CO5

Online Learning Resources

- **GeeksforGeeks:** Tutorials and articles on operating system concepts and implementations.
  https://www.geeksforgeeks.org/operating-systems/
- **TutorialsPoint:** Comprehensive guides on operating system principles and practices.
  https://www.tutorialspoint.com/operating_system/index.htm

- **NPTEL:** Video lectures and course materials on operating system fundamentals. https://nptel.ac.in/courses/106/106/106106144/
- **Coursera:** Courses on operating systems from leading universities. https://www.coursera.org/courses?query=operating%20systems

Concepts of Object Oriented Programming Using C++ Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Concepts of Object Oriented Programming Using C++ Lab	ENBC154	0-0-2	1
Type of Course:	Major		

Defined ing

COs	**Statements**
CO 1	Understanding the fundamentals of object-oriented programming and its basic concepts.
CO 2	Developing programs using classes, objects, and various constructors and destructors.
CO 3	Implementing inheritance and polymorphism in C++ programs.
CO 4	Utilizing advanced C++ features like file handling, templates, and exception handling.

S.N	**Lab Task**	**Mapped CO/COs**
1	Write a C++ program to understand the basic syntax and structure of a C++ program.	CO1
2	Implement a program using data types, variables, and control structures.	CO1
3	Develop a program using functions and arrays.	CO1
4	Create a C++ program to demonstrate the use of pointers.	CO1
5	Write a program to implement classes and objects in C++.	CO2
6	Develop a program using constructors and destructors.	CO2
7	Implement a program to demonstrate the use of friend functions and inline functions.	CO2
8	Write a program to use static class data and constant member functions.	CO2
9	Develop a program to demonstrate dynamic memory allocation using new and delete operators.	CO2
10	Implement single and multiple inheritance in C++.	CO3
11	Create a program to demonstrate polymorphism using function overloading.	CO3

S.N	Lab Task	Mapped CO/COs
12	Develop a program to demonstrate operator overloading.	CO3
13	Write a program to use pointers to objects and virtual functions.	CO3
14	Implement a program for string manipulation using C++ string class.	CO4
15	Develop a program for file handling operations (read, write, append).	CO4
16	Create a program to handle exceptions using try, catch, and throw.	CO4
17	Implement a program to use function templates.	CO4
18	Write a program to use class templates.	CO4
19	Develop a program to demonstrate formatted I/O operations.	CO4
20	Create a program to handle errors during file operations.	CO4
1	**Bank Management System:** Develop a bank management system using OOP concepts like classes, objects, inheritance, and polymorphism.	CO2, CO3
2	**Library Management System:** Create a library management system that uses file handling for data storage and retrieval.	CO4
3	**Student Record System:** Design a student record system implementing dynamic memory allocation and exception handling.	CO2, CO4
4	**Inventory Management System:** Develop an inventory management system using templates and polymorphism.	CO3, CO4
5	**Online Shopping System:** Create an online shopping system that demonstrates all learned OOP concepts including inheritance, polymorphism, file handling, and exception handling.	CO2, CO3, CO4

Online Learning Resources

- **GeeksforGeeks:** Tutorials and articles on C++ and object-oriented programming concepts.
  https://www.geeksforgeeks.org/c-plus-plus/
- **TutorialsPoint:** Comprehensive guides on C++ programming and OOP principles.
  https://www.tutorialspoint.com/cplusplus/index.htm
- **NPTEL:** Video lectures and course materials on object-oriented programming using C++.
  https://nptel.ac.in/courses/106/105/106105151/
- **Coursera:** Courses on C++ programming and object-oriented principles from leading universities.
  https://www.coursera.org/courses?query=c++

Minor Project-I

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Minor Project-I	ENSI152	0-0-0	2
Type of Course:	Project		
Pre-requisite(s), if any:	NA		

Course Perspective

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

Duration: 6 weeks.

Project must focus on the following aspects:

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

Guidelines:

1. **Project Selection:**

- Choose a societal problem relevant to the field of computer science and engineering.
- Ensure the problem is specific and well-defined.

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

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

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

- Good literature review: 15 marks
- Moderate literature review: 10 marks
- Basic literature review: 5 marks
- Poor literature review: 0 marks

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

Total: 100 Marks

Course Outcomes

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

1. **Understand Societal Issues:**
   - Demonstrating a basic understanding of societal problems and relevant issues within the concerned domain.
2. **Critical Thinking:**
   - Thinking critically about formulated problems and existing solutions.
3. **Literature Review:**
   - Conducting comprehensive literature reviews and identify gaps in existing solutions.
4. **Documentation:**
   - Documenting findings and analysis in a well-structured and appropriate format.
5. **Presentation Skills:**
   - Presenting findings and analysis effectively, using clear and concise communication skills.
6. **Problem Analysis:**
   - Analyzing problems from various perspectives and evaluate the effectiveness of existing solutions.

7. **Professional Development:**

   - Developing skills in research, analysis, documentation, and presentation, contributing to overall professional growth.

Semester: 3

Introduction to Data Structures

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Data Structures	ENBC201	3-1-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the fundamental concepts of data structures, focusing on arrays, linked lists, stacks, queues, searching, sorting, trees, and graphs. The course is divided into 4 units:

1. Foundations of Data Structures
2. Linear Data Structures
3. Searching and Sorting
4. Trees and Graph Algorithms

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the basic concepts and principles of data structures.
CO 2	Implementing and manipulate linear data structures like arrays, linked lists, stacks, and queues.
CO 3	Applying searching and sorting algorithms to solve problems.
CO 4	Utilizing tree and graph algorithms to manage hierarchical and networked data.

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

Course Outline

Unit Number: 1	Title: Foundations of Data Structures	No. of hours: 9
Content:		
• Introduction: Abstract Data Type, Elementary Data Organization • Measuring efficiency of an Algorithm: Time and Space Complexity Analysis, Asymptotic notations • Arrays: Single and Multidimensional Arrays, Representation of Arrays: Row Major Order, and Column Major Order, Application of arrays		
Unit Number: 2	**Title: Linear Data Structures**	**No. of hours: 11**
Content:		
• Linked lists: Array and Dynamic Implementation of Single Linked Lists, Doubly Linked List, Circularly Linked List, Operations on a Linked List • Stack operations: Push & Pop, Array and Linked list implementation of Stack, Applications: Prefix and Postfix Expressions, Evaluation of postfix expression • Queue operations: Create, Add, Delete, full and empty queues, Array and linked implementation of queues, Circular queues		
Unit Number: 3	**Title: Searching and Sorting**	**No. of hours: 10**
Content:		
• Searching: Sequential search, Binary Search • Sorting: Insertion Sort, Selection Sort, Bubble Sort, Quick Sort, Merge Sort • Hashing: Hash Function, Hash Table, Collision Resolution Strategies		
Unit Number: 4	**Title: Trees and Graph Algorithms**	**No. of hours: 10**
Content:		
• Trees: Basic terminology, Binary Trees, Array and linked list implementation, Types of Binary Tree, Tree Traversal algorithms: Inorder, Preorder and Postorder • Graphs: Representation (Matrix and Linked), Traversals, Shortest path, Minimum Spanning Tree Algorithms		

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** Engage students with the foundational concepts of data structures through interactive lectures that cover topics such as arrays, linked lists, stacks, and queues. Use real-world examples to demonstrate the practical applications of data structures in solving computational problems.
- **Hands-On Programming Labs:** Provide practical lab sessions where students implement and manipulate various data structures in programming assignments. Labs will focus on developing proficiency in using arrays, linked lists, stacks, queues, and applying sorting and searching algorithms.
- **Group Projects:** Facilitate collaborative learning by assigning group projects that require students to design and implement efficient data structures for specific applications. Projects will simulate real-world scenarios where students must choose the appropriate data structures to optimize performance.
- **Immediate Feedback:** Provide in-class coding challenges, quizzes, and problem-solving sessions to offer timely feedback on students' understanding of data structure concepts.

Outside Classroom Learning Experience

- **Assignments & Case Studies:** Assign regular programming assignments and case studies that challenge students to apply data structure concepts to solve complex problems. Case studies will include scenarios that require the use of trees, graphs, and hashing techniques to manage data effectively.
- **Collaborative Projects:** Promote teamwork on larger projects outside the classroom, where students design and implement complex data structures, applying them to real-world problems to optimize performance.
- **Online Discussions:** Facilitate online forums where students can discuss data structure challenges, share solutions, and collaborate on projects.
- **Peer Review:** Organize peer code review sessions where students critique and provide feedback on each other's implementations of data structures, improving code quality and conceptual understanding.
- **Self-Study:** Encourage students to explore advanced data structure concepts and algorithms through additional resources and exercises to deepen their knowledge beyond the classroom material.
- **Practice Problems:** Provide supplementary coding challenges focused on data structure implementations and algorithm optimizations to reinforce key concepts.

Text Books

- "Data Structures Using C" by Reema Thareja
- "Fundamentals of Data Structures in C" by Ellis Horowitz, Sartaj Sahni, and Susan Anderson-Freed

Reference Books

- "Data Structures and Algorithms Made Easy" by Narasimha Karumanchi
- "Data Structures Using C and C++" by Yedidyah Langsam, Moshe Augenstein, and Aaron M. Tenenbaum

Additional Readings

Self-Learning Components:

1. Link to Data Structures course on NPTEL: https://nptel.ac.in/courses/106/106/106106127/
2. Link to Data Structures on Coursera: https://www.coursera.org/courses?query=data%20structures
3. Link to Data Structures resources: https://www.geeksforgeeks.org/data-structures/
4. Link to Data Structures tutorials: https://www.tutorialspoint.com/data_structures_algorithms/index.htm
5. Link to Data Structures lectures: https://ocw.mit.edu/courses/electrical-engineering-an 6-006-introduction-to-algorithms-fall-2011/

Fundamentals of Cyptography

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Fundamentals of Cyptography	ENSP207	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	Basic Mathematics and Introduction to Cybersecurity		

Course Perspective: This course provides a comprehensive introduction to the principles and practices of cryptography. It covers both classical and modern cryptographic techniques, with a focus on their application in securing digital communications. The course is divided into 4 units:

1. Classical Cryptography
2. Modern Cryptography
3. Public Key Cryptography
4. Cryptographic Protocols and Applications

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the fundamental principles of cryptography and its importance in cybersecurity.
CO 2	Analyzing and implement classical cryptographic techniques.
CO 3	Applying modern cryptographic algorithms such as symmetric and asymmetric encryption.
CO 4	Developing secure cryptographic protocols for real-world applications.

A student is expected to have learned concepts and demonstrated/developed abilities or skills related to cryptographic methods and their applications in cybersecurity by the end of the course.

Course Outline

Unit Number: 1	Title: Classical Cryptography	No. of hours: 10
Content:		
• Introduction to Cryptography: History and Overview • Substitution Ciphers: Caesar Cipher, Vigenère Cipher • Transposition Ciphers: Rail Fence Cipher, Columnar Transposition • Cryptanalysis of Classical Ciphers: Frequency Analysis, Brute Force Attack • Introduction to Cryptographic Algorithms and Key Concepts		
Unit Number: 2	**Title: Modern Cryptography**	**No. of hours: 10**
Content:		
• Symmetric Key Cryptography: DES, 3DES, and AES • Block Ciphers and Stream Ciphers: Modes of Operation • Hash Functions: MD5, SHA-1, SHA-256 • Message Authentication Codes (MAC): HMAC • Cryptographic Attacks: Side-Channel Attacks, Replay Attacks		
Unit Number: 3	**Title: Public Key Cryptography**	**No. of hours: 10**
Content:		
• Introduction to Public Key Cryptography: RSA, Diffie-Hellman • Digital Signatures: Concept, RSA Digital Signature, DSA • Public Key Infrastructure (PKI): Certificates, Certificate Authorities • Key Exchange Protocols: Diffie-Hellman, Elliptic Curve Cryptography (ECC) • Cryptographic Protocols: SSL/TLS, PGP		
Unit Number: 4	**Title: Cryptographic Protocols and Applications**	**No. of hours: 10**
Content:		

- Key Management and Distribution: Concepts and Challenges
- Secure Email Protocols: S/MIME, PGP
- Secure Communication Protocols: SSL/TLS, IPsec
- Cryptography in Blockchain and Cryptocurrency
- Case Studies: Real-World Cryptography Applications
- Emerging Trends in Cryptography: Post-Quantum Cryptography

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** The course provides interactive lectures covering classical and modern cryptographic techniques, with a focus on cryptographic principles, algorithms, and their application in cybersecurity. These sessions encourage active student participation and engagement.
- **Hands-On Cryptographic Exercises:** Practical lab exercises will be conducted where students implement cryptographic algorithms such as Caesar Cipher, RSA, AES, and SHA-256. This hands-on experience helps students understand the practical applications of cryptography.
- **Case Studies and Real-World Applications:** Students will analyze case studies on the use of cryptographic protocols in securing digital communications and e-commerce. These discussions will bridge the gap between theory and practice.
- **Collaborative Problem Solving:** Group activities will involve cryptanalysis exercises where students work together to decrypt messages using classical techniques like frequency analysis. This collaborative learning approach fosters teamwork and critical thinking.

Outside Classroom Learning Experience

- **Project-Based Learning:** Students will work on projects where they design and implement secure communication systems using cryptographic protocols. These projects will help students apply their knowledge to real-world scenarios and develop practical skills in encryption and key management.
- **Self-Study and Research:** Students are encouraged to independently explore advanced topics in cryptography, such as post-quantum cryptography and blockchain cryptography. This will help them stay updated with the latest developments in the field.
- **Peer Collaboration and Review:** Students will participate in peer review sessions where they analyze each other's cryptographic implementations and provide constructive feedback. This process promotes collaborative learning and helps refine cryptographic techniques.

- **Continuous Assessment and Feedback:** Regular quizzes, assignments, and project evaluations will be used to provide continuous feedback on student performance. This will allow students to improve their understanding and application of cryptographic concepts.

Text Books

- T1: "Cryptography and Network Security: Principles and Practice" by William Stallings
- T2: "Applied Cryptography: Protocols, Algorithms, and Source Code in C" by Bruce Schneier

Reference Books

- R1: "Understanding Cryptography: A Textbook for Students and Practitioners" by Christof Paar and Jan Pelzl
- R2: "Cryptography Engineering: Design Principles and Practical Applications" by Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno

Online Learning Resources

- **Coursera:** Specializations and courses in cryptography by top universities.
  https://www.coursera.org/specializations/cryptography
- **Khan Academy:** Introductory cryptography courses and resources.
  https://www.khanacademy.org/computing/computer-science/cryptography
- **EDX:** Courses on cryptography and security offered by institutions like MIT and Stanford.
  https://www.edx.org/course/subject/computer-science/cryptography
- **NIST:** National Institute of Standards and Technology resources on cryptographic standards.
  https://csrc.nist.gov/

Basics of Probability & Statistics

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Basics of Probability & Statistics	ENBC203	4-0-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the fundamental concepts of probability and statistics, focusing on basic probability, probability distributions, descriptive statistics, and inferential statistics. The course is divided into 4 units:

1. Foundations of Probability
2. Engineering Applications of Probability Distributions
3. Descriptive Statistics and Regression Analysis
4. Inferential Statistics for Engineers

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding basic probability concepts and calculate probabilities for different events.
CO 2	Analyzing and apply different probability distributions in various scenarios.
CO 3	Using descriptive statistics and regression analysis to summarize and interpret data.
CO 4	Applying inferential statistics techniques to make data-driven decisions.

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

Course Outline

Unit Number: 1	Title: Foundations of Probability	No. of hours: 8
Content:		
• Basic notions of probability, events, and set operations • Conditional probability and independence of events • Applications of Bayes' theorem • Random variables: Discrete and continuous types • Cumulative distribution functions (CDF) and probability mass/density functions (PMF/PDF) • Mathematical expectation and moments		
Unit Number: 2	**Title: Engineering Applications of Probability Distributions**	**No. of hours: 8**
Content:		
• Joint and marginal distributions • Discrete distributions: Bernoulli, Binomial, Geometric, and Poisson distributions • Continuous distributions: Uniform, Exponential, and Normal distributions • Central Limit Theorem • Law of Large Numbers		
Unit Number: 3	**Title: Descriptive Statistics and Regression Analysis**	**No. of hours: 8**
Content:		
• Descriptive statistics: Measures of central tendency (mean, median, mode) and variability (variance, standard deviation) • Visualization techniques: Histograms, scatter plots • Correlation coefficient and covariance • Linear regression: Modeling relationships between variables • Least squares method for fitting regression models		
Unit Number: 4	**Title: Inferential Statistics for Engineers**	**No. of hours: 8**
Content:		

- Introduction to statistical inference
- Sampling distributions of mean and variance
- Estimation techniques: Point estimation and confidence intervals
- Hypothesis testing: Parametric tests (Z-test, T-test) and non-parametric tests

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** Utilize visual aids and real-life examples to explain fundamental concepts such as probability, distributions, and statistical inference, ensuring that students can relate these concepts to practical applications.
- **Hands-On Problem Solving:** Engage students in solving a variety of probability and statistics problems through guided practice sessions. Encourage the use of statistical software for data analysis and visualization, providing practical experience with tools like R or Excel.
- **Group Projects:** Foster collaborative learning by assigning group projects that involve collecting, analyzing, and interpreting data. Projects may include surveys, experiments, or analysis of existing datasets to apply statistical techniques learned in class.
- **Immediate Feedback:** Provide in-class exercises and quizzes to offer real-time feedback on students' understanding of probability and statistics concepts.

Outside Classroom Learning Experience

- **Assignments & Case Studies:** Provide assignments and case studies that require students to apply probability and statistical methods to real-world scenarios. These tasks will emphasize critical thinking and the application of statistical tools for decision-making in fields like economics, engineering, and social sciences.
- **Collaborative Projects:** Promote teamwork on larger statistical analysis projects outside the classroom, where students collect and interpret data, apply statistical methods, and present their findings.
- **Online Discussions:** Facilitate online forums where students can discuss statistical problems, share insights, and collaborate on project ideas, promoting peer learning.
- **Self-Study:** Encourage students to explore additional topics in probability and statistics through supplementary reading materials and practice exercises that deepen their understanding.
- **Practice Problems:** Provide additional problem sets for independent study to reinforce the key concepts of probability, statistical distributions, and data analysis techniques.

- **Peer Review:** Organize peer review sessions where students critique each other's data analyses, enhancing their ability to interpret and present statistical data effectively.

Text Books

- "Probability and Statistics for Engineers" by Richard A. Johnson
- "Probability & Statistics with R for Engineers and Scientists" by Michael Baron

Reference Books

- "Introduction to Probability and Statistics for Engineers and Scientists" by Sheldon M. Ross
- "Probability and Statistics for Engineering and the Sciences" by Jay L. Devore

Additional Readings

Self-Learning Components:

1. Link to Probability and Statistics course on NPTEL: https://nptel.ac.in/courses/111/106/111106112/
2. Link to Probability and Statistics on Coursera: https://www.coursera.org/courses?query=probability%20and%20statistics
3. Link to Probability and Statistics resources: https://www.khanacademy.org/math/statistics-probability
4. Link to Probability and Statistics tutorials: https://www.tutorialspoint.com/probability_and_statistics/index.htm
5. Link to Probability and Statistics lectures: https://ocw.mit.edu/courses/mathematics/18-05-introduction-to-probability-and-statistics-spring-2014/

Introduction to Java Programming

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Java Programming	ENBC205	3-1-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the fundamental concepts of Java programming, focusing on object-oriented programming (OOP), inheritance, polymorphism, 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 the basics of Java programming and OOP concepts.
CO 2	Implementing inheritance and polymorphism in Java programs.
CO 3	Handling exceptions and manage multithreading in Java.
CO 4	Performing file handling and utilize collections in Java.

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 and OOP	No. of hours: 10
Content:		

- Introduction to Java: Features, Importance, Java Virtual Machine, Byte Code
- Keywords, constants, variables, and Data Types, Operators and Expressions, Type casting and conversion
- Java Control Structure: Decision making (if, if-else, switch-case), Loop (do, while, for), jump statements (break and continue)
- Simple Input and Output: Scanner Class
- Arrays Handling: Single and Multi-dimensional, Referencing Arrays Dynamically
- Java Strings: String class, Creating & Using String Objects, Manipulating Strings, String Immutability & Equality
- OOP Paradigm: Features of OOP, Class and Object in Java, Overloading Member Methods, Static Members, this Keyword
- Constructors: default, parameterized, and copy constructors

Unit Number: 2	Title: Inheritance and Polymorphism	No. of hours: 10
Content:		

- Access Specifiers, Introduction to Inheritance: Derived Class and Super class, super Keyword
- Types of inheritance: simple, multilevel, hierarchical
- Polymorphism: Static (Method overloading), Dynamic (Method Overriding)
- Abstract Method and Abstract Class
- Interfaces: Defining and Implementing an Interface
- Packages: Creating, Naming, Using Package Members

Unit Number: 3	Title: Exception Handling and Multithreading	No. of hours: 10
Content:		

- Exception Handling: Dealing with Errors, Classification of Exceptions, Declaring and Throwing Exceptions, Catching Exceptions, finally clause
- Multithreaded Programming: Fundamentals, Java thread model, priorities, synchronization, thread classes, Runnable interface, inter-thread Communication
- Wrapper Classes: Autoboxing/Unboxing, Enumerations

Unit Number: 4	**Title: I/O Stream and Collections**	**No. of hours: 10**

Content:

- File Handling: File Class Methods, Reading from a File, Writing to a File, Buffered I/O, Character Streams, Byte Streams, File Input/Output Stream
- Java Collections Framework: Introduction, Collection Interfaces: List (ArrayList, LinkedList), Set (HashSet, LinkedHashSet), Map (HashMap)
- Working with Collections: Adding, Removing, Searching Elements, Iterating Elements

Learning Experience

Classroom Learning Experience

- **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.
- **Immediate Feedback:** Offer real-time feedback during quizzes, coding challenges, and in-class problem-solving activities to ensure students understand core Java concepts and their applications.

Outside Classroom Learning Experience

- **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, reinforcing practical knowledge.

- **Collaborative Projects:** Promote teamwork on larger projects outside the classroom, where students design and implement Java applications, applying concepts learned in class to create functional software solutions.
- **Online Discussions:** Facilitate online discussion boards where students can collaborate, ask questions, and share coding strategies for solving complex Java programming problems.
- **Peer Review:** Organize peer review sessions where students evaluate each other's Java code, providing constructive feedback on coding style, efficiency, and use of OOP principles.
- **Self-Study:** Encourage students to explore advanced Java topics such as JavaFX, streams, and design patterns through additional resources and practice exercises.
- **Practice Problems:** Provide supplementary coding challenges and problem sets for students to complete independently, reinforcing key Java programming concepts like exception handling, collections, and multithreading.

Text Books

- "Java: The Complete Reference" by Herbert Schildt
- "Head First Java" by Kathy Sierra and Bert Bates

Reference Books

- "Effective Java" by Joshua Bloch
- "Core Java Volume I–Fundamentals" by Cay S. Horstmann

Additional Readings

Self-Learning Components:

1. Link to Java Programming course on NPTEL: https://nptel.ac.in/courses/106/106/106106147/
2. Link to Java Programming on Coursera: https://www.coursera.org/courses?query=java%20programming
3. Link to Java Programming resources: https://www.geeksforgeeks.org/java/
4. Link to Java Programming tutorials: https://www.tutorialspoint.com/java/index.htm
5. Link to Java Programming lectures: https://ocw.mit.edu/courses/electrical-engineering-6-092-java-preparation-for-6-170-january-iap-2010/

Verbal Ability

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Verbal Ability	AEC006	3-0-0	3
Type of Course:	AEC		
Pre-requisite(s):	None		

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

The Course Outcomes (COs)

On completion of the course, the participants will be 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

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

Course Outline

Unit Number: 1	Title: Vocabulary Development and Application	No. of hours: 10
Content:		
Content Summary: Understanding the concept of root words, Prefix and suffix, Ways to enhance Vocabulary, Crosswords and word quizzes, Confusing words, One word substitution, Odd one out, Synonyms and Antonyms, Commonly misspelt words, Idioms and Phrases		
Unit Number: 2	**Title: Fundamentals of Grammar and Sentence Structure**	**No. of hours: 8**
Content:		
Content Summary: Introduction to Parts of Speech, Tenses and its 'rules, Sentences (Simple, Compound and Complex), Subject Verb Agreement, Pronoun Antecedent agreement, Phrases and Clauses		
Unit Number: 3	**Title: Mastering Sentence Accuracy and Completion Skills**	**No. of hours: 12**
Content:		
Content Summary: Spot the error (grammatical errors in a sentence), Sentence Correction (Improvement of sentences based on Grammar rules), Sentence Completion, Cloze Tests		
Unit Number: 4	**Title: Enhancing Sentence Structure and Reading Comprehension Skills**	**No. of hours: 6**
Content:		
Logical Arrangement of Sentences, Comprehending passages, Contextual questions, Anagrams, Analogies		

Learning Experience

Classroom Learning Experience

- **Interactive Sessions:** Facilitate interactive discussions and role-playing exercises focusing on vocabulary usage, sentence construction, and grammar applications. This hands-on approach helps students internalize grammar rules and enhance their word choice in real communication scenarios.
- **Workshops:** Organize workshops that include activities like crossword puzzles, quizzes on synonyms and antonyms, and idiomatic expressions to enrich vocabulary in a fun and engaging way.
- **Peer Review Sessions:** Conduct peer review sessions where students present short essays or paragraphs, receiving constructive feedback on grammar, vocabulary, and sentence structure from classmates and instructors.
- **Group Discussions:** Hold group discussions on various topics that require students to utilize their newly learned vocabulary and grammar in context, promoting active learning and practical application of language skills.

- **Immediate Feedback:** Provide immediate, actionable feedback during classroom activities to help students correct errors and refine their verbal abilities on the spot.

Outside Classroom Learning Experience

- **Assignments & Case Studies:** Assign homework and case studies that involve writing essays, reports, or analyses using appropriate grammar and rich vocabulary. These tasks should challenge students to apply their language skills in varied contexts and purposes.
- **Collaborative Projects:** Encourage students to engage in collaborative writing projects, where they can practice and improve their language skills while working in teams, thereby enhancing both their verbal and collaborative skills.
- **Online Forums:** Set up online forums where students can participate in writing challenges, share their work, and provide feedback to peers, thus creating a community of learning and mutual improvement.
- **Reading Groups:** Form reading groups that meet regularly to discuss assigned readings, focusing on language usage, style, and comprehension. This activity helps reinforce reading as a method of improving writing and speaking skills.
- **Self-Directed Learning:** Recommend a list of books, articles, and online resources for self-directed learning. Encourage students to explore these resources to enhance their understanding and mastery of complex language structures and vocabulary.

Reference Books

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

Additional Readings

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

Introduction to Java Programming Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Java Programming Lab	ENBC251	0-0-2	1
Type of Course:	Major		

Defined Course Outcomes

COs	**Statements**
CO 1	Understanding the basic syntax, features, and structure of Java programming language.
CO 2	Developing Java applications using OOP concepts such as classes, objects, inheritance, and polymorphism.
CO 3	Handling exceptions and implement multithreading in Java programs.
CO 4	Utilizing Java I/O streams and collections framework to handle data and perform file operations.

S.N	**Lab Task**	**Mapped CO/COs**
1	Write a Java program to understand the basic syntax and structure of a Java program.	CO1
2	Develop a program using data types, variables, and control structures.	CO1
3	Implement arrays handling (single and multi-dimensional) in Java.	CO1
4	Write a program to manipulate strings using Java String class.	CO1
5	Develop a program to demonstrate OOP features: classes, objects, and constructors.	CO2
6	Implement a program using static members and the ‘this‘ keyword.	CO2
7	Write a program to demonstrate method overloading and overriding.	CO2
8	Develop a program to implement inheritance (simple, multilevel, hierarchical).	CO2
9	Implement a program to demonstrate polymorphism in Java.	CO2
10	Create a program using abstract classes and interfaces.	CO2
11	Develop a program to handle exceptions using try, catch, throw, and finally.	CO3

S.N	Lab Task	Mapped CO/COs
12	Implement a multithreaded program demonstrating thread creation and synchronization.	CO3
13	Write a program to demonstrate autoboxing and unboxing using wrapper classes.	CO3
14	Develop a program for file handling operations: reading and writing to a file.	CO4
15	Implement a program to demonstrate the use of byte streams and character streams.	CO4
16	Create a program using Java collections framework: List, Set, and Map interfaces.	CO4
17	Develop a program to add, remove, search, and iterate elements in collections.	CO4
18	Implement a program to use function templates.	CO4
19	Write a program to use class templates.	CO4
20	Create a program to handle errors during file operations.	CO4
1	**Student Management System:** Develop a student management system using OOP concepts like classes, objects, inheritance, and polymorphism.	CO2, CO3
2	**Library Management System:** Create a library management system that uses file handling for data storage and retrieval.	CO4
3	**Banking System:** Design a banking system implementing dynamic memory allocation and exception handling.	CO2, CO4
4	**Inventory Management System:** Develop an inventory management system using templates and collections.	CO3, CO4
5	**Online Shopping System:** Create an online shopping system that demonstrates all learned OOP concepts including inheritance, polymorphism, file handling, and exception handling.	CO2, CO3, CO4

Online Learning Resources

- **GeeksforGeeks:** Tutorials and articles on Java programming and object-oriented programming concepts.
  https://www.geeksforgeeks.org/java/
- **TutorialsPoint:** Comprehensive guides on Java programming and OOP principles.
  https://www.tutorialspoint.com/java/index.htm
- **NPTEL:** Video lectures and course materials on Java programming.
  https://nptel.ac.in/courses/106/106/106106147/
- **Coursera:** Courses on Java programming and object-oriented principles from leading universities.

https://www.coursera.org/courses?query=java

Introduction to Data Structures Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Data Structures Lab	ENBC253	0-0-2	1
Type of Course:	Major		

Defined Course Outcomes

COs	**Statements**
CO 1	Understanding the fundamentals of data structures, including arrays and their applications.
CO 2	Implementing linear data structures such as linked lists, stacks, and queues.
CO 3	Applying sorting and searching algorithms to various data structures.
CO 4	Implementing tree and graph algorithms and understand their applications.

S.N	**Lab Task**	**Mapped CO/COs**
1	Write a program to understand the basic concepts of arrays and their applications.	CO1
2	Implement a program to measure the time and space complexity of an algorithm.	CO1
3	Develop a program to demonstrate single and multi-dimensional arrays.	CO1
4	Implement a program to demonstrate row major and column major order representation.	CO1
5	Write a program to implement single linked lists and perform various operations on them.	CO2
6	Develop a program to implement doubly linked lists and perform various operations on them.	CO2
7	Implement a program to demonstrate circular linked lists and perform operations.	CO2
8	Create a program to perform stack operations using arrays and linked lists.	CO2
9	Write a program to evaluate postfix expressions using stack.	CO2
10	Develop a program to perform queue operations using arrays and linked lists.	CO2
11	Implement a program to demonstrate circular queues.	CO2

S.N	Lab Task	Mapped CO/COs
12	Write a program to perform sequential and binary search on an array.	CO3
13	Develop a program to implement insertion sort on an array.	CO3
14	Implement a program to perform selection sort on an array.	CO3
15	Write a program to implement bubble sort on an array.	CO3
16	Develop a program to perform quick sort on an array.	CO3
17	Implement a program to perform merge sort on an array.	CO3
18	Write a program to demonstrate hash table and collision resolution strategies.	CO3
19	Develop a program to implement binary tree and perform various tree traversal algorithms.	CO4
20	Implement a program to demonstrate graph representation and perform graph traversal algorithms.	CO4
1	**Library Management System:** Develop a library management system using data structures like linked lists, stacks, and queues for various operations.	CO2, CO3
2	**Hospital Management System:** Create a hospital management system to manage patient records using arrays and linked lists.	CO2, CO3
3	**Social Network Analysis:** Implement a social network analysis tool using graph algorithms to find shortest paths and minimum spanning trees.	CO4
4	**Inventory Management System:** Develop an inventory management system using sorting and searching algorithms for efficient data retrieval.	CO3
5	**Pathfinding Algorithm:** Create a pathfinding algorithm for a maze using tree and graph traversal algorithms.	CO4

Online Learning Resources

- **GeeksforGeeks:** Tutorials and articles on data structures and algorithms. https://www.geeksforgeeks.org/data-structures/
- **TutorialsPoint:** Comprehensive guides on data structures and their implementations. https://www.tutorialspoint.com/data_structures_algorithms/index.htm
- **NPTEL:** Video lectures and course materials on data structures and algorithms. https://nptel.ac.in/courses/106/106/106106127/
- **Coursera:** Courses on data structures and algorithms from leading universities. https://www.coursera.org/courses?query=data%20structures

Fundamentals of Cyptography Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Fundamentals of Cyptography Lab	ENSP259	0-0-2	1
Type of Course:	Minor		

Defined Course Outcomes

COs	**Statements**
CO 1	Implementing and analyze classical cryptographic techniques such as substitution and transposition ciphers.
CO 2	Developing programs to perform symmetric key encryption and decryption using algorithms like AES and DES.
CO 3	Applying public key cryptography techniques, including RSA and Diffie-Hellman key exchange.
CO 4	Implementing cryptographic protocols and analyze their security properties.

Proposed Lab Experiments

Ex. No	Experiment Title	Mapped CO/COs
1	Implement Caesar Cipher and analyze its security against brute force attacks.	CO 1
2	Implement the Vigenère Cipher and demonstrate how frequency analysis can be used to break it.	CO 1
3	Develop a program to perform encryption and decryption using the Rail Fence Cipher and Columnar Transposition Cipher.	CO 1
4	Implement symmetric key encryption using the Data Encryption Standard (DES) and analyze its operation.	CO 2
5	Develop a program to encrypt and decrypt messages using the Advanced Encryption Standard (AES) and compare it with DES.	CO 2
6	Implement RSA encryption and decryption, and demonstrate key generation and management in RSA.	CO 3
7	Perform Diffie-Hellman key exchange and demonstrate how two parties can securely exchange keys over an insecure channel.	CO 3
8	Implement and analyze digital signatures using RSA and demonstrate their use in verifying message integrity.	CO 4
9	Develop a program to implement HMAC for message authentication and analyze its security properties.	CO 4
10	Implement SSL/TLS handshake simulation to understand the cryptographic protocols involved in secure communication.	CO 4

Online Learning Resources

- **Coursera:** Specializations and courses in cryptography by top universities.
  https://www.coursera.org/specializations/cryptography
- **Khan Academy:** Introductory cryptography courses and resources.
  https://www.khanacademy.org/computing/computer-science/cryptography
- **EDX:** Courses on cryptography and security offered by institutions like MIT and Stanford.
  https://www.edx.org/course/subject/computer-science/cryptography
- **NIST:** National Institute of Standards and Technology resources on cryptographic standards.
  https://csrc.nist.gov/

Summer Internship-I

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Summer Internship-I	SIBC251	0-0-0-	2
Type of Course:	INT		

Duration

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

Internship Options

Students can choose from the following options:

1. **Industry Internship (Offline):**
   - 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. **Research Internship:**
   - Students can engage in a research internship under the mentorship of a faculty member for six weeks.
4. **On-Campus Industry Internship Programs:**
   - The university will offer on-campus internships in collaboration with industry partners.
5. **Internships at Renowned Institutions:**
   - Students can pursue summer internships at esteemed institutions such as IITs, NITs, Central Universities, etc.

Report Submission and Evaluation

1. **Report Preparation:**
   - Students must prepare a detailed report documenting their internship experience and submit it to the department. A copy of the report will be kept for departmental records.

2. **Case Study/Project/Research Paper:**
   - Each student must complete one of the following as part of their internship outcome:
     - A case study
     - A project
     - A research paper suitable for publication
3. **Presentation:**
   - Students are required to present their learning outcomes and results from their summer internship as part of the evaluation process.

Evaluation Criteria for Summer Internship (Out of 100 Marks)

1. **Relevance to Learning Outcomes (30 Marks)**
   - **Case Study/Project/Research Paper Relevance (15 Marks):**
     - Directly relates to core subjects: 15 marks
     - Partially relates to core subjects: 10 marks
     - Minimally relates to core subjects: 5 marks
     - Not relevant: 0 marks
   - **Application of Theoretical Knowledge (15 Marks):**
     - Extensive application of theoretical knowledge: 15 marks
     - Moderate application of theoretical knowledge: 10 marks
     - Minimal application of theoretical knowledge: 5 marks
     - No application of theoretical knowledge: 0 marks
2. **Skill Acquisition (30 Marks)**
   - **New Technical Skills Acquired (15 Marks):**
     - Highly relevant and advanced technical skills: 15 marks
     - Moderately relevant technical skills: 10 marks
     - Basic technical skills: 5 marks
     - No new skills acquired: 0 marks
   - **Professional and Soft Skills Development (15 Marks):**
     - Significant improvement in professional and soft skills: 15 marks
     - Moderate improvement in professional and soft skills: 10 marks
     - Basic improvement in professional and soft skills: 5 marks
     - No improvement: 0 marks
3. **Report Quality (20 Marks)**
   - **Structure and Organization (10 Marks):**
     - Well-structured and organized report: 10 marks
     - Moderately structured report: 7 marks

  - Poorly structured report: 3 marks
  - No structure: 0 marks
- **Clarity and Comprehensiveness (10 Marks):**
  - Clear and comprehensive report: 10 marks
  - Moderately clear and comprehensive report: 7 marks
  - Vague and incomplete report: 3 marks
  - Incomprehensible report: 0 marks

4. **Presentation (20 Marks)**
   - **Content Delivery (10 Marks):**
     - Clear, engaging, and thorough delivery: 10 marks
     - Clear but less engaging delivery: 7 marks
     - Somewhat clear and engaging delivery: 3 marks
     - Unclear and disengaging delivery: 0 marks
   - **Visual Aids and Communication Skills (10 Marks):**
     - Effective use of visual aids and excellent communication skills: 10 marks
     - Moderate use of visual aids and good communication skills: 7 marks
     - Basic use of visual aids and fair communication skills: 3 marks
     - No use of visual aids and poor communication skills: 0 marks

Total: 100 Marks

Course Outcomes

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

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

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

Competitive Coding Bootcamp- I

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Competitive Coding Bootcamp-I		2-0-0-	0
Type of Course:	AUDIT		

Course Outcomes (COs):

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

Course Outline

Unit No.	**Title**	**No. of hours**
1	**Foundations of Competitive Programming** **Content:** • Introduction to Competitive Programming Platforms: Overview of major platforms like Codeforces, LeetCode, HackerRank, etc. • Setting up accounts and environment for competitive programming. • Solving introductory problems to get familiar with the platforms. • Problem-Solving Strategies: Techniques for solving problems, Greedy Algorithms, Divide and Conquer, Brute Force methods.	8

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

Experiment List

Problem Statement	Mapped COs
Two Sum: Find two numbers that add up to a specific target.	CO1
Best Time to Buy and Sell Stock: Maximize profit from stock prices.	CO1
Valid Parentheses: Check if a string contains valid parentheses.	CO1
Greedy Algorithm: Jump Game - Can you reach the end of the array?	CO1
Divide and Conquer: Merge Sort implementation to sort an array.	CO1
Brute Force: Find all subsets of a given set.	CO1
Greedy Algorithm: Minimum Number of Platforms Required for Trains	CO1
Divide and Conquer: Maximum Subarray (Kadane's Algorithm)	CO1
Brute Force: Count number of occurrences of a substring in a string.	CO1
Greedy Algorithm: Coin Change Problem (Minimum Coins)	CO1
Time Complexity: Check if a number is prime using $O(\sqrt{n})$ complexity.	CO2
Sorting: QuickSort algorithm with O(n log n) complexity.	CO2
Big O Notation: Analyze time complexity of an algorithm.	CO2
Space Complexity: Fibonacci with O(n) space complexity.	CO2
Time Complexity: Find first duplicate element in an array with O(n) time.	CO2
Time Complexity: Search an element in a rotated sorted array in O(log n) time.	CO2
Complexity Analysis: Binary Search Tree operations with complexity O(log n).	CO2
Analyze best, average, and worst case for Insertion Sort.	CO2
Time and Space Complexity: Check the complexity of an algorithm (recurrences).	CO2
Time Complexity: Compute factorial recursively with complexity analysis.	CO2
Recursion: Generate all permutations of a string.	CO3
Dynamic Memory Allocation: Implement a dynamic array (vector) from scratch.	CO3
Backtracking: Solve the N-Queens problem using recursion.	CO3
Pointers: Swap two numbers using pointers in C++.	CO3
File Handling: Read and write data to a file in Python/C++/Java.	CO3
Function Overloading: Implement overloaded functions for adding integers and floats.	CO3
Dynamic Memory Allocation: Use malloc and free to manage memory in C.	CO3
Recursion: Solve Tower of Hanoi using recursion.	CO3
Arrays: Rotate an array to the right by k steps.	CO4
Strings: Find the longest substring without repeating characters.	CO4

Learning Experiences

- Understanding Memory Management: Students grasp the concept of memory allocation and deallocation through pointers, gaining insights into how data is stored and accessed in memory.

- Pointer Arithmetic: Learners practice pointer arithmetic to navigate arrays and structures, enhancing their ability to perform low-level data manipulations efficiently.
- Dynamic Memory Allocation: Students experience dynamic memory allocation with functions like malloc, calloc, and free, learning to manage memory dynamically during runtime.
- Pointer and Function Interactions: Students explore how pointers are used to pass arguments by reference, leading to more efficient function calls and manipulation of data within functions.
- Pointer to Pointer Concepts: Learners work with pointer to pointer (double pointers) to understand multi-level indirection and its applications in complex data structures and dynamic memory management.
- Debugging with Pointers: Students enhance their debugging skills by identifying and fixing pointer-related issues such as memory leaks, dangling pointers, and segmentation faults.

Textbooks

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

Online Resources

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

List of Suggested Competitive Programming Courses

- Algorithms and Data Structures by MIT OpenCourseWare
- Introduction to Competitive Programming by NPTEL
- Competitive Programming by HackerRank
- The Bible of Competitive Programming & Coding Interviews

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

Web References

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

References to Interview Questions

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

VAC-III

Design Thinking & Innovations for Engineers

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Design Thinking & Innovations for Engineers	VAC170	0-0-0-	2
Type of Course:	VAC		

Course Perspective: This course aims to cultivate an innovative mindset and enhance creative problem-solving skills through practical experience in design thinking processes and innovation methodologies. It guides students from the inception of an idea to the execution of a startup.

The Course Outcomes (COs)

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

COs	Statements
CO 1	Understanding and applying the principles of design thinking to solve engineering problems.
CO 2	Developing innovative ideas through ideation and prototyping techniques.
CO 3	Implementing innovation strategies in engineering projects.
CO 4	Integrating design thinking methodologies in real-world engineering applications, ensuring sustainable and user-centric solutions.

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

Course Outline

Unit Number: 1	**Title: Introduction to Design Thinking**	**No. of hours: NA**
Content:		

- Overview of Design Thinking: History, principles, and importance.
- Key Stages of Design Thinking: Empathize, Define, Ideate, Prototype, and Test.
- Innovation Types: Incremental vs. Disruptive Innovation.
- Tools and Techniques for Design Thinking: Brainstorming, Mind Mapping, Sketching.
- Practical Exercise: Identify and Define a Problem Statement.

Unit Number: 2	**Title: Ideation and Prototyping**	**No. of hours: NA**

Content:

- Idea Generation Techniques: Brainstorming, SCAMPER, Reverse Engineering.
- Prototyping: Importance, Methods, and Tools.
- User-Centered Design: Conducting User Research and Testing.
- Practical Exercise: Develop and Prototype a Solution for the Identified Problem.
- Evaluation and Feedback: Iterative Process for Refinement.

Unit Number: 3	**Title: Innovation Strategies and Entrepreneurship Life Cycle**	**No. of hours: NA**

Content:

- Types of innovation: Incremental, Disruptive, Radical.
- Innovation frameworks and models.
- Introduction to Entrepreneurship: Definition, Characteristics, and Importance.
- Stages of Entrepreneurship Life Cycle: Ideation, Validation, Scaling, and Exit.
- Business Model Canvas: Value Proposition, Customer Segments, Channels, Revenue Streams.
- Funding and Investment: Sources of Funding, Pitching to Investors.
- Practical Exercise: Create a Business Model Canvas for the Prototype.

Unit Number: 4	**Title: Application of Design Thinking in Engineering**	**No. of hours: NA**

Content:

- Applying design thinking in engineering projects.
- Case studies: IDEO's Shopping Cart, Airbnb, Tesla, Google's Design Sprint, and a local startup success story.

Learning Experience

Classroom Learning Experience

- **Collaborative Workshops:** Facilitate hands-on workshops where students collaborate in teams to apply design thinking methodologies, encouraging diverse perspectives and interdisciplinary approaches to problem-solving.
- **Ideation Sessions:** Conduct brainstorming and ideation sessions using techniques like SCAMPER and Reverse Engineering, enabling students to generate innovative ideas and explore creative solutions to real-world problems.
- **Prototyping Activities:** Engage students in rapid prototyping exercises, allowing them to transform their ideas into tangible models. These activities will emphasize the importance of user feedback and iterative improvement.
- **Case Studies and Industry Examples:** Analyze case studies of successful innovations and startups, such as IDEO's Shopping Cart, Airbnb, and Tesla, to illustrate the practical application of design thinking in engineering and entrepreneurship.
- **Feedback and Iteration:** Provide continuous feedback during workshops and prototyping sessions, encouraging students to iterate and refine their ideas based on peer and instructor input.

Outside Classroom Learning Experience

- **User-Centered Design Projects:** Assign projects where students must conduct user research, develop prototypes, and test their designs with real users, ensuring that the solutions are practical, sustainable, and user-centric.
- **Business Model Development:** Guide students through the process of creating a Business Model Canvas for their prototypes, focusing on value proposition, customer segments, and revenue streams, preparing them for the entrepreneurial aspects of engineering innovation.
- **Pitching and Presentation Skills:** Train students in pitching their innovative ideas to potential investors or stakeholders, enhancing their communication and presentation skills.
- **Self-Study and Case Analysis:** Encourage students to explore additional case studies of innovative products and services, reinforcing the application of design thinking principles in different industries.
- **Practice Problems:** Provide practice scenarios where students can apply design thinking frameworks to develop solutions to hypothetical engineering challenges.

Text & Reference Books

- "Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation" by Tim Brown
- "Creative Confidence: Unleashing the Creative Potential Within Us All" by Tom Kelley and David Kelley

- "Design Thinking for Strategic Innovation: What They Can't Teach You at Business or Design School" by Idris Mootee
- "The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail" by Clayton M. Christensen
- "Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days" by Jake Knapp, John Zeratsky, and Braden Kowitz
- "The Design of Everyday Things" by Don Norman
- "The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses" by Eric Ries
- "Making Ideas Happen: Overcoming the Obstacles Between Vision and Reality" by Scott Belsky
- "Innovation and Entrepreneurship" by Peter F. Drucker

AWS Cloud Fundamentals

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
AWS Cloud Fundamentals	VAC171	0-0-0-	2
Type of Course:	VAC		

Course Perspective: This course provides a foundational understanding of Amazon Web Services (AWS), focusing on its core services, architecture, security measures, and best practices. It prepares students to effectively deploy and manage applications on the AWS platform.

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the fundamentals of cloud computing and the benefits of using AWS.
CO 2	Identifying and utilizing core AWS services for computing, storage, and networking.
CO 3	Implementing security measures and best practices on AWS.
CO 4	Deploying, monitoring, and managing applications on the AWS platform.

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

Course Outline

Unit Number: 1	**Title: Introduction to Cloud Computing and AWS**	**No. of hours: 7**
Content:		

- Overview of cloud computing models (IaaS, PaaS, SaaS)
- Introduction to AWS and its global infrastructure
- AWS Management Console and key concepts
- Amazon EC2 instances
- Billing, pricing models, and account management
- Introduction to AWS Free Tier and hands-on lab setup

Unit Number: 2	**Title: Core AWS Services**	**No. of hours: 7**

Content:

- Amazon EC2: Virtual servers in the cloud
- Amazon S3: Scalable storage in the cloud, Creation of S3 Bucket, S3 versioning, S3 cross region replication
- Amazon RDS: Managed relational database service, AWS DynamoDB
- Amazon VPC: Virtual Private Cloud: Creation VPC, Subnet, Net gateway, and Route Table
- AWS Route 53, Creation of Route 53 in AWS.
- AWS Simple Notification Service (SNS), How to send email and SMS from SNS
- AMI: Creation of AMI, Copy AMI into another AWS Account, Attaching root volume with another EC2 instance

Unit Number: 3	**Title: AWS Security and Compliance**	**No. of hours: 7**

Content:

- AWS Lambda: Serverless computing, Trigger, Downstream Resources and Runtime
- Amazon CloudFront: Content delivery network simulation
- Identity and Access Management (IAM), Cross account access using IAM Role, how to connect Windows AD server to AWS.
- Creation of Security groups, network ACLs, and VPC security
- Elastic Block Storage
- AWS Auto scaling
- AWS compliance programs and certifications
- Monitoring and logging with AWS CloudTrail and CloudWatch
- Data encryption and security best practices

Unit Number: 4	**Title: Deploying and Managing Applications on AWS**	**No. of hours: 7**

Content:

- Deploying web applications and portfolio of students using AWS EC2 instance on cloud
- Deploying Web services on different servers i.e. Microsoft servers, Linux, and AWS servers
- Hosting Static Website on S3 Bucket
- Load Balancer: Creation of Load Balancer on VPC or among VPC
- Monitoring and troubleshooting applications
- Backup and disaster recovery strategies
- Case studies and real-world applications

Learning Experience

Classroom Learning Experience

- **Hands-On Labs:** Engage students in practical, hands-on lab exercises using the AWS Free Tier. These labs will cover the setup and management of core AWS services like EC2, S3, and VPC, providing real-world experience in cloud computing.
- **Interactive Demonstrations:** Utilize live demonstrations to show the deployment of applications on AWS, including the creation and configuration of instances, setting up virtual networks, and managing databases.
- **Case Studies:** Analyze real-world case studies where AWS is used to solve com-

plex business problems, helping students understand the practical applications and benefits of cloud computing.

- **Collaborative Projects:** Encourage students to work in teams to design, deploy, and manage a cloud-based solution. This project will cover the complete lifecycle, from architecture design to deployment and monitoring.
- **Guest Lectures:** Invite industry experts to deliver guest lectures on advanced AWS topics and emerging trends in cloud computing, providing students with insights into current industry practices.
- **Quizzes and Assessments:** Implement periodic quizzes and assessments to reinforce understanding of key concepts like cloud architecture, security, and best practices.

Outside Classroom Learning Experience

- **Discussion Forums:** Facilitate online discussion forums where students can share experiences, discuss challenges, and exchange solutions related to AWS services.
- **Certification Preparation:** Provide resources and guidance for students interested in pursuing AWS certifications, including practice exams, study materials, and certification roadmaps.
- **Self-Study:** Encourage students to explore additional AWS services such as Lambda, CloudFront, and RDS by accessing AWS documentation, tutorials, and online courses.
- **Practice Projects:** Assign individual practice projects where students can experiment with advanced AWS services and architectures, reinforcing concepts learned in class.
- **Collaborative Work:** Encourage students to work in teams on larger cloud-based projects, applying their knowledge to solve complex problems and optimize cloud infrastructure.

Reference Books

- "AWS Certified Solutions Architect Official Study Guide" by Joe Baron, Hisham Baz, Tim Bixler, Biff Gaut, Kevin E. Kelly, Sean Senior, John Stamper
- "AWS Certified Developer Official Study Guide" by Nick Alteen, Jennifer Fisher, Jason Leznek, John Stamper
- "Amazon Web Services in Action" by Andreas Wittig and Michael Wittig
- "AWS for Solutions Architects" by Alberto Artasanchez
- "Cloud Computing: Concepts, Technology & Architecture" by Thomas Erl, Zaigham Mahmood, Ricardo Puttini

Additional Readings

1. "AWS Lambda in Action" by Danilo Poccia
2. "Serverless Architectures on AWS" by Peter Sbarski
3. "Cloud Native Transformation" by Pini Reznik, Jamie Dobson, Michelle Gienow
4. "The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win" by Gene Kim, Kevin Behr, George Spafford

Web Development with Open Source Frameworks

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Web Development with Open Source Frameworks	VAC172	0-0-2	2
Type of Course:	VAC		
Pre-requisite(s):	None		

Course Perspective: This course provides hands-on experience in web development using various open-source frameworks, focusing on practical application to create functional web applications.

The Course Outcomes (COs)

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

COs	Statements
CO 1	Understanding and setting up web development environments using open source tools.
CO 2	Developing and style web pages using HTML, CSS, and JavaScript frameworks.
CO 3	Building dynamic web applications using open-source backend frameworks.
CO 4	Deploying and managing web applications using open-source deployment tools.

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

Course Outline

Unit Number: 1	Title: Introduction to Web Development and Setup	No. of hours: 7
Content:		
• Overview of Web Development: Client-Server Architecture • Introduction to Open-Source Tools: Git, VSCode, Browser Developer Tools • Setting Up Development Environment: Installing and Configuring Tools • Version Control with Git and GitHub • Hands-on Project: Set up a basic web development environment and create a simple HTML page.		
Unit Number: 2	**Title: Frontend Development with Open-Source Frameworks**	**No. of hours: 7**
Content:		
• HTML5 and CSS3: Structure and Styling • Responsive Design with Bootstrap • Introduction to JavaScript: Basics and DOM Manipulation • JavaScript Frameworks: Overview of React.js and Vue.js • Hands-on Project: Create a responsive webpage with dynamic content using Bootstrap and JavaScript.		
Unit Number: 3	**Title: Backend Development with Open Source Frameworks**	**No. of hours: 5**
Content:		
• Introduction to Backend Development: Server-Side Scripting • Overview of Backend Frameworks: Node.js with Express.js and Django • RESTful APIs: Creation and Consumption • Database Integration: MongoDB (with Node.js) and SQLite (with Django) • Hands-on Project: Build a basic web application with user authentication and data storage using Node.js and Express.js or Django.		
Unit Number: 4	**Title: Deployment and Real-World Projects**	**No. of hours: 5**
Content:		

- Overview of Deployment Tools: Docker, Heroku, Netlify
- Continuous Integration and Deployment (CI/CD) with GitHub Actions
- Real-World Project 1: Develop and Deploy a Blog Website
- Real-World Project 2: Develop and Deploy a To-Do List Application
- Final Project: Students develop and deploy their own web application.

Learning Experience

Classroom Learning Experience

- **Project-Based Learning:** Engage students in real-world projects that require them to build functional web applications using open-source frameworks. This hands-on approach ensures they apply theoretical knowledge to practical tasks.
- **Interactive Labs:** Conduct interactive lab sessions where students can experiment with different web development tools and frameworks like React.js, Vue.js, Node.js, Express.js, and Django. These labs will provide step-by-step guidance to help students understand the nuances of each tool.
- **Version Control Practice:** Implement version control with Git and GitHub throughout the course, encouraging students to track changes, collaborate on projects, and manage code repositories effectively.
- **Responsive Design Techniques:** Focus on creating responsive web designs that adapt to different screen sizes using frameworks like Bootstrap. Students will learn to ensure their applications are user-friendly across devices.
- **Collaborative Development:** Promote teamwork by assigning group projects where students can collaborate on the frontend and backend of web applications, simulating a real-world development environment.
- **Guest Lectures and Webinars:** Invite industry experts to discuss emerging trends in web development and the latest open-source tools, providing students with insights into the current state of the industry.

Outside Classroom Learning Experience

- **Continuous Integration and Deployment (CI/CD):** Introduce students to CI/CD practices using tools like GitHub Actions, Docker, and Heroku. They will learn to automate testing, deployment, and scaling of their applications.
- **Final Capstone Project:** At the end of the course, students will develop and deploy their own web application using the frameworks and tools they have learned. This capstone project will demonstrate their understanding and ability to create a complete, functional web application from scratch.

- **Self-Study:** Encourage students to explore additional features of open-source frameworks and tools, such as serverless architecture and advanced frontend optimization techniques, to enhance their web development skills.
- **Peer Code Review:** Organize peer code review sessions where students critique and provide feedback on each other's projects, improving code quality and collaboration.
- **Online Discussion Forums:** Facilitate online forums where students can ask questions, share their experiences with frameworks, and collaborate on solving challenges.

Text Books and Online Resources

- "Eloquent JavaScript" by Marijn Haverbeke
- "Learning Web Design" by Jennifer Robbins
- MDN Web Docs (https://developer.mozilla.org/)
- freeCodeCamp (https://www.freecodecamp.org/)
- W3Schools (https://www.w3schools.com/)

Tools Used

- VSCode (https://code.visualstudio.com/)
- Git and GitHub (https://github.com/)
- Bootstrap (https://getbootstrap.com/)
- React.js (https://reactjs.org/)
- Node.js (https://nodejs.org/)
- Express.js (https://expressjs.com/)
- Django (https://www.djangoproject.com/)
- MongoDB (https://www.mongodb.com/)
- Heroku (https://www.heroku.com/)

Google Data Analytics

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Google Data Analytics	VAC173	0-0-2	2
Type of Course:	Value Added Course (VAC)		
Pre-requisite(s):	None		

Course Perspective: This course provides hands-on experience in data analytics using Google's tools. Students will learn to effectively utilize Google Analytics, Google Data Studio, Google Sheets, and BigQuery for comprehensive data analyses and visualizations.

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding and utilizing Google Analytics for tracking and reporting website traffic.
CO 2	Performing data manipulation and analysis using Google Sheets.
CO 3	Creating interactive data visualizations using Google Data Studio.
CO 4	Conducting advanced data analysis and querying using Google BigQuery.

Course Outline

Unit Number: 1	**Title: Google Analytics Overview**	**No. of hours: 7**
Content:		
• Setting up Google Analytics account and properties • Understanding Key Metrics and Dimensions • Analyzing Traffic Sources • Tracking Goals and Conversions • Hands-on Project: Implementing Google Analytics for a website		
Unit Number: 2	**Title: Data Manipulation with Google Sheets**	**No. of hours: 7**
Content:		

- Basics of Google Sheets interface
- Techniques for Data Cleaning and Analysis
- Creating and Analyzing Pivot Tables
- Hands-on Project: Data analysis with Google Sheets

Unit Number: 3	Title: Visualizations with Google Data Studio	No. of hours: 7

Content:

- Introduction to Google Data Studio
- Connecting Data Sources and Creating Reports
- Advanced Visualization Techniques
- Hands-on Project: Developing an interactive dashboard

Unit Number: 4	Title: Advanced Analytics with Google BigQuery	No. of hours: 7

Content:

- Overview of Google BigQuery
- Writing and Executing SQL Queries
- Data Import and Export Functions
- Integration with Google Data Studio
- Hands-on Project: Comprehensive data analysis using BigQuery

Learning Experience

Classroom Learning Experience

- **Hands-On Training:** Engage students in practical exercises using Google Analytics, Google Sheets, Google Data Studio, and Google BigQuery. The hands-on approach ensures that students can apply their learning to real-world data analysis tasks.
- **Interactive Labs:** Conduct lab sessions where students can explore and experiment with Google's tools under guided supervision. These labs provide a platform for students to deepen their understanding through experimentation.
- **Case Studies and Real-World Applications:** Incorporate case studies from industry to illustrate how data analytics with Google tools is applied in various sectors, such as e-commerce, marketing, and finance.

- **Collaborative Learning:** Encourage group projects where students collaborate on data analysis tasks, share insights, and critique each other's work, simulating a team-based work environment.
- **Guest Lectures and Webinars:** Invite industry professionals to share their experiences and best practices in data analytics using Google tools, offering students a glimpse into current trends and techniques.
- **Continuous Feedback:** Provide real-time feedback during lab sessions and project work, allowing students to improve their data analysis skills through iteration.

Outside Classroom Learning Experience

- **Project-Based Learning:** Throughout the course, students will work on projects that simulate real-life data analytics scenarios, such as setting up Google Analytics for a website, performing data cleaning and analysis in Google Sheets, and creating interactive dashboards in Google Data Studio.
- **Capstone Project:** The course will culminate in a comprehensive capstone project where students apply all the tools and techniques they've learned to analyze a large dataset and present their findings using Google Data Studio.
- **Self-Study:** Encourage students to explore advanced features of Google tools, such as API integration, advanced queries in BigQuery, and custom report creation in Google Analytics.
- **Peer Review:** Organize peer review sessions where students critique and provide feedback on each other's data analysis projects, helping to improve their skills in presenting and interpreting data.
- **Online Discussion Forums:** Facilitate online discussions where students can ask questions, share their experiences with Google tools, and troubleshoot challenges collaboratively.

Text Books and Online Resources

- "Google Analytics Breakthrough" by Feras Alhlou, Shiraz Asif, Eric Fettman
- "Learning Google Data Studio" by Mina Ozgen
- Google Analytics Academy (https://analytics.google.com/analytics/academy/)
- Google Data Studio Help (https://support.google.com/datastudio/)
- BigQuery Documentation (https://cloud.google.com/bigquery/docs)

Tools Used

- Google Analytics (https://analytics.google.com/)
- Google Sheets (https://www.google.com/sheets/about/)
- Google Data Studio (https://datastudio.google.com/)
- Google BigQuery (https://cloud.google.com/bigquery)

Software Testing using Open Source Frameworks

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Software Testing using Open Source Frameworks	VAC174	0-0-2	2
Type of Course:	Value Added Course (VAC)		
Pre-requisite(s):	None		

Course Perspective: This course provides hands-on experience in software testing using various open-source frameworks. Students will learn to set up a testing environment, develop automated test scripts, implement unit and integration tests, and integrate automated testing into CI/CD pipelines.

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding and set up a software testing environment using open source tools.
CO 2	Developing and execute automated test scripts using Selenium.
CO 3	Implementing unit and integration testing using JUnit and TestNG.
CO 4	Integrating automated testing into CI/CD pipelines using Jenkins.

Course Outline

Unit Number: 1	**Title: Overview of Software Testing**	**No. of hours: 7**
Content:		

- Importance, types, and life cycle of software testing
- Introduction to Open-Source Testing Tools: Selenium, JUnit, TestNG, Jenkins
- Setting Up the Testing Environment: Installing and configuring tools
- Version Control with Git and GitHub
- Hands-on Project: Set up a software testing environment and create a simple test script using Selenium

Unit Number: 2	**Title: Introduction to Selenium**	**No. of hours: 7**

Content:

- WebDriver, IDE, and Grid
- Writing Test Scripts: Locators, actions, and assertions
- Handling Web Elements: Forms, alerts, frames, and windows
- Test Execution: Running tests on different browsers and platforms
- Hands-on Project: Develop and execute automated test scripts for a sample web application using Selenium WebDriver

Unit Number: 3	**Title: Unit and Integration Testing with JUnit and TestNG**	**No. of hours: 7**

Content:

- Introduction to JUnit: Annotations, assertions, and test suites
- Writing Unit Tests: Best practices and patterns
- Introduction to TestNG: Annotations, groups, and data providers
- Integration Testing: Writing and executing integration tests
- Hands-on Project: Implement unit and integration tests for a sample application using JUnit and TestNG

Unit Number: 4	**Title: Continuous Testing with Jenkins**	**No. of hours: 7**

Content:

- Introduction to Jenkins: Setup and configuration
- Creating Jenkins Pipelines: Declarative and scripted pipelines
- Integrating Selenium, JUnit, and TestNG with Jenkins
- Continuous Testing: Running automated tests in CI/CD pipelines
- Hands-on Project: Set up a Jenkins pipeline to automate the testing process for a sample project, integrating Selenium, JUnit, and TestNG tests

Learning Experience

Classroom Learning Experience

- **Hands-On Learning:** Engage students in practical exercises that involve setting up testing environments, writing automated test scripts, and implementing unit and integration tests. The hands-on approach ensures that students gain real-world experience in software testing using open-source frameworks.
- **Interactive Labs:** Conduct lab sessions where students can explore different testing tools like Selenium, JUnit, TestNG, and Jenkins under guided supervision. These labs provide an opportunity for students to deepen their understanding through experimentation and practice.
- **Collaborative Learning:** Encourage group projects where students collaborate on testing tasks, share insights, and critique each other's work, simulating a team-based work environment often found in software development.
- **Case Studies and Real-World Applications:** Incorporate case studies from the industry to illustrate how software testing with open-source tools is applied in various sectors, such as e-commerce, finance, and enterprise software development.
- **Guest Lectures and Webinars:** Invite industry professionals to share their experiences and best practices in software testing using open-source frameworks, offering students a glimpse into current trends and techniques.
- **Continuous Feedback:** Provide real-time feedback during labs and projects, helping students improve their testing strategies through iteration and refinement.

Outside Classroom Learning Experience

- **Project-Based Approach:** Throughout the course, students will work on projects that simulate real-life testing scenarios, such as developing and executing automated tests for web applications, setting up CI/CD pipelines, and integrating testing tools.
- **Capstone Project:** The course will culminate in a comprehensive capstone project where students apply all the tools and techniques they've learned to develop a complete testing strategy for a real or simulated software project.

- **Self-Study:** Encourage students to explore additional open-source testing tools, advanced test automation strategies, and the integration of testing into DevOps workflows through online resources and tutorials.
- **Peer Review:** Organize peer review sessions where students evaluate each other's testing frameworks and approaches, providing constructive feedback to improve testing methodologies.
- **Online Discussion Forums:** Facilitate online forums where students can discuss challenges, share solutions, and collaborate on testing strategies, promoting peer learning and community engagement.

Books and Online Resources

- "Selenium Testing Tools Cookbook" by Unmesh Gundecha
- "JUnit in Action" by Petar Tahchiev, Felipe Leme, Vincent Massol, Gary Gregory
- "Continuous Integration: Improving Software Quality and Reducing Risk" by Paul M. Duvall, Steve Matyas, Andrew Glover
- Selenium Documentation (https://www.selenium.dev/documentation/)
- JUnit 5 User Guide (https://junit.org/junit5/docs/current/user-guide/)
- TestNG Documentation (https://testng.org/doc/)
- Jenkins User Documentation (https://www.jenkins.io/doc/)

Database Management with Open Source Frameworks

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Database Management with Open Source Frameworks	VAC175	0-0-2	2
Type of Course:	Value Added Course (VAC)		
Pre-requisite(s):	None		

Course Perspective: This course covers the fundamental concepts of database management systems (DBMS). Topics include data models, relational database design, SQL, transaction management, and database security. Students will learn to design and implement a relational database, write SQL queries, and understand the principles of database management.

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding and designing databases using Entity-Relationship (ER) diagrams.
CO 2	Applying SQL queries for data extraction, manipulation, and management.
CO 3	Utilizing MySQL and PostgreSQL for backend application development.
CO 4	Integrating databases with Python for data-driven applications.

Course Outline

Unit Number: 1	**Title: Introduction to Database Design**	**No. of hours: 7**
Content:		

- Overview of Database Management Systems (DBMS): Concepts and benefits
- Introduction to Open-Source RDBMS: MySQL, PostgreSQL
- Database Design: Principles and best practices
- Entity-Relationship (ER) Diagrams: Entities, relationships, attributes, and cardinality
- Hands-on Project: Design an ER diagram for a sample application (e.g., a library management system)

Unit Number: 2	**Title: SQL for Data Extraction and Manipulation**	**No. of hours: 7**

Content:

- Introduction to SQL: Basic syntax and structure
- Data Definition Language (DDL): CREATE, ALTER, DROP
- Data Manipulation Language (DML): SELECT, INSERT, UPDATE, DELETE
- SQL Joins: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN
- Hands-on Project: Write SQL queries to create and manipulate tables based on the ER diagram from Unit 1

Unit Number: 3	**Title: Advanced SQL and Database Operations**	**No. of hours: 7**

Content:

- Advanced SQL Queries: Subqueries, nested queries, and set operations
- Indexing and Optimization: Improving query performance
- Transactions and Concurrency: COMMIT, ROLLBACK, and transaction isolation levels
- Stored Procedures and Triggers: Creating and using stored procedures and triggers
- Hands-on Project: Develop complex SQL queries and procedures for the sample application, including indexing and optimization strategies

Unit Number: 4	**Title: Database Integration with Python**	**No. of hours: 7**

Content:

- Introduction to Python Database Connectivity: Using libraries like SQLAlchemy, Psycopg2, and MySQL Connector
- CRUD Operations with Python: Implementing create, read, update, and delete operations
- Data Analysis with Pandas: Loading and manipulating data from databases
- Real-World Project 1: Develop a Python application to interact with the MySQL/PostgreSQL database created in previous units
- Real-World Project 2: Perform data analysis on database data using Python and Pandas, creating visualizations of the results

Learning Experience

Classroom Learning Experience

- **Hands-On Learning:** Engage students in practical exercises where they design databases, write SQL queries, and integrate databases with Python applications. This hands-on approach ensures that students gain real-world experience in database management using open-source frameworks.
- **Interactive Labs:** Conduct lab sessions where students can explore MySQL, PostgreSQL, and Python database integration under guided supervision. These labs provide an opportunity for students to deepen their understanding through experimentation and practice.
- **Collaborative Learning:** Encourage group projects where students collaborate on database design and management tasks, share insights, and critique each other's work, simulating a team-based work environment often found in software development.
- **Case Studies and Real-World Applications:** Incorporate case studies from the industry to illustrate how database management using open-source tools is applied in various sectors, such as finance, e-commerce, and healthcare.
- **Guest Lectures and Webinars:** Invite industry professionals to share their experiences and best practices in database management using open-source frameworks, offering students a glimpse into current trends and techniques.

Outside Classroom Learning Experience

- **Project-Based Approach:** Throughout the course, students will work on projects that simulate real-life scenarios, such as designing and implementing a relational database, developing complex SQL queries, and creating Python applications that interact with databases.
- **Capstone Project:** The course will culminate in a comprehensive capstone project where students apply all the tools and techniques they've learned to design, implement, and manage a complete database system for a real or simulated application.

- **Self-Study:** Encourage students to explore advanced database management concepts like normalization, indexing, and database optimization through additional resources, such as documentation, tutorials, and online courses.
- **Peer Review:** Organize peer review sessions where students evaluate each other's database designs and implementations, providing constructive feedback on database structure and query optimization.
- **Online Discussion Forums:** Facilitate online forums where students can discuss challenges, share solutions, and collaborate on database management projects, promoting a peer-learning environment.

Tools Used

- MySQL (https://www.mysql.com/)
- PostgreSQL (https://www.postgresql.org/)
- SQLAlchemy (https://www.sqlalchemy.org/)
- Pandas (https://pandas.pydata.org/)

Cyber Security with Open Source Frameworks

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Cyber Security with Open Source Frameworks	VAC176	0-0-2	2
Type of Course:	Value Added Course (VAC)		
Pre-requisite(s):	None		

Course Perspective: This course is designed to provide hands-on experience in cyber security using various open-source tools and frameworks. Students will learn to identify, analyze, and mitigate security threats, implement security measures, and use open-source tools for network security, application security, and incident response. By the end of the course, students will be capable of securing systems and networks and conducting effective security assessments.

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the fundamentals of cyber security and the importance of using open-source tools.
CO 2	Implementing network security measures using open-source frameworks.
CO 3	Conducting application security assessments and vulnerability testing.
CO 4	Performing incident response and digital forensics using open-source tools.

Course Outline

Unit Number: 1	**Title: Introduction to Cybersecurity**	**No. of hours: 5**
Content:		

- Overview of Cyber Security: Concepts, importance, and threat landscape
- Introduction to Open-Source Security Tools: Kali Linux, Wireshark, Metasploit, Nmap
- Setting Up a Cyber Security Lab: Installing and configuring Kali Linux
- Basic Network Security: Understanding firewalls, VPNs, and IDS/IPS
- Hands-on Project: Setting up a cyber security lab environment and performing basic network scanning using Nmap

Unit Number: 2	**Title: Open Source Security Tools**	**No. of hours: 5**

Content:

- Introduction to open-source software and its benefits in cybersecurity
- Overview of key open-source security tools (Wireshark, Metasploit, Nmap, OpenVAS, Snort)
- Installation and configuration of security tools
- Use cases and practical applications of each tool

Unit Number: 3	**Title: Implementing Security with Open Source Frameworks**	**No. of hours: 5**

Content:

- Using OpenSSL for encryption and securing communications
- Implementing firewalls with pfSense
- Intrusion detection and prevention with Snort
- Vulnerability assessment with OpenVAS
- Network security monitoring with Zeek (formerly Bro)
- Security information and event management (SIEM) with ELK Stack (Elasticsearch, Logstash, Kibana)

Unit Number: 4	**Title: Advanced Topics and Case Studies**	**No. of hours: 5**

Content:

- Incident response and forensic analysis with open-source tools
- Penetration testing methodologies and tools
- Case studies of real-world cyber attacks and defenses
- Ethical hacking and legal considerations
- Project-based learning: securing a sample network
- Future trends in cybersecurity and open-source development

Learning Experience

Classroom Learning Experience

- **Hands-On Lab Environment:** Engage students in a practical, hands-on lab environment using Kali Linux and other open-source security tools. This experience helps students to familiarize themselves with real-world cybersecurity scenarios.
- **Interactive Learning:** Conduct interactive sessions where students actively participate in live demonstrations of security attacks and defenses, followed by group discussions on the implications and countermeasures.
- **Collaborative Projects:** Encourage students to collaborate on cybersecurity projects, fostering teamwork and the sharing of knowledge on different tools and techniques for securing systems.
- **Case Studies and Real-World Applications:** Incorporate case studies of major cyber incidents to help students understand the practical application of cybersecurity tools and strategies, as well as to learn from past incidents.
- **Guest Lectures and Industry Insights:** Invite cybersecurity professionals to share their experiences and insights on emerging threats, best practices in the industry, and the role of open-source tools in modern cybersecurity.
- **Continuous Feedback:** Provide ongoing feedback during lab exercises and projects, helping students improve their techniques in real-time and iterate on their approaches.

Outside Classroom Learning Experience

- **Project-Based Learning:** Throughout the course, students will work on projects that simulate cyber threats and defense mechanisms, including setting up firewalls, conducting penetration tests, and performing vulnerability assessments.
- **Capstone Project:** The course will culminate in a capstone project where students apply their skills to secure a simulated network environment, incorporating all the tools and techniques learned during the course.

- **Self-Study:** Encourage students to explore additional open-source cybersecurity tools, advanced penetration testing methods, and ethical hacking practices through independent research and online resources.
- **Peer Review:** Organize peer review sessions where students evaluate each other's cybersecurity defenses and strategies, providing constructive feedback on how to improve network security.
- **Online Discussion Forums:** Facilitate online forums where students can discuss challenges, share solutions, and collaborate on cybersecurity issues, promoting peer-to-peer learning and engagement.

Books and Online Resources

- "The Web Application Hacker's Handbook" by Dafydd Stuttard and Marcus Pinto
- "Metasploit: The Penetration Tester's Guide" by David Kennedy, Jim O'Gorman, Devon Kearns, and Mati Aharoni
- "Practical Malware Analysis" by Michael Sikorski and Andrew Honig
- Kali Linux Documentation (https://www.kali.org/docs/)
- OWASP ZAP Documentation (https://www.zaproxy.org/docs/)
- Wireshark Documentation (https://www.wireshark.org/docs/)

Tools Used

- Kali Linux (https://www.kali.org/)
- Nmap (https://nmap.org/)
- Wireshark (https://www.wireshark.org/)
- Metasploit (https://www.metasploit.com/)
- OWASP ZAP (https://www.zaproxy.org/)
- Snort (https://www.snort.org/)
- Autopsy (https://www.sleuthkit.org/autopsy/)
- Volatility (https://www.volatilityfoundation.org/)

Practical Robotics and UAV Applications

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Practical Robotics and UAV Applications	VAC185	0-0-2	2
Type of Course:	Value Added Course (VAC)		
Pre-requisite(s):	None		

Course Preface: This course provides comprehensive hands-on training in the field of robotics and UAV (Unmanned Aerial Vehicles). Students will learn to work with various robotic kits, develop basic robots, understand the working principles of UAVs, and gain practical knowledge of different robotics components and their programming. The course aims to equip students with the skills needed to start developing and working on various robotic applications, fostering innovation and practical problem-solving abilities.

The Course Outcomes (COs)

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

COs	Statements
CO 1	Identifying and describing various types and components of robots.
CO 2	Developing and debugging basic Arduino programs for robotic applications.
CO 3	Explaining and implementing key components and control algorithms for UAVs.
CO 4	Designing and creating simple robotic and UAV applications.

Course Outline

Unit Number: 1	**Title: Introduction to Robotics and Basic Components**	**No. of hours: 8**
Content:		

- Overview of Robotics: Types and applications of robots
- Mechanical Components
- Sensors
- Actuators
- Controllers and Microcontrollers
- Power Supply Components
- Communication Modules
- PCBs and Breadboards

Unit Number: 2	**Title: Arduino Programming and Development of Basic Robotic Applications**	**No. of hours: 8**

Content:

- Introduction to Arduino
- Basic Arduino Programming
- Controlling LEDs
- Motor Control
- Sensor Interfacing
- Communication Protocols
- Building Simple Robots:
  - Assembling robotic kits into functional robots
  - Calibrating and testing sensors and actuators
- Robot Programming:
  - Writing and debugging basic programs for robot control
  - Implementing basic navigation and obstacle avoidance algorithms
- Hands-On Projects:
  - Building and programming a line-following robot
  - Developing a simple robotic arm

Unit Number: 3	**Title: UAVs and Advanced Robotic Components**	**No. of hours: 8**

Content:

- Introduction to UAVs:
  - Types and applications of UAVs
  - Key components of UAVs (e.g., frame, motors, propellers, flight controllers)
- Advanced Robotics Components:
  - Working with advanced sensors (e.g., IMUs, GPS, LIDAR)
  - Communication modules (e.g., Bluetooth, Wi-Fi, ZigBee)
- UAV Programming and Control:
  - Basic flight control and stabilization algorithms
  - Mission planning and autonomous navigation

Unit Number: 4	**Title: Robotic Applications and Projects**	**No. of hours: 6**

Content:

- Simple Robotic Applications:
  - Developing a home automation system using robotics
  - Implementing a surveillance robot
- UAV Applications:
  - Creating a UAV for aerial photography
  - Developing a UAV for environmental monitoring (e.g., monitoring crops, water, and air quality)

Learning Experience

Classroom Learning Experience

- **Hands-On Learning:** Students will engage in extensive hands-on activities, assembling, programming, and testing various robotic kits and UAV components. This approach ensures that students gain practical experience and confidence in working with robotics and UAV technology.
- **Interactive Sessions:** Interactive workshops and sessions will be held to demonstrate the real-world applications of robotics and UAVs. Students will participate in live demonstrations, followed by hands-on replication of the projects.
- **Collaborative Learning:** Students will work in teams to complete complex projects, fostering collaboration and teamwork. This will also allow them to share different perspectives and solutions to common challenges in robotics and UAV development.
- **Problem-Solving Focus:** The course is designed to enhance problem-solving skills by challenging students to design and debug robotic systems, implement control algorithms, and troubleshoot issues in real-time.

- **Guest Lectures and Field Visits:** The course will include guest lectures from industry experts in robotics and UAVs, as well as potential field visits to relevant industries, providing students with insights into the latest trends and technologies.

Outside Classroom Learning Experience

- **Project-Based Approach:** The course emphasizes project-based learning, where students apply their knowledge to build functional robots and UAVs. This includes tasks such as developing a line-following robot, constructing a robotic arm, and programming UAVs for specific missions.
- **Capstone Project:** The course will culminate in a capstone project where students will integrate the skills they have learned to design and develop a complete robotic or UAV system. This project will demonstrate their ability to apply theoretical knowledge to practical, real-world problems.
- **Self-Study and Research:** Encourage students to explore advanced topics like sensor fusion, autonomous navigation, and UAV flight control through self-study and research to deepen their understanding.
- **Peer Review:** Organize peer review sessions where students evaluate each other's designs and code, providing constructive feedback on system performance and improvements.
- **Online Discussions and Collaboration:** Facilitate online forums where students can discuss technical challenges, share solutions, and collaborate on robotics and UAV projects, promoting continuous learning and engagement.

Books and Online Resources

- "Robotics: Modelling, Planning and Control" by Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo
- "Introduction to Robotics: Mechanics and Control" by John J. Craig
- "Learning ROS for Robotics Programming" by Enrique Fernández, Luis Sánchez, Anil Mahtani, Aaron Martinez
- "Robotics: Everything You Need to Know About Robotics from Beginner to Expert" by Peter Mckinnon
- "Unmanned Aerial Vehicles: Embedded Control" by Rogelio Lozano

Applied Automotive Engineering: Hands-On Practices and Innovations

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Applied Automotive Engineering: Hands-On Practices and Innovations	VAC186	0-0-2	2
Type of Course:	Value Added Course (VAC)		
Pre-requisite(s):	None		

Course Perspective: The Automotive Engineering course offers a practical, hands-on approach to understanding and applying the fundamentals of automotive technology. Emphasizing real-world applications, this course covers vehicle classification, IC engine mechanics, troubleshooting, and maintenance, alongside the integration of modern technologies. Students will engage in diagnostic testing, incorporate smart technologies, and develop live projects, equipping them with the skills and experiences necessary to excel in the automotive industry and address its evolving challenges. This course ensures that students are industry-ready, capable of innovative thinking and effective problem-solving.

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Planing and proposing live automotive projects based on hands-on learning.
CO 2	Developing and executing live projects with practical applications.
CO 3	Testing and evaluating the performance of live automotive projects.
CO 4	Presenting and documenting project outcomes effectively.

Course Outline

Unit Number: 1	**Title: Introduction to Automotive Engineering**	**No. of hours: 8**
Content:		

- Overview of automotive engineering
- Classification of vehicles
- Constructional details of automotive components
- Troubleshooting and Maintenance:
  - Basic troubleshooting techniques
  - Regular maintenance procedures
- Role of Accessories and Mountings:
  - Importance of automotive accessories
  - Mounting techniques and their significance

Unit Number: 2	**Title: Engines and Vehicle Components**	**No. of hours: 8**

Content:

- IC Engines: Overview and Constructional Details:
  - Basics of Internal Combustion (IC) engines
  - Constructional details of 2-stroke and 4-stroke engines
- Engine Types: Compression Ignition and Spark Ignition:
  - Differences between Compression Ignition (CI) and Spark Ignition (SI) engines
  - Practical understanding of engine components and functioning
- Troubleshooting and Maintenance:
  - Identifying and resolving common engine issues
  - Maintenance practices for IC engines
- Electrical and Electronics Components:
  - Overview of electrical and electronic components in vehicles
  - Hands-on exercises on component testing and maintenance

Unit Number: 3	**Title: Real-Time Projects/Hands-On Projects**	**No. of hours: 8**

Content:

- Problem Identification in Automobiles:
  - Techniques for identifying problems in automobiles
  - Practical exercises on vehicle diagnostics
- Testing of Automobiles:
  - Hands-on testing of automotive systems
  - Using diagnostic tools and equipment
- Incorporating Latest Smart Technology:
  - Adding and integrating smart technologies in vehicles
  - Practical modification exercises
- Project Execution and Documentation:
  - Planning and executing hands-on projects
  - Documenting project work and findings

Unit Number: 4	**Title: Live Project Development**	**No. of hours: 6**

Content:

- Project Planning and Proposal:
  - Planning live projects based on semester learning
  - Preparing project proposals and timelines
- Hands-On Project Development:
  - Executing live projects with hands-on practices
  - Collaborating and working in teams
- Testing and Evaluation:
  - Testing project outcomes and functionality
  - Evaluating performance and making necessary adjustments
- Final Presentation and Report:
  - Preparing and presenting the final project
  - Writing comprehensive project reports

Learning Experience

Classroom Learning Experience

- **Hands-On Learning:** This course is centered around practical, hands-on experience where students will engage directly with automotive components, engines, and diagnostic tools. Through structured workshops and lab sessions, students will apply theoretical knowledge to real-world automotive systems.
- **Interactive Workshops:** The course includes interactive workshops on automotive diagnostics, maintenance, and the integration of smart technologies. Students will gain experience in troubleshooting and maintaining various vehicle systems, ensuring a comprehensive understanding of automotive engineering.
- **Collaborative Learning:** Students will work in teams to tackle complex automotive projects, fostering collaboration and teamwork. This approach mirrors industry practices, preparing students for professional roles where cooperation and joint problem-solving are essential.
- **Real-Time Problem Solving:** The course emphasizes real-time problem-solving, where students will diagnose and fix issues in live automotive projects. This hands-on approach is designed to enhance critical thinking and technical skills.
- **Guest Lectures and Industry Visits:** The course will include guest lectures from automotive industry experts and potential field visits to automotive workshops or manufacturing facilities. These experiences provide insights into the latest industry trends and practices.

Outside Classroom Learning Experience

- **Project-Based Learning:** Students will participate in project-based learning, where they will design, develop, and test automotive projects. These projects will be aligned with current industry challenges, allowing students to innovate and find solutions to practical problems.
- **Capstone Project:** The course will culminate in a capstone project where students will integrate the skills and knowledge gained throughout the semester to develop a comprehensive automotive project. This project will involve planning, execution, testing, and final presentation, simulating a real-world engineering project lifecycle.
- **Self-Study and Research:** Encourage students to explore advanced automotive topics such as electric vehicles, autonomous systems, and hybrid technologies through self-study and research to deepen their understanding of modern innovations.
- **Peer Review:** Organize peer review sessions where students evaluate each other's projects and provide constructive feedback on design, performance, and problem-solving strategies.
- **Online Collaboration and Discussion:** Facilitate online forums where students can collaborate, share insights, and troubleshoot technical challenges in automotive engineering, promoting continuous learning and engagement.

Books and Online Resources

- "Robotics: Modelling, Planning and Control" by Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo
- "Introduction to Robotics: Mechanics and Control" by John J. Craig
- "Learning ROS for Robotics Programming" by Enrique Fernández, Luis Sánchez, Anil Mahtani, Aaron Martinez
- "Robotics: Everything You Need to Know About Robotics from Beginner to Expert" by Peter Mckinnon
- "Unmanned Aerial Vehicles: Embedded Control" by Rogelio Lozano

Practical Research Methodology for Engineers

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Practical Research Methodology for Engineers	VAC187	0-0-2	2
Type of Course:	Value Added Course (VAC)		
Pre-requisite(s):	None		

Course Perspective: This course provides students with a practical and hands-on approach to research methodology, focusing on the use of open-source tools and techniques relevant to various engineering domains. Students will learn the fundamentals of research methodology, explore a range of open-source tools, and apply these tools in conducting research, culminating in the preparation of an effective research paper.

The Course Outcomes (COs)

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

COs	Statements
CO 1	Identifying and explaining the fundamental principles of research methodology.
CO 2	Utilizing and demonstrating the use of open-source tools for data collection, analysis, and presentation.
CO 3	Developing and conducting research projects using appropriate methodologies and tools.
CO 4	Preparing and presenting a research paper effectively using open-source tools.

Course Outline

Unit Number: 1	**Title: Fundamentals of Research Methodology**	**No. of hours: 8**
Content:		

- Introduction to Research Methodology:
  - Definition and importance of research
  - Types of research: qualitative vs. quantitative, applied vs. fundamental
  - Research ethics and plagiarism
- Research Design and Planning:
  - Formulating research questions and hypotheses
  - Literature review techniques
  - Designing research methodologies: experimental, survey, case study
- Data Collection Methods:
  - Primary vs. secondary data
  - Techniques for data collection: surveys, interviews, observations
  - Sampling methods

Unit Number: 2	**Title: Open-Source Tools for Research**	**No. of hours: 8**

Content:

- Literature Review and Reference Management:
  - Using Zotero and Mendeley for reference management
  - Conducting literature reviews with Google Scholar and PubMed
- Data Analysis Tools:
  - Introduction to R and Python for statistical analysis
  - Using JASP and PSPP for statistical tests and analysis
- Survey and Data Collection Tools:
  - Designing surveys with Google Forms and LimeSurvey
  - Collecting and managing data with OpenRefine

Unit Number: 3	**Title: Conducting Research with Open-Source Tools**	**No. of hours: 8**

Content:

- Qualitative Data Analysis:
  - Using NVivo and QDA Miner for qualitative analysis
  - Coding and thematic analysis techniques
- Quantitative Data Analysis:
  - Advanced statistical techniques with R and Python
  - Data visualization with Matplotlib and ggplot2
- Document Preparation and Presentation:
  - Writing research papers with LaTeX and Overleaf
  - Creating presentations with Beamer and LibreOffice Impress

Unit Number: 4	**Title: Research Project and Paper Preparation**	**No. of hours: 6**

Content:

- Project Planning and Execution:
  - Selecting a research topic and formulating objectives
  - Planning and conducting experiments or surveys
- Data Analysis and Interpretation:
  - Analyzing collected data using appropriate tools
  - Interpreting results and drawing conclusions
- Writing and Presenting Research Paper:
  - Structuring and writing a research paper
  - Preparing presentations and posters for conferences

Learning Experience

Classroom Learning Experience

- **Hands-On Research Training:** This course provides hands-on experience in research methodology, allowing students to engage directly with data collection, analysis, and presentation tools. The practical approach ensures that students can apply theoretical knowledge in real-world research scenarios.
- **Interactive Workshops:** The course includes interactive workshops on various research tools and techniques. Students will learn to use open-source software for data analysis, reference management, and document preparation, providing them with essential skills for conducting high-quality research.
- **Collaborative Learning:** Students will collaborate in groups to tackle research problems, fostering teamwork and the exchange of ideas. This collaborative ap-

proach prepares students for professional research environments where interdisciplinary teamwork is often essential.

- **Real-Time Problem Solving:** The course emphasizes real-time problem-solving through research. Students will address actual research questions, develop hypotheses, and test them using appropriate methodologies, enhancing their critical thinking and analytical skills.
- **Guest Lectures and Research Seminars:** The course includes guest lectures and research seminars by experienced researchers and academics. These sessions provide insights into current research trends, best practices, and real-world challenges in conducting research.

Outside Classroom Learning Experience

- **Project-Based Learning:** Students will participate in project-based learning, where they will design and conduct their own research projects. These projects will cover all stages of the research process, from planning and data collection to analysis and presentation, ensuring a comprehensive learning experience.
- **Capstone Research Project:** The course culminates in a capstone research project where students apply all the skills and knowledge gained to conduct a full-fledged research study. This project includes planning, execution, data analysis, and the preparation of a research paper, simulating the entire research process.
- **Self-Study and Literature Review:** Encourage students to conduct extensive literature reviews and explore academic papers related to their research topics. This self-study enhances their understanding of existing research and aids in the development of their research questions.
- **Peer Review:** Organize peer review sessions where students evaluate each other's research proposals, hypotheses, and findings, providing constructive feedback to improve their research quality and methodologies.
- **Online Collaboration and Discussion:** Facilitate online forums where students can collaborate, share insights, and troubleshoot research challenges, promoting continuous learning and engagement beyond the classroom.

Books and Online Resources

- "Research Methodology: A Step-by-Step Guide for Beginners" by Ranjit Kumar
- "Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python" by Peter Bruce and Andrew Bruce
- "The LaTeX Companion" by Frank Mittelbach and Michel Goossens
- "Qualitative Data Analysis with NVivo" by Patricia Bazeley and Kristi Jackson

Semester: 4

Fundamentals of Algorithm Design & Analysis

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Fundamentals of Algorithm Design & Analysis	ENBC202	3-1-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the fundamental concepts of algorithm design and analysis, focusing on complexity analysis, algorithm design techniques, and graph algorithms. The course is divided into 4 units:

1. Introduction and Complexity Analysis
2. Divide and Conquer, Greedy Algorithms, and Dynamic Programming
3. Graph Algorithms
4. Advanced Algorithms and Techniques

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the basic concepts of algorithms and their importance in problem-solving.
CO 2	Analyzing the time and space complexity of algorithms.
CO 3	Applying various algorithm design techniques to solve problems.
CO 4	Implementing graph algorithms and understand their applications.

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

Course Outline

Unit Number: 1	Title: Introduction and Complexity Analysis	No. of hours: 10
Content:		

- Introduction to Algorithms: Definition, importance, specification, and role in problem-solving
- Algorithm Analysis: RAM computational models, Time and space complexity, Asymptotic Notations, best, average, and worst-case analysis
- Recurrence Relations: Solving recurrences using substitution and recursion tree
- Sorting: Analysis of Time complexities of comparison and Linear sorting Algorithms

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

- Divide and Conquer: General method, Merge Sort, Quick Sort, Binary Search
- Greedy Algorithms: Concept and characteristics, Fractional Knapsack, Activity Selection
- Dynamic Programming: General Method, Longest Common Subsequence, 0/1 Knapsack problem

Unit Number: 3	Title: Graph Algorithms	No. of hours: 10
Content:		

- Graph Representation: Adjacency matrix, adjacency list
- Graph Traversal Algorithms: Depth First Search (DFS), Breadth First Search (BFS)
- Shortest Path Algorithms: Dijkstra's algorithm, Bellman-Ford algorithm
- Minimum Spanning Tree Algorithms: Kruskal's algorithm, Prim's algorithm

Unit Number: 4	Title: Advanced Algorithms and Techniques	No. of hours: 10
Content:		

- Backtracking: Concept, examples (N-Queens problem, Sum of subsets)
- Branch and Bound: Concept, examples (Traveling Salesman Problem)
- String Matching Algorithms: Naive algorithm, Rabin-Karp algorithm

Learning Experience

Classroom Learning Experience

- **Interactive Lectures and Discussions:** This course offers interactive lectures that encourage active participation and discussions, allowing students to deepen their understanding of complex algorithmic concepts and clarify doubts in real-time.
- **Problem-Solving Sessions:** Regular problem-solving sessions are integrated into the course to help students apply theoretical concepts to practical problems. These sessions reinforce learning by providing hands-on experience in designing and analyzing algorithms.
- **Case Studies and Real-World Applications:** The course includes case studies and examples from various domains to illustrate the real-world applications of different algorithms. This approach helps students understand the practical significance of algorithm design and analysis.
- **Collaborative Learning:** Students will work in groups on certain assignments and projects, promoting collaborative learning and the exchange of ideas. This collaborative approach mirrors the teamwork often required in professional settings.
- **Use of Visual Tools:** Visual tools and software will be utilized to help students better understand algorithm behavior and performance. These tools provide a graphical representation of algorithms, making complex concepts easier to grasp.
- **Continuous Assessment and Feedback:** Students will receive continuous feedback through quizzes, assignments, and project evaluations. This feedback is designed to help them improve their understanding and performance throughout the course.

Outside Classroom Learning Experience

- **Algorithm Design Projects:** Students will engage in mini-projects focused on the design and implementation of algorithms. These projects provide an opportunity to work on real-world problems, fostering creativity and innovation in algorithm development.
- **Performance Analysis and Optimization:** The course emphasizes the importance of performance analysis and optimization of algorithms. Students will learn how to analyze the time and space complexity of their solutions and optimize them for better efficiency.
- **Self-Study and Research:** Encourage students to independently explore advanced algorithmic concepts, such as dynamic programming, greedy algorithms, and graph theory, through self-study and research.
- **Peer Review:** Organize peer review sessions where students critique and provide feedback on each other's algorithmic solutions, fostering a deeper understanding of optimization techniques and alternative approaches.

- **Final Project and Presentation:** The course culminates in a final project where students apply all the knowledge and skills they have acquired. They will design, implement, and present an algorithmic solution to a complex problem, demonstrating their mastery of the subject.

Text Books

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

Reference Books

- "Algorithms" by Robert Sedgewick and Kevin Wayne
- "The Algorithm Design Manual" by Steven S. Skiena

Additional Readings

Self-Learning Components:

1. Link to Algorithms course on NPTEL: https://nptel.ac.in/courses/106/106/106106131/
2. Link to Algorithms on Coursera: https://www.coursera.org/courses?query=algorithms
3. Link to Algorithms resources: https://www.geeksforgeeks.org/fundamentals-of-algorithms
4. Link to Algorithms tutorials: https://www.tutorialspoint.com/data_structures_algorithms/index.htm
5. Link to Algorithms lectures: https://ocw.mit.edu/courses/electrical-engineering-and-co 6-006-introduction-to-algorithms-fall-2011/

Introduction to Database Management Systems

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Database Management Systems	ENBC204	3-1-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the fundamental concepts of database management systems, focusing on database architecture, relational query languages, transaction processing, and database security. The course is divided into 4 units:

1. Introduction
2. Relational Query Languages
3. Transaction Processing and Storage Strategies
4. Advanced Topics and Database Security

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the basic concepts and architecture of database management systems.
CO 2	Using relational query languages to interact with databases.
CO 3	Managing transactions and apply storage strategies in database systems.
CO 4	Ensuring database security and explore advanced database topics.

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

Course Outline

Unit Number: 1	Title: Introduction	No. of hours: 12
Content:		

- Introduction to DBMS: Overview, benefits, and applications
- Database System Architecture: Schemas, Instances, Data abstraction, data models (network model, relational model, object-oriented data model)
- Entity-Relationship Model: Entity Types, Entity Sets, Attributes, and Keys, Relationship Types, ER diagrams
- Integrity Constraints: Primary key, foreign key, unique, not null, check constraints

Unit Number: 2	Title: Relational Query Languages	No. of hours: 8
Content:		

- Relational Database Design, Relational query languages, Relational algebra
- SQL: DDL (Data Definition Language), DML (Data Manipulation Language), DCL (Data Control Language)
- Query Processing and Optimization: Evaluation of relational algebra expressions, query optimization algorithms
- Database Design: Functional dependencies, normalization (1NF, 2NF, 3NF, BCNF)
- Overview of MySQL, Oracle, SQL Server

Unit Number: 3	Title: Transaction Processing and Storage Strategies	No. of hours: 12
Content:		

- Transaction Management: ACID properties, transaction states, serializability
- Concurrency Control: Lock-based protocols, timestamp-based protocols
- Database Recovery: Recovery concepts, recovery techniques
- Storage Strategies: File organization, indexing (single-level, multi-level), B-tree, B+ tree, hashing

Unit Number: 4	Title: Advanced Topics and Database Security	No. of hours: 8
Content:		

- Database Security: Authentication, authorization, access control
- Intrusion Detection: Techniques and tools, SQL injection prevention
- Introduction to Object-oriented databases and web databases
- Introduction to Distributed Databases: Concepts, architecture
- Introduction to Data Warehousing and Data Mining: Concepts, architecture

Learning Experience

Classroom Learning Experience

- **Interactive Lectures and In-Class Discussions:** The course includes interactive lectures that encourage students to engage with fundamental concepts of database management. These sessions are designed to provoke thought and foster a deep understanding of database architecture, relational models, and more.
- **Hands-On SQL Workshops:** Practical, hands-on workshops focusing on SQL provide students with the opportunity to write and optimize queries in a real-time environment. These workshops help bridge the gap between theory and practical application, enhancing student confidence in database management.
- **Case Studies and Practical Applications:** Through the analysis of case studies, students explore real-world applications of database management systems in various domains. This approach helps in understanding the relevance of database concepts in different industries and enhances problem-solving skills.
- **Use of Database Management Tools:** The course incorporates the use of popular database management tools like MySQL, Oracle, and SQL Server. These tools are used in lab sessions to provide students with a hands-on understanding of database operations, query optimization, and transaction processing.
- **Regular Assessments and Feedback:** Continuous assessment through quizzes, assignments, and lab exercises ensures that students receive timely feedback on their understanding of the course material. This helps in identifying areas of improvement and reinforcing learning.

Outside Classroom Learning Experience

- **Collaborative Group Projects:** Students will participate in group projects that simulate real-world database design and management tasks. These collaborative efforts allow students to apply concepts such as ER modeling, normalization, and transaction management in a team setting, mirroring industry practices.
- **Real-World Scenario Simulations:** Simulations of real-world scenarios, such as transaction processing and database recovery, are conducted to help students understand the challenges and solutions in managing large-scale databases. These simulations prepare students for practical challenges they may face in their careers.

- **Self-Study and Research:** Encourage students to independently explore advanced database concepts, including distributed databases, NoSQL databases, and cloud-based database management, through additional readings and tutorials.
- **Final Project and Presentation:** The course culminates in a final project where students design and implement a database system. This project, accompanied by a presentation, allows students to demonstrate their comprehensive understanding of the course material and their ability to apply it in a practical context.
- **Online Collaboration and Peer Review:** Facilitate online forums where students can collaborate, share insights, and review each other's database designs, promoting peer-to-peer learning and constructive feedback.

Text Books

- "Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan
- "Fundamentals of Database Systems" by Ramez Elmasri and Shamkant B. Navathe

Reference Books

- "An Introduction to Database Systems" by C.J. Date
- "Database Management Systems" by Raghu Ramakrishnan and Johannes Gehrke

Additional Readings

Self-Learning Components:

1. Link to Database Management Systems course on NPTEL: https://nptel.ac.in/courses/106/106/106106220/
2. Link to Database Management Systems on Coursera: https://www.coursera.org/courses?query=database%20management
3. Link to Database Management Systems resources: https://www.geeksforgeeks.org/dbms/
4. Link to Database Management Systems tutorials: https://www.tutorialspoint.com/dbms/index.htm
5. Link to Database Management Systems lectures: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-830-database-systems-fall-2012/

Introduction to Computer Networks

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Computer Networks	ENBC206	3-1-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the fundamental concepts of computer networks, focusing on the evolution of networking, data link layer, network layer, transport layer, and application layer. The course is divided into 4 units:

1. Evolution of Computer Networking
2. Data Link Layer
3. Introduction to Network Layer and Transport Services
4. Application Layer

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the basic components and evolution of computer networks.
CO 2	Explaining the data link layer and its protocols for error detection and correction.
CO 3	Describing the network layer, IP addressing, and transport services.
CO 4	Understanding the application layer protocols and their functionalities.

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

Course Outline

Unit Number: 1	Title: Evolution of Computer Networking	No. of hours: 10
Content:		
• Data communication components: Representation of data and its flow, Networks, Various connection topologies • Protocols and standards, OSI model, Access networks, physical media • Packet switching, Circuit switching, Network of networks, Packet delay and loss, End-to-end throughput		
Unit Number: 2	**Title: Data Link Layer**	**No. of hours: 10**
Content:		
• Error detection and error correction: Fundamentals, Block coding, Hamming distance, CRC • Flow control and error control protocols: Stop and Wait, Go back – N ARQ, Selective Repeat ARQ, Sliding Window • Multiple access protocols: Pure ALOHA, Slotted ALOHA, CSMA/CD		
Unit Number: 3	**Title: Introduction to Network Layer and Transport Services**	**No. of hours: 10**
Content:		
• Network Layer: Switching, Logical addressing – IPV4, IPV6 • Address mapping – ARP, RARP, BOOTP, and DHCP • Transport Layer: Process to Process Communication, User Datagram Protocol (UDP), Transmission Control Protocol (TCP), Congestion Control • Quality of Service: QoS improving techniques - Leaky Bucket and Token Bucket algorithm		
Unit Number: 4	**Title: Application Layer**	**No. of hours: 10**
Content:		
• Application Layer: Domain Name Space (DNS), TELNET, EMAIL, File Transfer Protocol (FTP), WWW, HTTP, SNMP • Bluetooth, Firewalls, Basic concepts of Cryptography		

Learning Experience

Classroom Learning Experience

- **Interactive Lectures and Discussions:** The course begins with interactive lectures, where fundamental networking concepts are introduced through engaging discussions. These sessions are designed to help students build a solid understanding of the evolution of computer networks, their architecture, and essential protocols.
- **Hands-On Lab Sessions:** Throughout the course, students participate in lab sessions that provide hands-on experience with network simulations and configurations. Using tools like Cisco Packet Tracer and Wireshark, students explore network topologies, configure network devices, and analyze packet flows in real-time.
- **Case Studies and Real-World Applications:** The course includes analysis of case studies that demonstrate the application of networking concepts in real-world situations. These case studies provide insights into the challenges and solutions in network design, security, and management across various industries.
- **Use of Networking Tools and Technologies:** The course incorporates the use of industry-standard networking tools and technologies. Students gain practical experience in configuring and managing network protocols, addressing schemes, and security mechanisms, preparing them for the demands of the industry.
- **Simulation-Based Learning:** Simulation tools are used to create virtual network environments where students can experiment with different network configurations and protocols. This approach enhances understanding of how networks operate and how various components interact within a network.
- **Continuous Assessment and Feedback:** Students receive regular assessments through quizzes, assignments, and lab exercises. Continuous feedback is provided to ensure students understand key concepts and can apply them effectively, allowing for ongoing improvement throughout the course.

Outside Classroom Learning Experience

- **Group Projects and Peer Learning:** Collaborative group projects are an integral part of the learning experience. Students work together to design and implement small-scale network models, applying the concepts learned in class to real-world scenarios. Peer learning is encouraged to foster teamwork and enhance problem-solving skills.
- **Self-Study and Research:** Encourage students to independently explore advanced networking topics such as routing protocols, network security, and wireless networks through self-study and research to deepen their understanding.
- **Final Project and Presentation:** The course culminates in a final project where students design and implement a comprehensive network solution. This project allows students to integrate their learning into a practical application, demonstrating their ability to design, configure, and troubleshoot computer networks effectively.
- **Online Collaboration and Peer Review:** Facilitate online forums where students can collaborate on network design challenges, share insights, and review each other's solutions, promoting peer-to-peer learning and constructive feedback.

- **Practice Labs and Simulations:** Provide supplementary lab exercises and simulations for students to practice network configurations and troubleshoot potential issues, reinforcing concepts outside of regular class hours.

Text Books

- "Data Communication and Networking" by Behrouz A. Forouzan, 5th Edition, McGraw-Hill, 2012
- "Computer Networks" by Andrew S. Tanenbaum and David J. Wetherall, Pearson, 5th Edition, 2010
- "Computer Networking: A Top-Down Approach" by James F. Kurose and Keith W. Ross, 5th Edition, Pearson

Reference Books

- "Introduction to Computer Networks" by Larry L. Peterson and Bruce S. Davie
- "Networking: A Beginner's Guide" by Bruce Hallberg

Additional Readings

Self-Learning Components:

1. Link to Computer Networks course on NPTEL: https://nptel.ac.in/courses/106/106/106106089/
2. Link to Computer Networks on Coursera: https://www.coursera.org/courses?query=computer%20networks
3. Link to Computer Networks resources: https://www.geeksforgeeks.org/computer-network-t
4. Link to Computer Networks tutorials: https://www.tutorialspoint.com/computer_fundamentals/computer_networking.htm
5. Link to Computer Networks lectures: https://ocw.mit.edu/courses/electrical-engineering-6-033-computer-system-engineering-spring-2018/lecture-videos/

Introduction to Database Management Systems Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Database Management Systems Lab	ENBC252	0-0-2	1
Type of Course:	Major		

Defined Course Outcomes

COs	**Statements**
CO 1	Understanding the fundamental concepts and architecture of database management systems.
CO 2	Developing proficiency in writing and optimizing SQL queries.
CO 3	Implementing transaction management and understand concurrency control mechanisms.
CO 4	Exploring advanced database topics including security, object-oriented databases, and data warehousing.

S.N	**Lab Task**	**Mapped CO/COs**
1	Write a program to understand the basic concepts and architecture of DBMS.	CO1
2	Develop an ER diagram for a given scenario.	CO1
3	Implement database schema based on ER diagram.	CO1
4	Write SQL queries to create and manipulate database tables using DDL commands.	CO2
5	Develop SQL queries for data insertion, updating, and deletion using DML commands.	CO2
6	Implement integrity constraints such as primary key, foreign key, unique, and not null.	CO2
7	Write complex SQL queries using joins, subqueries, and set operations.	CO2
8	Develop queries for aggregate functions and grouping of data.	CO2
9	Implement stored procedures and functions in SQL.	CO2
10	Write SQL queries to implement triggers and views.	CO2
11	Perform normalization up to BCNF for a given database schema.	CO2

S.N	Lab Task	Mapped CO/COs
12	Write queries to perform transaction management using ACID properties.	CO3
13	Implement concurrency control mechanisms using lock-based protocols.	CO3
14	Develop a program to demonstrate database recovery techniques.	CO3
15	Implement file organization and indexing techniques like B-tree and B+ tree.	CO3
16	Write a program to demonstrate hash-based indexing.	CO3
17	Implement database security mechanisms for authentication and authorization.	CO4
18	Develop a program to prevent SQL injection attacks.	CO4
19	Explore the concepts of object-oriented databases by implementing a basic object-oriented schema.	CO4
20	Develop a mini-project to demonstrate the concepts of data warehousing and data mining.	CO4
1	**Library Management System:** Develop a library management system using database concepts learned, including ER diagrams, SQL queries, and normalization.	CO1, CO2
2	**Hospital Management System:** Create a hospital management system with advanced SQL queries, transaction management, and security features.	CO2, CO3, CO4
3	**Online Retail Store:** Implement an online retail store database with product catalogs, customer orders, and inventory management using SQL and indexing techniques.	CO2, CO3
4	**Employee Management System:** Develop an employee management system incorporating transaction management, concurrency control, and recovery techniques.	CO3
5	**Student Information System:** Create a student information system with data warehousing and mining functionalities to analyze student performance data.	CO4

Online Learning Resources

- **GeeksforGeeks:** Tutorials and articles on database management systems. https://www.geeksforgeeks.org/dbms/
- **TutorialsPoint:** Comprehensive guides on database management systems and SQL. https://www.tutorialspoint.com/dbms/index.htm
- **NPTEL:** Video lectures and course materials on database management systems. https://nptel.ac.in/courses/106/106/106106093/

- **Coursera:** Courses on database management systems from leading universities. https://www.coursera.org/courses?query=dbms

Fundamentals of Algorithm Design & Analysis Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Fundamentals of Algorithm Design & Analysis Lab	ENBC254	0-0-2	1
Type of Course:	Major		

Defined Course Outcomes

COs	**Statements**
CO 1	Understanding the basic concepts of algorithms and their importance in problem-solving.
CO 2	Analyzing the complexity of algorithms using asymptotic notations and recurrence relations.
CO 3	Implementing algorithms using divide and conquer, greedy, and dynamic programming techniques.
CO 4	Developing and applying graph algorithms and understand advanced algorithms such as backtracking and branch and bound.

S.N	**Lab Task**	**Mapped CO/COs**
1	Write a program to understand the basic concepts and importance of algorithms.	CO1
2	Develop a program to analyze the time and space complexity of an algorithm.	CO2
3	Implement sorting algorithms and analyze their time complexities.	CO2
4	Solve recurrence relations using substitution and recursion tree methods.	CO2
5	Implement merge sort and quick sort using the divide and conquer method.	CO3
6	Write a program to perform binary search using the divide and conquer technique.	CO3
7	Develop a program for the fractional knapsack problem using the greedy algorithm.	CO3
8	Implement the activity selection problem using the greedy approach.	CO3

S.N	Lab Task	Mapped CO/COs
9	Write a program for the longest common subsequence problem using dynamic programming.	CO3
10	Implement the 0/1 knapsack problem using dynamic programming.	CO3
11	Develop a program to represent a graph using adjacency matrix and adjacency list.	CO4
12	Implement depth first search (DFS) and breadth first search (BFS) algorithms for graph traversal.	CO4
13	Write a program to find the shortest path using Dijkstra's algorithm.	CO4
14	Implement the Bellman-Ford algorithm for shortest path determination.	CO4
15	Develop a program to find the minimum spanning tree using Kruskal's algorithm.	CO4
16	Implement Prim's algorithm to find the minimum spanning tree.	CO4
17	Write a program to solve the N-Queens problem using backtracking.	CO4
18	Develop a program to solve the sum of subsets problem using backtracking.	CO4
19	Implement the Traveling Salesman Problem using the branch and bound technique.	CO4
20	Write a program for string matching using the naive algorithm and Rabin-Karp algorithm.	CO4
1	**Sorting Algorithm Analysis:** Develop a project to compare and analyze various sorting algorithms (merge sort, quick sort, bubble sort, etc.) in terms of their time and space complexities.	CO2
2	**Graph Traversal Visualizer:** Create a visual representation tool for graph traversal algorithms (DFS, BFS) to demonstrate their workings and applications.	CO4
3	**Dynamic Programming Solver:** Implement a tool that solves dynamic programming problems (0/1 knapsack, longest common subsequence) and provides step-by-step solutions.	CO3
4	**Shortest Path Finder:** Develop a project to find the shortest path in a graph using Dijkstra's and Bellman-Ford algorithms and compare their performance.	CO4
5	**Algorithm Efficiency Analyzer:** Create a project to analyze the efficiency of different algorithmic approaches (divide and conquer, greedy, dynamic programming) for solving common problems.	CO2, CO3, CO4

Online Learning Resources

- **GeeksforGeeks:** Tutorials and articles on algorithm design and analysis.
  https://www.geeksforgeeks.org/fundamentals-of-algorithms/
- **Coursera:** Courses on algorithms from leading universities.
  https://www.coursera.org/courses?query=algorithms
- **Khan Academy:** Lessons on algorithm design and analysis.
  https://www.khanacademy.org/computing/computer-science/algorithms
- **MIT OpenCourseWare:** Free course materials on algorithms.
  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/

Introduction to Computer Networks Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Introduction to Computer Networks Lab	ENBC256	0-0-2	1
Type of Course:	Major		

Defined Course Outcomes

COs	**Statements**
CO 1	Understanding the basic concepts and evolution of computer networking.
CO 2	Implementing error detection and correction, flow control, and multiple access protocols.
CO 3	Developing a comprehensive understanding of network and transport layer services and protocols.
CO 4	Exploring and implementing various application layer protocols and basic network security concepts.

S.N	**Lab Task**	**Mapped CO/COs**
1	Write a program to understand the basic concepts and evolution of computer networks.	CO1
2	Develop a simulation for data communication components and data flow representation.	CO1
3	Implement and analyze different network topologies.	CO1
4	Write a program to demonstrate the OSI model and its layers.	CO1
5	Develop a simulation for packet switching and circuit switching techniques.	CO1
6	Implement error detection techniques such as parity check, checksum, and CRC.	CO2
7	Write a program to implement error correction techniques using Hamming code.	CO2
8	Develop a simulation for flow control protocols: Stop and Wait, Go-Back-N ARQ, and Selective Repeat ARQ.	CO2
9	Implement multiple access protocols: Pure ALOHA, Slotted ALOHA, and CSMA/CD.	CO2
10	Write a program to simulate IP addressing and subnetting.	CO3

S.N	Lab Task	Mapped CO/COs
11	Develop a simulation for address mapping protocols: ARP and RARP.	CO3
12	Implement the basics of the transport layer: UDP and TCP protocols.	CO3
13	Write a program to demonstrate congestion control algorithms.	CO3
14	Develop a simulation for QoS techniques: Leaky Bucket and Token Bucket algorithms.	CO3
15	Implement a DNS lookup program.	CO4
16	Write a program to simulate email protocols (SMTP, POP3, IMAP).	CO4
17	Develop a simulation for file transfer using FTP.	CO4
18	Implement a simple web server and client using HTTP.	CO4
19	Write a program to demonstrate the basics of network security: cryptographic algorithms.	CO4
20	Develop a firewall simulation program.	CO4
1	**Network Topology Visualizer:** Develop a project to visualize and analyze different network topologies and their performance.	CO1
2	**Network Protocol Simulator:** Create a simulation tool to demonstrate the working of various network protocols (ARP, RARP, TCP, UDP).	CO2, CO3
3	**QoS and Congestion Control Analyzer:** Implement a project to analyze the impact of QoS techniques and congestion control algorithms on network performance.	CO3
4	**Network Security Suite:** Develop a suite of programs to implement basic network security measures including cryptography and firewall.	CO4
5	**Application Layer Protocol Simulator:** Create a simulation for various application layer protocols such as HTTP, FTP, DNS, and Email.	CO4

Online Learning Resources

- **GeeksforGeeks:** Tutorials and articles on computer networks and protocols. https://www.geeksforgeeks.org/computer-network-tutorials/
- **Coursera:** Courses on computer networks from leading universities. https://www.coursera.org/courses?query=computer%20networks
- **Khan Academy:** Lessons on computer networks. https://www.khanacademy.org/computing/computer-science/internet-intro
- **Cisco Networking Academy:** Courses and certifications on networking. https://www.netacad.com/

Communication & Personality Development

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Communication & Personality Development	AEC007	3-0-0	3
Type of Course:	AEC		
Pre-requisite(s):	None		

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

The Course Outcomes (COs)

On completion of the course, the participants will be 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.
CO 3	**Cultivating** essential soft skills required at the different workplaces.

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

Course Outline

Unit Number: 1	Title: Developing self and others	No. of hours: 8
Content:		
Content Summary: Self Awareness, Personality Concepts (Personality Assessments -Johari Window, Myers & Brigg), Self-Management, Self Esteem, Self-Efficacy, Interpersonal skills, mindset, grit and working in teams.		
Unit Number: 2	**Title: Enhancing Reading and Writing Skills**	**No. of hours: 6**
Content:		
Content Summary: Speed reading and its importance in competitive examinations, techniques for speed reading, note-taking, and critical analysis. Paragraph Writing, Essay and Summary writing, Business Letter, Email writing		
Unit Number: 3	**Title: Effective Communication and Public Speaking**	**No. of hours: 7**
Content:		
Content Summary: Communication Framework, barriers & overcoming these barriers, Group Discussions, Extempore & Public Speaking drills, to manage stage fright and anxiety. Structuring and organizing a presentation (Oral & PPT), Etiquettes, Grooming, Body Language and Conversation starters, TMAY		
Unit Number: 4	**Title: Career Guide and readiness**	**No. of hours: 15**
Content:		
Cover Letter, ATS friendly resume, Elevator Pitch, Video Resume (Visume), Networking, Group Discussion, Mock Interviews. Capstone Project		

Learning Experience

Classroom Learning Experience

- **Interactive Sessions:** Conduct interactive sessions that engage students in public speaking exercises, role-plays, and group discussions to enhance communication skills and self-confidence.
- **Workshops:** Organize workshops on personality assessment tools like the Johari Window and MBTI, helping students gain deeper insights into their personality traits and how these impact their communication.
- **Presentations:** Students deliver presentations on various topics, followed by feedback sessions to improve their verbal communication and presentation skills.
- **Simulation Exercises:** Use simulations and role-playing scenarios to teach conflict resolution, negotiation, and leadership skills, crucial for effective interpersonal communication.
- **Peer Feedback:** Implement a structured peer feedback system where students evaluate each other's communication style and effectiveness, promoting a culture of

continuous improvement.

Outside Classroom Learning Experience

- **Real-world Assignments:** Assign projects that require students to apply their communication skills in real-world settings, such as organizing an event or conducting interviews.
- **Online Discussions:** Facilitate moderated online forums where students can practice digital communication and engage in discussions on relevant topics to refine their writing and conversational skills.
- **Mentoring:** Pair students with mentors from the professional world to provide guidance, career advice, and feedback on their communication strategies and personality development efforts.
- **Community Engagement:** Encourage participation in community service or local groups where students can practice interpersonal skills, public speaking, and receive feedback from the community.
- **Reflective Journals:** Require students to keep journals reflecting on their communication experiences and personal growth, fostering self-awareness and continuous development.

Text Books

- "Personality Development and Soft Skills" by Barun K. Mitra
- "Business Communication and Personality Development" by C.S. Rayudu

Reference Books

- “Developing Communication Skills” by Krishna Mohan and Meera Banerji
- “The Time Trap: The Classic Book on Time Management” by R. Alec Mackenzie

Additional Readings

Self-Learning Components:

1. Link to Communication Skills course on NPTEL: https://nptel.ac.in/courses/109/104/109104031/
2. Link to Soft Skills on Coursera: https://www.coursera.org/courses?query=soft%20skills
3. Link to Time Management resources: https://www.mindtools.com/pages/main/newMN_HTE.htm
4. Link to Presentation Skills tutorials: https://www.skillsyouneed.com/present/presentation-skills.html
5. Link to Leadership Skills lectures: https://www.coursera.org/learn/leadership-skills

References

- R1: "Talking to Strangers" – Malcolm Gladwell
- R2: "Fierce Conversation" - Susan Scott
- R3: "Public Speaking" - William S. Pfeiffer, Pearson
- R4: "Soft Skills for Everyone" – Jeff Butterfield
- R5: "Business Communication" – Rajendra Pal, J S Korlahalli
- R6: "The Power of Positive Attitude" - Roger Fritz
- R7: "Believe in Yourself" – Dr. Joseph Murphy

J. Additional Readings

Websites & MOOCs

- https://www.16personalities.com
- https://www.tonyrobbins.com

Specific Research Papers

- GALLUP PRESS RESEARCH
- FRANKLIN COVEY LEADERSHIP CENTRE

Videos

- "The 7 Habits of Highly Effective People," Dr. Stephen R. Covey
- "I Am Not Your Guru," Tony Robbins

Podcast

- The Tim Ferriss Show

Magazines

- SUCCESS Magazine

Journals

- The IUP Journal of Soft Skills

Minor Project-II

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Minor Project-II	SIBC252	0-0-0	2
Type of Course:	Proj		
Pre-requisite(s):	None		

Duration

The minor project will last for three months.

Project Requirements

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

Guidelines

1. **Project Selection:**
   - Choose a societal problem relevant to the field of computer science and engineering.
   - Ensure the problem is specific and well-defined.
2. **Literature Review:**

- Conduct a thorough review of existing literature and solutions related to the problem.
- Identify gaps in existing solutions and potential areas for further investigation.

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

Evaluation Criteria for Minor Project (Out of 100 Marks)

1. **Understanding of Societal Problems (15 Marks):**
   - Comprehensive understanding of the problem: 15 marks
   - Good understanding of the problem: 12 marks
   - Basic understanding of the problem: 9 marks
   - Poor understanding of the problem: 5 marks
   - No understanding of the problem: 0 marks
2. **Critical Thinking and Analysis (20 Marks):**
   - Exceptional critical thinking and analysis: 20 marks
   - Good critical thinking and analysis: 15 marks
   - Moderate critical thinking and analysis: 10 marks
   - Basic critical thinking and analysis: 5 marks
   - Poor critical thinking and analysis: 0 marks

3. **Data Gathering and ETL Activities (20 Marks):**
    - Comprehensive and effective ETL activities: 20 marks
    - Good ETL activities: 15 marks
    - Moderate ETL activities: 10 marks
    - Basic ETL activities: 5 marks
    - Poor ETL activities: 0 marks
4. **Innovation and Idea Development (25 Marks):**
    - Highly innovative and feasible ideas: 25 marks
    - Good innovative ideas: 20 marks
    - Moderate innovative ideas: 15 marks
    - Basic innovative ideas: 10 marks
    - Poor innovative ideas: 5 marks
    - No innovative ideas: 0 marks
5. **Documentation Quality (10 Marks):**
    - Well-structured and detailed documentation: 10 marks
    - Moderately structured documentation: 7 marks
    - Poorly structured documentation: 3 marks
    - No documentation: 0 marks
6. **Presentation (10 Marks):**
    - Clear, concise, and engaging presentation: 10 marks
    - Clear but less engaging presentation: 7 marks
    - Somewhat clear and engaging presentation: 3 marks
    - Unclear and disengaging presentation: 0 marks

Total: 100 Marks

Course Outcomes

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

- **Understand Societal Issues:**
    - Demonstrating a basic understanding of societal problems and relevant issues within the concerned domain.
- **Critical Thinking:**
    - Thinking critically about formulated problems and existing solutions.
- **Data Management:**
    - Gathering relevant data and perform ETL activities to prepare the data for analysis.

- **Innovation and Entrepreneurship:**
  - Developing innovative ideas or entrepreneurial solutions to address identified problems.
- **Literature Review:**
  - Conducting comprehensive literature reviews and identify gaps in existing solutions.
- **Documentation:**
  - Documenting findings and analysis in a well-structured and appropriate format.
- **Presentation Skills:**
  - Presenting findings and analysis effectively, using clear and concise communication skills.
- **Problem Analysis:**
  - Analyzing problems from various perspectives and evaluate the effectiveness of existing solutions.
- **Professional Development:**
  - Developing skills in research, analysis, documentation, and presentation, contributing to overall professional growth.

Competitive Coding Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Competitive Coding Lab	SEC036	0-0-4	2
Type of Course:	SEC		

Defined Course Outcomes

COs	**Statements**
CO 1	Demonstrating the ability to implement and analyze basic data structures and algorithms for various computational problems.
CO 2	Developing and optimizing advanced data structures and their associated algorithms to solve complex problems efficiently.
CO 3	Applying dynamic programming and greedy algorithms to solve optimization problems and analyze their computational complexity.
CO 4	Implementing and evaluating graph algorithms for various real-world applications, focusing on shortest paths, spanning trees, and string matching problems.

S.N	**Lab Task**	**Mapped CO/COs**
1	Two Sum Problem: Find indices of two numbers that add up to a target number.	CO1
2	Reverse Integer: Reverse the digits of a given 32-bit signed integer.	CO1
3	Longest Substring Without Repeating Characters: Find the length of the longest substring without repeating characters.	CO1
4	Median of Two Sorted Arrays: Find the median of two sorted arrays.	CO1
5	Longest Palindromic Substring: Return the longest palindromic substring.	CO1
6	Zigzag Conversion: Convert a string into a zigzag pattern on a given number of rows.	CO1
7	Container With Most Water: Find two lines that together with the x-axis form a container that holds the most water.	CO1
8	Integer to Roman: Convert an integer to a Roman numeral.	CO1
9	Roman to Integer: Convert a Roman numeral to an integer.	CO1
10	Valid Parentheses: Determine if a string with characters (,), , , [, and] is valid.	CO1

S.N	Lab Task	Mapped CO/COs
11	Merge Two Sorted Lists: Merge two sorted linked lists into a single sorted list.	CO1
12	Remove Nth Node From End of List: Remove the nth node from the end of a linked list.	CO1
13	Valid Palindrome: Determine if a string is a palindrome, considering only alphanumeric characters.	CO1
14	Longest Common Prefix: Find the longest common prefix among an array of strings.	CO1
15	3Sum: Find all unique triplets in an array that sum up to zero.	CO1
16	Letter Combinations of a Phone Number: Return all possible letter combinations that a number could represent.	CO1
17	Generate Parentheses: Generate all combinations of well-formed parentheses.	CO1
18	Merge k Sorted Lists: Merge k sorted linked lists into a single sorted linked list.	CO1
19	Group Anagrams: Group anagrams together from an array of strings.	CO1
20	Maximum Subarray: Find the contiguous subarray with the largest sum.	CO1
21	Coin Change: Compute the fewest number of coins needed to make up a given amount.	CO1
22	Longest Increasing Subsequence: Return the length of the longest strictly increasing subsequence.	CO1
23	Edit Distance: Return the minimum number of operations required to convert one string to another.	CO1
24	Shortest Path in Binary Matrix: Return the length of the shortest clear path in an n x n binary matrix.	CO1
25	Dijkstra's Algorithm: Find the shortest path between nodes in a graph using Dijkstra's algorithm.	CO1
26	Kruskal's Algorithm: Find the minimum spanning tree of a given graph using Kruskal's algorithm.	CO1
27	Knapsack Problem: Find the maximum total value in a knapsack given weights and values of n items.	CO1
28	Bellman-Ford Algorithm: Find the shortest path from a single source vertex to all other vertices in a weighted graph.	CO1
29	Travelling Salesman Problem: Find the shortest possible route that visits each city exactly once and returns to the origin city.	CO1
30	Rabin-Karp Algorithm: Implement the Rabin-Karp algorithm for substring search.	CO1

Online Learning Resources

- **GeeksforGeeks:** Tutorials and articles on competitive coding problems and solutions.
  https://www.geeksforgeeks.org/competitive-programming/
- **LeetCode:** Platform for practicing coding problems and participating in coding contests.
  https://leetcode.com/
- **HackerRank:** Coding practice platform with problems from various domains.
  https://www.hackerrank.com/
- **Codeforces:** Online platform for competitive programming and coding contests.
  https://codeforces.com/

Competitive Coding Bootcamp- II

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Competitive Coding Bootcamp-II	-	2-0-0	0
Type of Course:	Audit		
Pre-requisite(s):	None		

Course Outcomes (COs):

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

Course Outline

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

Unit No.	Title	No. of hours
2	**Linked Lists, Stacks and Queues** **Content:** • Linked Lists: Singly and doubly linked lists: Creation, insertion, deletion, traversal. • Stacks and Queues: Stack operations: Push, pop, top, isEmpty. Queue operations: Enqueue, dequeue, front, isEmpty. • Applications: Parentheses matching, queue-based problems (e.g., sliding window problems from LeetCode).	8
3	**Sorting & Searching** **Content:** • Basic Sorting Algorithms: Implementing Bubble Sort, Selection Sort, Insertion Sort. • Advanced Sorting Algorithms: Implementing Merge Sort, Quick Sort, Heap Sort. • Binary Search: Implementing binary search for sorted arrays.	8
4	**Trees** **Content:** • Basic Tree Concepts: Introduction to tree terminology and operations, Tree Traversals: Preorder, inorder, postorder. • Binary Trees: Basic operations on binary trees: Insertion, deletion, searching. • Binary Search Trees: Understanding BST properties: Every left subtree is smaller, and every right subtree is larger.	6

Lab Experiments

Problem Statement	Mapped COs
Design a Parking Lot System using OOP concepts (Classes, Objects, Inheritance, Polymorphism).	CO1
Implement a Student Management System with Classes and Objects.	CO1
Create a Banking System with Constructors and Destructors.	CO1
Implement Method Overloading and Overriding in a chosen language.	CO1
Demonstrate Multiple Inheritance with a practical example.	CO1
Design a Library Management System with OOP principles.	CO1
Use Virtual Functions to implement polymorphism.	CO1
Implement Abstract Classes and Interfaces for a Payment System.	CO1

Problem Statement	Mapped COs
Create a simple calculator with Operator Overloading.	CO1
Build a Polymorphic class hierarchy (e.g., Shapes) to showcase polymorphism.	CO1
Reverse a Linked List (Iterative and Recursive).	CO2
Detect a cycle in a Linked List using Floyd's Cycle-Finding Algorithm.	CO2
Implement basic operations on a Singly Linked List (Insertion, Deletion).	CO2
Implement and traverse a Doubly Linked List.	CO2
Implement Stack operations (Push, Pop, Top) using arrays or linked lists.	CO2
Implement Queue operations (Enqueue, Dequeue, Front) using arrays or linked lists.	CO2
Solve the Parentheses Matching problem using Stack.	CO2
Implement Sliding Window Maximum using Deque.	CO2
Check for balanced parentheses using Stack.	CO2
Design a Circular Queue using linked list or array.	CO2
Implement Bubble Sort and analyze its time complexity.	CO3
Implement Merge Sort to sort an array of integers.	CO3
Find the Kth largest element in an array using Quick Sort.	CO3
Perform Binary Search to find an element in a sorted array.	CO3
Implement Heap Sort to sort a list of elements.	CO3
Find the position to insert an element in a sorted array using Binary Search.	CO3
Implement a custom sort based on frequency of elements.	CO3
Compare sorting results using Insertion Sort and Bubble Sort.	CO3
Perform Preorder, Inorder, and Postorder Traversal on a Binary Tree.	CO4
Find the Lowest Common Ancestor in a Binary Search Tree.	CO4
Implement an algorithm to check if a Binary Tree is balanced.	CO4
Determine if two Binary Trees are identical.	CO4
Find the maximum path sum in a Binary Tree.	CO4
Convert a Binary Search Tree to a Greater Tree.	CO4
Count the number of nodes in a complete Binary Tree.	CO4
Flatten a Binary Tree to a linked list using preorder traversal.	CO4
Serialize and deserialize a Binary Tree.	CO4
Find the diameter of a Binary Tree.	CO4
Check if a Binary Tree is a subtree of another Binary Tree.	CO4
Find the level order traversal of a Binary Tree.	CO4

Learning Experiences

- **Interactive Lectures:** Engage students with visual PPTs, encourage questions, and simplify complex concepts.
- **Problem-Based Assignments:** Assign theory problems and practical lab tasks (e.g., segment trees, shortest path algorithms).

- **Continuous Assessment:** Regular quizzes, mini-projects, and formative assessments to reinforce learning.
- **Peer Collaboration:** Group activities, case studies, and peer reviews for diverse perspectives.
- **Feedback and Support:** Instructors provide timely feedback, and students seek help as needed.
- **Life-Long Learning:** Foster curiosity, critical thinking, and skills beyond the syllabus.

Textbooks

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

Online References

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

Semester: 5

Computer Organization and Architecture

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Computer Organization and Architecture	ENBC301	3-1-0	4
Type of Course:	Major		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the fundamental concepts of computer organization and architecture, focusing on computer systems, memory hierarchy, processor design, and input/output systems. The course is divided into 4 units:

1. Introduction
2. Memory Hierarchy, Storage, and I/O
3. The Processor
4. Input/Output Systems and Advanced Topics

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the basic concepts of computer architecture and data representation.
CO 2	Explaining the memory hierarchy and storage systems.
CO 3	Describing the design and operation of processors.
CO 4	Understanding the input/output systems and advanced computer architecture topics.

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

Course Outline

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

- Introduction to Computer Architecture: Definitions and Concepts, Levels of abstraction, Von Neumann Architecture
- Functional Blocks of a Computer: CPU, memory, input-output subsystems, control unit
- Instruction Set Architecture (ISA) of CPU: Registers, instruction execution cycle, RTL (Register Transfer Language) interpretation of instructions, addressing modes, instruction set
- Types of Instruction Set Architectures: Reduced Instruction Set Computer (RISC) and Complex Instruction Set Computer (CISC)
- Data Representation: Number Systems (binary, octal, decimal, hexadecimal), Arithmetic Operations (addition, subtraction, multiplication, division)

Unit Number: 2	Title: Memory Hierarchy, Storage and I/O	No. of hours: 10
Content:		

- Memory Hierarchy: Types of memory: RAM, ROM, Cache, and Secondary Storage, SRAM vs. DRAM, Locality of reference
- Caching: Different indexing mechanisms: direct-mapped, set-associative, fully associative, Processor-cache interactions for read/write requests, Cache replacement policies: Least Recently Used (LRU), First-In-First-Out (FIFO)
- Storage: Introduction to magnetic disks, Flash memory: NAND and NOR flash
- I/O Data Transfer Techniques: Programmed I/O, Interrupt-Driven I/O, Direct Memory Access (DMA)

Unit Number: 3	Title: The Processor	No. of hours: 10
Content:		

- Building a Datapath: Introduction, Logic Design Conventions, A Simple Implementation scheme, Overview of Pipelining: Pipelined Datapath and Control, Data Hazards: Forwarding versus Stalling, Control Hazards and their mitigation
- Clocking Methodology: Revisiting clocking methodology, Amdahl's Law and its implications
- Processor Design: Single cycle processor design, Multi-cycle processor design, Instruction pipelining: stages and performance considerations

Unit Number: 4	**Title: Input/Output Systems and Advanced Topics**	**No. of hours: 10**

Content:

- I/O Systems: I/O Mapped vs. Memory-Mapped I/O, I/O Data Transfer Techniques: Programmed I/O, Interrupt-Driven I/O, Direct Memory Access (DMA)
- Storage Technologies: Introduction to Magnetic Disks: Tracks, Sectors, Flash Memory Technology: Structure and Performance Characteristics
- Cache Memory: Different Indexing Mechanisms: Direct-Mapped, Set-Associative, Fully Associative Caches, Processor-Cache Interactions for Read/Write Requests, Cache Replacement Policies: Least Recently Used (LRU), First-In-First-Out (FIFO)

Learning Experience

Classroom Learning Experience

- **Hands-On Hardware Labs:** The course integrates practical hardware labs where students interact with and assemble basic computer components. Students gain hands-on experience in understanding the physical architecture of computers by dismantling and reassembling hardware components such as processors, memory modules, and storage devices.
- **Simulation-Based Learning:** Students use simulation tools like Logisim and Multisim to design and analyze circuits and processors. These simulations allow students to visualize data flow and the functioning of different components within a computer system, reinforcing theoretical concepts with practical application.
- **Problem-Solving Workshops:** Regular workshops are conducted to solve complex problems related to memory management, cache optimization, and processor pipelining. These workshops are designed to enhance critical thinking and analytical skills, allowing students to apply theoretical knowledge in practical scenarios.
- **Case Studies and Industry Examples:** The course includes case studies and real-world examples from the computer industry, such as the design of modern CPUs by companies like Intel and AMD. These case studies help students understand how theoretical concepts are applied in the design and optimization of commercial processors.

- **Continuous Assessment and Feedback:** Students are continuously assessed through quizzes, assignments, and lab reports. Regular feedback is provided to help students identify areas of improvement and enhance their understanding of course material.
- **Guest Lectures and Industry Insights:** Guest lectures by industry professionals and experts in computer architecture are organized to provide students with insights into current trends and future directions in the field. These sessions are designed to bridge the gap between academic learning and industry practice.

Outside Classroom Learning Experience

- **Collaborative Group Projects:** The course includes group projects where students collaborate to design and implement a simple computer system. These projects emphasize the integration of various concepts such as memory hierarchy, processor design, and I/O systems, fostering teamwork and problem-solving skills.
- **Capstone Project:** The course culminates in a capstone project where students design a simplified, yet fully functional, processor or memory hierarchy system. This project synthesizes all the learning from the course and allows students to demonstrate their proficiency in computer organization and architecture.
- **Self-Study and Research:** Encourage students to explore advanced topics such as multicore processors, parallel processing, and hardware optimization through independent research and self-study.
- **Peer Review and Collaboration:** Organize peer review sessions where students provide constructive feedback on each other's designs and projects, promoting collaboration and improving problem-solving strategies.
- **Online Collaboration and Practice Tools:** Provide access to online tools and discussion forums where students can collaborate on projects, troubleshoot hardware simulations, and share insights on computer architecture challenges.

Text Books

- "Computer Organization and Design" by David A. Patterson and John L. Hennessy
- "Computer System Architecture" by M. Morris Mano

Reference Books

- "Computer Architecture: A Quantitative Approach" by John L. Hennessy and David A. Patterson
- "Structured Computer Organization" by Andrew S. Tanenbaum

Additional Readings

Self-Learning Components:

1. Link to Computer Organization course on NPTEL: https://nptel.ac.in/courses/106/103/106103180/
2. Link to Computer Architecture on Coursera: https://www.coursera.org/courses?query=computer%20architecture
3. Link to Computer Organization resources: https://www.geeksforgeeks.org/computer-organi
4. Link to Computer Organization tutorials: https://www.tutorialspoint.com/computer_fundamentals/computer_architecture.htm
5. Link to Computer Architecture lectures: https://ocw.mit.edu/courses/electrical-engineer 6-823-computer-system-architecture-fall-2005/

Ethical Hacking Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Ethical Hacking Lab	ENSP367	0-0-4	2
Type of Course:	Minor		

Defined Course Outcomes

COs	Statements
CO 1	Understanding the ethical implications and legal issues in hacking and penetration testing.
CO 2	Performing vulnerability assessments and penetration testing on real-world systems.
CO 3	Developing and implementing mitigation strategies for identified vulnerabilities.
CO 4	Analyzing and reporting the results of ethical hacking activities, including documentation and communication of findings.

Proposed Lab Experiments

Ex. No	Experiment Title	Mapped CO/COs
1	Setting Up a Penetration Testing Lab: Installing and Configuring Kali Linux and Metasploit.	CO 1
2	Performing Reconnaissance: Information Gathering Using Tools like Nmap, WHOIS, and Shodan.	CO 2
3	Network Scanning and Enumeration: Identifying Open Ports, Services, and Potential Vulnerabilities.	CO 2
4	Vulnerability Assessment: Using Nessus to Identify Vulnerabilities in a Target System.	CO 2, CO 3
5	Exploiting Vulnerabilities: Performing Exploitation Using Metasploit and Custom Scripts.	CO 2
6	Web Application Penetration Testing: Testing for SQL Injection, XSS, and CSRF Vulnerabilities.	CO 2, CO 3
7	Wireless Network Hacking: Cracking WEP/WPA/WPA2 Encryption and Exploiting Wireless Networks.	CO 2, CO 3
8	Social Engineering Attacks: Simulating Phishing Attacks and Analyzing Human Factors in Security.	CO 2, CO 4
9	Password Cracking: Implementing Brute Force and Dictionary Attacks on Passwords.	CO 2, CO 3
10	Post-Exploitation: Maintaining Access, Covering Tracks, and Extracting Sensitive Data.	CO 2, CO 4
11	Privilege Escalation Techniques: Escalating Privileges on a Compromised System.	CO 2, CO 3
12	Denial of Service (DoS) and Distributed DoS (DDoS) Attacks: Simulating and Mitigating Attacks.	CO 3
13	Exploiting and Securing IoT Devices: Vulnerability Assessment and Hardening IoT Devices.	CO 2, CO 3
14	Mobile Application Security Testing: Analyzing and Exploiting Mobile App Vulnerabilities.	CO 2, CO 3
15	Reverse Engineering and Malware Analysis: Analyzing and Dissecting Malware to Understand Its Functionality.	CO 2, CO 3
16	Developing and Deploying a Custom Exploit: Creating Exploits for Identified Vulnerabilities.	CO 2, CO 3
17	Comprehensive Penetration Test Report: Documenting Findings, Mitigation Strategies, and Presenting Results.	CO 4

Online Learning Resources

- **TryHackMe:** Online platform offering hands-on labs and challenges in ethical hacking.

https://tryhackme.com/

- **Hack The Box:** A platform for penetration testing and cybersecurity challenges. https://www.hackthebox.eu/
- **OWASP:** Resources and tools for web application security. https://owasp.org/
- **Metasploit Unleashed:** Free course on using the Metasploit Framework for penetration testing. https://www.offensive-security.com/metasploit-unleashed/

Summer Internship-II

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Summer Internship-II	SIBC351	0-0-0	2
Type of Course:	INT		

Duration

The minor project will last for three months.

Project Requirements

1. **Problem Identification and Analysis:**
   - Identify a relevant problem in society or industry.
   - Conduct a thorough analysis of the problem, considering various perspectives and implications.
2. **Implementation:**
   - Develop and implement a solution to address the identified problem.
3. **Data Visualization:**
   - Utilize appropriate data visualization techniques to represent the problem, solution, and outcomes effectively.
4. **Presentation of Solutions:**
   - Prepare a comprehensive presentation of the implemented solution, including its development process, outcomes, and impact.
5. **Case Studies:**
   - Conduct case studies related to the problem and solution, analyzing existing examples and drawing relevant insights.

Guidelines

1. **Project Selection:**
   - Choose a societal or industrial problem relevant to the field of computer science and engineering.
   - Ensure the problem is specific and well-defined.
2. **Literature Review:**

- Conduct a thorough review of existing literature and solutions related to the problem.
- Identify gaps in existing solutions and potential areas for further investigation.

3. **Implementation:**
   - Develop a detailed plan for implementing the solution.
   - Execute the implementation using appropriate tools, technologies, and methodologies.
4. **Data Visualization:**
   - Collect relevant data and use visualization techniques to represent the problem, solution, and outcomes.
   - Ensure the visualizations are clear, accurate, and effectively communicate the information.
5. **Documentation:**
   - Document the entire process, including problem identification, literature review, implementation, data visualization, and case studies.
   - Use appropriate formats and standards for documentation.
6. **Presentation:**
   - Prepare a presentation summarizing the problem, existing solutions, implementation process, data visualization, and case studies.
   - Ensure the presentation is clear, concise, and well-structured.

Evaluation Criteria for Minor Project (Out of 100 Marks)

1. **Problem Identification and Analysis (15 Marks):**
   - Comprehensive identification and analysis of the problem: 15 marks
   - Good identification and analysis of the problem: 12 marks
   - Basic identification and analysis of the problem: 9 marks
   - Poor identification and analysis of the problem: 5 marks
   - No identification and analysis of the problem: 0 marks
2. **Implementation (30 Marks):**
   - Successful and thorough implementation: 30 marks
   - Good implementation: 25 marks
   - Moderate implementation: 20 marks
   - Basic implementation: 15 marks
   - Poor implementation: 10 marks
   - No implementation: 0 marks
3. **Data Visualization (20 Marks):**

- Effective and clear data visualization: 20 marks
- Good data visualization: 15 marks
- Moderate data visualization: 10 marks
- Basic data visualization: 5 marks
- Poor data visualization: 0 marks

4. **Presentation of Solutions (15 Marks):**
   - Clear, concise, and engaging presentation: 15 marks
   - Clear but less engaging presentation: 12 marks
   - Somewhat clear and engaging presentation: 9 marks
   - Unclear and disengaging presentation: 5 marks
   - No presentation: 0 marks
5. **Case Studies (20 Marks):**
   - Comprehensive and insightful case studies: 20 marks
   - Good case studies: 15 marks
   - Moderate case studies: 10 marks
   - Basic case studies: 5 marks
   - Poor case studies: 0 marks

Total: 100 Marks

Course Outcomes

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

- **Identify and Analyze Problems:**
  - Identifying relevant societal or industrial problems and conduct a thorough analysis of these problems.
- **Implement Solutions:**
  - Developing and implement effective solutions to address identified problems using appropriate tools and technologies.
- **Visualize Data:**
  - Utilizing data visualization techniques to represent problems, solutions, and outcomes clearly and effectively.
- **Present Solutions:**
  - Preparing and deliver comprehensive presentations summarizing the implementation process, outcomes, and impact of their solutions.
- **Conduct Case Studies:**

  - Conducting case studies related to the problem and solution, analyzing existing examples and drawing relevant insights.

- **Literature Review:**
  - Conducting comprehensive literature reviews to identify gaps in existing solutions and potential areas for further investigation.

- **Documentation:**
  - Documenting the entire process, including problem identification, literature review, implementation, data visualization, and case studies, using appropriate formats and standards.

- **Professional Development:**
  - Developing skills in research, analysis, implementation, data visualization, documentation, and presentation, contributing to overall professional growth.

Arithmetic and Reasoning Skills

Program Name:	B.Sc (Hons.) Computer Science		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Arithmetic and Reasoning Skills	AEC008	3-0-0	3
Type of Course:	AEC		
Pre-requisite(s):	None		

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

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding arithmetic algorithms required for solving mathematical problems.
CO 2	Applying arithmetic algorithms to improve proficiency in calculations.
CO 3	Analyzing cases, scenarios, contexts and variables, and understanding their inter-connections in a given problem.
CO 4	Evaluating & deciding approaches and algorithms to solve mathematical & reasoning problems.

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

Course Outline

Unit Number: 1	Title: Mathematical Essentials	No. of hours: 15
Content:		
Vedic Maths, Classification of Numbers and Divisibility Rule, Percentage, Ratio and Proportion		
Unit Number: 2	**Title: Fundamentals of Logical Reasoning**	**No. of hours: 6**
Content:		
Blood Relations, Direction Sense, Coding Decoding		
Unit Number: 3	**Title: Elementary Quantitative Skills**	**No. of hours: 18**
Content:		
Simple and Compound Interest, Average, Partnership, Time and Work, Time Speed & Distance		
Unit Number: 4	**Title: Advanced Quantitative Skills**	**No. of hours: 6**
Content:		
Permutation & Combination, Probability		
Unit Number: 5	**Title: Employability Skills**	**No. of hours: 6**
Content:		

Learning Experience

Classroom Learning Experience

- **Lecture Sessions:** Conduct structured lectures on mathematical principles and reasoning techniques to establish a strong foundational understanding.
- **Interactive Problem Solving:** Engage students in interactive problem-solving sessions where they apply arithmetic algorithms and reasoning strategies to solve problems in real-time.
- **Group Activities:** Organize group activities that focus on collaborative problem solving, which can help students learn from peers and improve their communication and reasoning skills.
- **Practical Demonstrations:** Use practical demonstrations of arithmetic techniques, such as Vedic Maths, to show alternative and efficient methods of calculation.
- **Immediate Feedback:** Provide immediate feedback on students' problem-solving approaches, helping them refine their techniques and understand mistakes.

Outside Classroom Learning Experience

- **Online Quizzes and Exercises:** Offer online quizzes and exercises that students can use to practice and perfect their arithmetic and reasoning skills at their own pace.
- **Discussion Forums:** Create discussion forums where students can pose and answer arithmetic problems and logical puzzles, fostering a community of learning and peer support.
- **Study Groups:** Encourage the formation of study groups where students can meet regularly to discuss difficult concepts and solve complex problems together.
- **Real-world Applications:** Assign tasks that require students to apply their mathematical and reasoning skills to real-world scenarios, such as budget planning, scheduling, or strategic games.
- **Guest Lectures:** Arrange for guest lectures from professionals in fields that utilize advanced arithmetic and reasoning, providing students insights into practical applications and career opportunities.

Text Books

- "Quantitative Aptitude for Competitive Examinations" by R.S. Aggarwal
- "A Modern Approach to Logical Reasoning" by R.S. Aggarwal

References

R1. Guha Abhijit: Quantitative Aptitude for Competitive Examinations, Tata McGraw Hill Publication

R2. Quantitative Aptitude by R.S. Aggarwal

R3. Verbal & Non-Verbal Reasoning by R.S. Aggarwal

Additional Readings

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

Career Readiness Boot Camp

Program Name:	B.Tech, BCA, MCA, B.Sc		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Career Readiness Boot Camp	VAC IV	0-0-0	2
Type of Course:	VAC		
Pre-requisite(s):	None		

Course Perspective: The Boot Camp (Training and Placement) module is a comprehensive course designed to equip final-year B.Tech, BCA, MCA, and B.Sc students with the necessary skills and knowledge to excel in campus placement drives. All students are required to pass the individual components to receive the final marks out of 100 and earn 2 credits for the Practical Training Module (Bootcamp training) in their course structure. Students must obtain specific free certifications from Infosys Springboard (https://infytq.onwingspan.com/web/en/page/home).

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Applying data structures and algorithms to solve complex programming problems.
CO 2	Implementing object-oriented programming principles and develop robust Java applications.
CO 3	Designing and managing databases efficiently using advanced SQL and database management techniques.
CO 4	Demonstrating aptitude, soft skills, and interview readiness through practical evaluations.

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

Course Outline

Module: 1	Title: Data Structures and Algorithms - Part 1	No. of hours: 30 (Online, Self-Paced)
Content Summary:		
• Foundational data structures including arrays, strings, and linked lists • Key operations and practical applications		
Module: 2	**Title: Data Structures and Algorithms - Part 2**	**No. of hours: 30 (Online, Self-Paced)**
Content Summary:		
• Advanced data structures such as stacks, queues, trees, and graphs • Essential operations and real-world applications		
Module: 3	**Title: Object-Oriented Programming**	**No. of hours: 46 (Online, Self-Paced)**
Content Summary:		
• Fundamental concepts of OOP: classes, objects, inheritance, polymorphism, and encapsulation • Designing and implementing software using these principles		
Module: 4	**Title: Programming using Java**	**No. of hours: 113 (Online, Self-Paced)**
Content Summary:		
• Basics of Java: syntax, data types, operators, and control structures • Object-oriented principles specific to Java: classes, objects, inheritance, and polymorphism • Advanced topics: exception handling and file I/O		
Module: 5	**Title: Database Management Systems (Part I)**	**No. of hours: 64 (Online, Self-Paced)**
Content Summary:		

<table>
<tr><td colspan="3">Fundamental concepts of database systems: database models, relational databases, and SQL
Key topics: entity-relationship modeling, normalization, and basic query operations</td></tr>
<tr><td>Module: 6</td><td>Title: Database Management Systems (Part II)</td><td>No. of hours: 40 (Online, Self-Paced)</td></tr>
<tr><td colspan="3">Content Summary:</td></tr>
<tr><td colspan="3">Advanced database concepts: transaction management, concurrency control, and database security
Complex SQL queries, stored procedures, and triggers
Performance optimization techniques</td></tr>
<tr><td>Module: 7</td><td>Title: Aptitude Exam</td><td>No. of hours: Online</td></tr>
<tr><td>Module: 8</td><td>Title: Independent Evaluation through 3rd party</td><td>No. of hours: Offline</td></tr>
<tr><td>Module: 9</td><td>Title: Soft Skills</td><td>No. of hours: Online</td></tr>
<tr><td>Module: 10</td><td>Title: Mock Interview</td><td>No. of hours: Hybrid</td></tr>
</table>

Text Books

- "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
- "Cracking the Coding Interview" by Gayle Laakmann McDowell
- "Elements of Programming Interviews" by Adnan Aziz, Tsung-Hsien Lee, and Amit Prakash
- "Head First Java" by Kathy Sierra and Bert Bates
- "Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan
- "Introduction to the Theory of Computation" by Michael Sipser
- "Programming Challenges: The Programming Contest Training Manual" by Steven S. Skiena and Miguel A. Revilla
- "The Algorithm Design Manual" by Steven S. Skiena
- "Algorithms" by Robert Sedgewick and Kevin Wayne
- "Effective Java" by Joshua Bloch

Discipline Specific Elective - I (Cyber Security)

Secure Coding and Vulnerabilities

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Secure Coding and Vulnerabilities	ENSP301	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	Basic Programming and Cybersecurity Concepts		

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

1. Introduction to Coding and Security
2. Secure Application Design and Architecture
3. Secure Coding Practices and Vulnerabilities
4. Application Security Testing and Deployment

The Course Outcomes (COs)

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

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

A student is expected to have learned concepts and demonstrated/developed abilities or skills related to secure coding and vulnerabilities by the end of the course.

Course Outline

Unit Number: 1	Title: Introduction to Coding and Security	No. of hours: 10
Content Summary:		

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

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

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

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

- Input validation techniques: whitelist validation, regular expressions.
- Authentication and authorization.
- Cryptography.
- Buffer overflows.
- Session management and protection against session-related attacks.
- Secure error handling and logging practices.
- SQL Injection techniques and remedies.
- Race conditions.

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

Content Summary:

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

Learning Experience

Classroom Learning Experience

- **Hands-on Vulnerability Testing:** In-class practical exercises where students identify and mitigate common software vulnerabilities using open-source tools.
- **Code Review Sessions:** Peer-based code review sessions are conducted where students identify potential security flaws in each other's code, focusing on improving secure coding practices.
- **Case Studies:** Interactive analysis of real-world security breaches helps students understand how vulnerabilities are exploited and how they could be prevented through better security practices.
- **Interactive Labs:** Students will implement secure coding techniques such as input validation, buffer overflow protection, and session management in lab environments, enhancing their technical skills in secure development.

- **Role Play:** Instructors guide students through simulations where they alternate roles between attackers and defenders, learning how to exploit and mitigate vulnerabilities in various scenarios.
- **Collaborative Learning:** Group activities and discussions allow students to collaborate on designing security testing plans, assessing risks, and developing strategies for real-world applications.

Outside Classroom Learning Experience

- **Project-Based Learning:** Students develop secure applications using industry best practices in secure design, coding, and testing. These projects will reinforce the concepts learned during classroom sessions and labs.
- **Security Audits:** Students will perform security audits on sample applications in their own time, assessing vulnerabilities to threats such as SQL injection, session hijacking, and cross-site scripting.
- **Tools Exploration:** Through self-study and lab assignments, students will learn to use static and dynamic application security testing tools (SAST & DAST) on real-world applications, gaining proficiency in modern security assessment tools.
- **Real-World Simulations:** Students will conduct vulnerability scanning and penetration testing in simulated deployment environments, allowing them to practice what they've learned in a controlled, yet realistic, setting.
- **Collaborative Learning:** Outside the classroom, students will work in teams to design and execute security testing plans. They will assess security risks in various application environments and present their findings to peers and instructors for feedback.

Text Books and References

- **T1:** "Secure Coding: Principles and Practices" by Mark G. Graff, Kenneth R. Van Wyk, O'Reilly Media.
- **T2:** "Writing Secure Code" by Michael Howard and David LeBlanc, Microsoft Press, 2nd Edition, 2004.
- **T3:** "Buffer Overflow Attacks: Detect, Exploit, Prevent" by Jason Deckard, Syngress, 1st Edition, 2005.
- **T4:** "Threat Modeling" by Frank Swiderski and Window Snyder, Microsoft Professional, 1st Edition, 2004.
- **T5:** "The Software Vulnerability Guide (Programming Series)" by H. Thompson, Scott G. Chase, 2005.

Additional Readings

- **OWASP - Secure Coding Practices:** Quick Reference Guide. https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/

- **NPTEL - Secure Coding:** Offered by IITs through NPTEL, covering secure coding practices and principles.
  https://nptel.ac.in/courses/106105162
- **Mozilla Developer Network (MDN) - Web Security:** Comprehensive documentation on web security principles, secure coding practices, and common vulnerabilities.
  https://developer.mozilla.org/en-US/docs/Web/Security
- **Google Code University - Web Security:** Learn about web security from Google, including secure coding practices and how to protect web applications from common threats.
  https://code.google.com/archive/p/google-code-university/

Secure Coding and Vulnerabilities lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Secure Coding and Vulnerabilities lab	ENSP351	0-0-2	1
Type of Course:	Minor		

Defined Course Outcomes

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

S.N	**Experiment Title**	**Mapped CO/COs**
P1	Project Title: Secure Memory Management System Problem Statement: Develop a secure memory management system for a critical application such as a healthcare management system. This system should handle dynamic memory allocation and deallocation securely, preventing common memory management vulnerabilities.	CO1
P2	Project Title: Secure E-commerce Platform Design Problem Statement: Design and implement a secure e-commerce platform that ensures data security throughout the software development life cycle (SDLC). The platform should handle sensitive user information securely and provide a robust security architecture.	CO2

S.N	Experiment Title	Mapped CO/COs
P3	Project Title: Secure Banking Application Problem Statement: Develop a secure online banking application that ensures the protection of user data and prevents common vulnerabilities such as SQL injection, buffer overflow, and session hijacking.	CO3
P4	Project Title: Comprehensive Security Testing and Deployment for a Social Media Platform Problem Statement: Develop a social media platform with a focus on security testing and secure deployment. The platform should protect user data and provide a secure environment for social interactions.	CO4

Online Learning Resources

- **OWASP Documentation:** Comprehensive guide on secure coding practices. https://owasp.org/
- **SANS Institute:** Security training and resources for developers. https://www.sans.org/
- **NIST Secure Coding Guidelines:** Guidelines for secure software development. https://csrc.nist.gov/
- **Coursera:** Courses on secure coding and software security. https://www.coursera.org/courses?query=secure%20coding

Cyber Crime Investigation & Digital Forensics

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Cyber Crime Investigation & Digital Forensics	ENSP303	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	None		

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

1. Introduction
2. Types of Cyber Crimes
3. Investigation of Cyber Crimes
4. Forensic Tools and Processing of Electronic Evidence

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the nature and classification of conventional and cybercrimes.
CO 2	Analyzing and identify various types of cybercrimes and their modes of operation.
CO 3	Evaluating the impact of cybercrimes on individuals, organizations, and society.
CO 4	Developing an understanding of digital forensics and the investigative procedures used in cyber-crime cases.
CO 5	Applying forensic tools and techniques to retrieve and analyze digital evidence.

Course Outline

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

- Introduction to Digital Forensics, Definition and types of cybercrimes.
- Electronic evidence and handling, electronic media, collection, searching, and storage of electronic media.
- Introduction to internet crimes, hacking and cracking, credit card and ATM frauds.
- Web technology, cryptography, emerging digital crimes.

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

- Crimes targeting computers: Unauthorized Access, Packet Sniffing, Malicious Codes including Trojans, Viruses, etc.
- Online based Cyber Crimes: Phishing, Web Spoofing, E-mail Spoofing, Cyber Stalking, Web defacement, ATM and Card Crimes.
- Spamming, Commercial espionage, Commercial extortion, Piracy, Money Laundering.

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

- Investigation of malicious applications, agencies for investigation in India, their powers and constitution as per Indian laws.
- Procedures followed by first responders; evidence collection and seizure procedures.
- Securing the scene, documenting the scene, evidence collection and transportation, data acquisition and analysis.

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

- Introduction to forensic tools, usage of slack space, tools for disk imaging, data recovery, vulnerability assessment.
- Tools: Encase and FTK tools, Anti-Forensics, retrieving information, digital investigations.
- Processing digital evidence, recovering data from desktops, laptops, mobiles, damaged SIMs, multimedia evidence.

Learning Experience

Classroom Learning Experience

- **Interactive Lectures and Video Sessions:** Students will engage in interactive presentations and video lectures designed to enhance understanding through both visual and auditory learning techniques. These sessions will cover key concepts in cybercrime investigation and forensics.
- **Problem-Based Theory Assignments:** In-class theory assignments will focus on real-world cybercrime scenarios, helping students bridge the gap between theoretical concepts and their practical application in forensics.
- **Project-Based Lab Assignments:** Hands-on lab sessions will provide practical experience in evidence collection, data analysis, and the use of forensic methodologies. These assignments are designed to give students a real-world experience in digital forensics.
- **Collaborative Group Work:** Students will work together in groups to tackle complex case studies, promoting teamwork, critical thinking, and peer learning. This collaboration simulates real-world forensics team environments.
- **Continuous Assessment and Feedback:** Students will be assessed regularly through quizzes and lab sessions, receiving continuous feedback from instructors to help improve their understanding and practical skills.

Outside Classroom Learning Experience

- **ICT Tools and Moodle LMS:** Students will have access to course materials, notes, assignments, and model papers through the Moodle Learning Management System (LMS). This will allow them to continue their learning outside the classroom at their own pace.
- **Engagement with Question Bank and Model Papers:** A comprehensive question bank will be available to students, aiding in their exam preparation and helping them test their knowledge of key concepts in cybercrime investigation.
- **Application of Theoretical Knowledge to Practical Scenarios:** Students will apply their theoretical knowledge to practical case studies, simulating the full cycle of a cybercrime investigation, from evidence collection to report writing.

- **Collaborative Group Work Outside Class:** Students will continue group work outside the classroom, analyzing complex forensic cases and presenting their findings. This helps them develop practical problem-solving skills and enhances collaboration.
- **Self-Directed Learning and Continuous Improvement:** Students will engage in self-study, using ICT tools and other resources provided, and will receive ongoing feedback on their assignments, helping them continuously improve their forensic investigation skills.

Text Books & References

- Moore, Robert. (2011). *Cybercrime: Investigating High-Technology Computer Crime*, 2nd Ed. Elsevier.
- C. Altheide & H. Carvey. *Digital Forensics with Open Source Tools*, Syngress, 2011.
- Majid Yar. *Cybercrime and Society*, SAGE Publications, 2nd Ed, 2013.
- Robert M Slade. *Software Forensics: Collecting Evidence from the Scene of a Digital Crime*, Tata McGraw Hill, 2004.

Additional Readings

Online Learning References:

1. Cybrary - Digital Forensics. A free course covering tools, techniques, and procedures for investigating cybercrimes. **Link:** https://www.cybrary.it/course/digital-forensics/
2. Pluralsight - Digital Forensics Fundamentals. This course covers the fundamentals, tools, and techniques used in digital forensics. **Link:** https://www.pluralsight.com/courses/digital-forensics-fundamentals
3. SANS Institute - Digital Forensics and Incident Response Blog. A blog providing case studies and updates on digital forensics. **Link:** https://www.sans.org/blog/
4. OWASP - Open Web Application Security Project. Resources on web security, secure coding, and tools for vulnerability assessment. **Link:** https://www.owasp.org/

Cyber Crime Investigation & Digital Forensics lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Cyber Crime Investigation & Digital Forensics lab	ENSP353	0-0-2	1
Type of Course:	Minor		

Defined Course Outcomes

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

S.N	**Experiment Title**	**Mapped CO/COs**
1	Project Title: Comprehensive Study on Cybercrime and Digital Forensics Problem Statement: Conduct a comprehensive study on various types of cybercrimes and the role of digital forensics in investigating these crimes. The project will involve collecting electronic evidence, understanding cybercrime techniques, and applying digital forensics methodologies.	CO1
2	Project Title: Simulation and Prevention of Cyber Crimes Problem Statement: Develop a comprehensive simulation and prevention strategy for various types of cybercrimes. The project will involve creating scenarios for unauthorized access, phishing, and malware attacks, and implementing preventive measures.	CO2

S.N	Experiment Title	Mapped CO/COs
3	Project Title: Investigation and Reporting of Cyber Crime Incidents Problem Statement: Investigate a simulated cybercrime incident, collect and analyze digital evidence, and report the findings. The project will cover the entire investigation process from securing the scene to data analysis and reporting.	CO3
4	Project Title: Advanced Digital Forensics and Evidence Processing Problem Statement: Develop a system for advanced digital forensics and processing of electronic evidence. The project will involve using forensic tools for data recovery, vulnerability assessment, and processing digital evidence from various devices.	CO4

Online Learning Resources

- **NIST Computer Forensics Guidelines:** https://csrc.nist.gov/
- **SANS Institute Cyber Forensics Training:** https://www.sans.org/cyber-security-training/
- **OWASP Forensics Resources:** https://owasp.org/
- **Coursera:** Digital forensics courses from leading universities. https://www.coursera.org/courses?query=digital%20forensics

AI in Cyber Security

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
AI in Cyber Security	ENSP305	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	None		

Course Perspective: This course delves into the integration of Artificial Intelligence (AI) techniques within the realm of cyber security, highlighting the transformative potential of AI in detecting, preventing, and responding to cyber threats. As cyber threats evolve in complexity and scale, AI offers advanced methodologies to enhance security measures and mitigate risks effectively. The course provides a comprehensive understanding of the applications of AI in cyber security, covering fundamental machine learning and deep learning techniques, as well as their practical implementations in threat detection and prevention. Through detailed case studies and practical examples, students will explore the history, evolution, and current trends of AI in cyber security, gaining insights into the ethical considerations and challenges associated with AI technologies.

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the concepts and applications of AI in the field of cyber security.
CO 2	Expressing the ethical and legal considerations associated with the use of AI in cyber security.
CO 3	Determining emerging trends and technologies in AI for cyber security, and their potential impact on the field.
CO 4	Identifying strategies for integrating AI-driven solutions into existing cyber security frameworks, policies, and practices.
CO 5	Articulating critical thinking and problem-solving skills to address real-world cyber security challenges using AI techniques.
CO 6	Designing machine learning techniques for threat detection and prevention in cyber security, including supervised and unsupervised algorithms.

Course Outline

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

Content:		
• Overview of Artificial Intelligence and its applications in Cyber Security. • Evolution and impact of AI on Cyber Security. • Understanding Cyber Security threats and the role of AI. • Basic principles of Machine Learning (ML) and Deep Learning (DL) in Cyber Security. • Ethical considerations and challenges of AI in Cyber Security.		
Unit Number: 2	**Title: Machine Learning Techniques for Cyber Security**	**No. of hours: 10**
Content:		
• Introduction to Machine Learning techniques relevant to Cyber Security. • Overview of supervised and unsupervised ML models. • Feature engineering and data preparation for ML models. • Practical applications and case studies of ML in Cyber Security.		
Unit Number: 3	**Title: Deep Learning Techniques for Cyber Security**	**No. of hours: 10**
Content:		
• Introduction to Deep Learning and its significance in Cyber Security. • Applications of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). • Overview of Generative Adversarial Networks (GANs) and their use in Cyber Security. • Case studies illustrating the use of DL techniques for Cyber Security problems.		
Unit Number: 4	**Title: AI for Cyber Security: Threat Detection and Prevention**	**No. of hours: 10**
Content:		

- AI applications in threat detection and prevention.
- Overview of traditional vs. AI-driven threat detection methods.
- Fundamentals of supervised and unsupervised ML algorithms for threat detection.
- Advanced deep learning techniques for threat detection (CNNs, RNNs).
- Feature selection, emerging trends, and challenges in AI for Cyber Security.

Learning Experience

Classroom Learning Experience

- **Interactive Lectures:** Students will engage with multimedia-rich lectures that incorporate PowerPoint presentations (PPTs) and video materials, providing a comprehensive overview of AI applications in Cyber Security. These sessions will cover foundational concepts of Machine Learning (ML) and Deep Learning (DL) within the context of cybersecurity.
- **Hands-On Assignments:** In-class problem-based assignments and lab exercises will help students apply theoretical knowledge to practical, real-world cybersecurity scenarios. Students will focus on threat detection and prevention using ML and DL techniques.
- **Case Studies and Discussions:** Students will analyze and discuss case studies that showcase the practical application of ML and DL in Cyber Security. These discussions will foster critical thinking and problem-solving skills, enabling students to apply learned concepts to complex cybersecurity challenges.
- **Group Projects:** Students will collaborate in groups to work on AI-driven Cyber Security projects, gaining practical experience in designing and implementing secure systems. This group work promotes teamwork, peer learning, and hands-on problem-solving.
- **Continuous Assessment:** Regular quizzes and in-class assignments will be used to assess students' understanding and application of course materials. Feedback will be provided, allowing students to improve their performance and mastery of AI in Cyber Security concepts.

Outside Classroom Learning Experience

- **Use of ICT Tools:** Students will utilize Moodle LMS for accessing course materials, submitting assignments, and engaging in interactive discussions. The platform will enhance the learning experience by providing a centralized space for communication and collaboration.
- **Group Projects Outside Class:** Group collaboration will continue outside the classroom as students work on developing and implementing AI-driven cybersecu-

rity solutions. This allows for further teamwork and the application of concepts to real-world projects.

- **Hands-On Practice and Assignments:** Students will engage in self-directed learning, working on hands-on assignments and lab exercises at their own pace, applying ML and DL techniques to various cybersecurity scenarios.
- **Continuous Support and Feedback:** Students will benefit from continuous support and feedback from instructors. They will have opportunities to seek help and clarify doubts through online forums and one-on-one consultations, ensuring a deep understanding of complex topics.

Text Books & References

- Bhaskar Sinha. *Artificial Intelligence for Cybersecurity*, Auerbach Publications.
- Clarence Chio and David Freeman. *Machine Learning and Security: Protecting Systems with Data and Algorithms*, O'Reilly Media.

Additional Readings

Online Learning Resources:

1. Cybrary - Introduction to Artificial Intelligence for Cyber Security. Insights into how AI can be applied to cyber security, including threat detection and response. **Link:** https://www.cybrary.it/course/artificial-intelligence-for-cybersecurity/
2. Pluralsight - Machine Learning and AI for Cybersecurity. In-depth look at how machine learning and AI can be used to enhance cyber security measures. **Link:** https://www.pluralsight.com/courses/machine-learning-ai-cybersecurity
3. FutureLearn - Artificial Intelligence for Cyber Security by Coventry University. Explores the application of AI in cyber security, covering topics like threat detection, response, and mitigation. **Link:** https://www.futurelearn.com/courses/artificial-intelligence-for-cyber-security
4. MIT OpenCourseWare - Artificial Intelligence. Lecture notes, assignments, and exams from MIT's course on Artificial Intelligence, providing a deep dive into AI concepts applicable to cyber security. **Link:** https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2
5. IBM - Introduction to Cyber Security Tools & Cyber Attacks. A course covering various cyber security tools and techniques, including AI and machine learning for threat detection and prevention. **Link:** https://www.ibm.com/training/course/cyber-security-tools

AI in Cyber Security Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
AI in Cyber Security Lab	ENSP355	0-0-2	1
Type of Course:	Minor		

Defined Course Outcomes

COs	Statements
CO 1	Analyzing the history, evolution, and ethical considerations of AI in cyber security, documenting key milestones and advancements, and discussing the implications of AI applications.
CO 2	Implementing and evaluate machine learning models for classifying and detecting cyber threats, using various datasets and techniques such as supervised and unsupervised learning, deep learning, and anomaly detection.
CO 3	Developing and apply feature engineering, data preparation, and model training techniques to enhance the performance and accuracy of cyber security models.
CO 4	Conducting comprehensive case studies and surveys on the application of AI in cyber security, identifying emerging trends, challenges, and documenting methodologies and findings.

S.N	**Experiment Title**	**Mapped CO/COs**
P1	Project Title: Comprehensive Analysis of AI in Cyber Security Problem Statement: Conduct a comprehensive analysis of the role of AI in cyber security. The project will involve studying the history, evolution, and current trends in AI applications for cyber security, and understanding the basic principles of machine learning and deep learning in this context.	CO1
P2	Project Title: Machine Learning Models for Cyber Threat Detection Problem Statement: Develop and evaluate different machine learning models for detecting cyber threats. The project will involve implementing supervised and unsupervised learning techniques, performing feature engineering, and analyzing case studies.	CO2

S.N	Experiment Title	Mapped CO/COs
P3	Project Title: Deep Learning Models for Advanced Cyber Threat Detection Problem Statement: Develop and evaluate deep learning models for advanced cyber threat detection. The project will involve implementing CNNs, RNNs, and GANs, and analyzing their applications in cyber security.	CO3
P4	Project Title: AI-Based Comprehensive Threat Detection System Problem Statement: Develop a comprehensive AI-based system for threat detection and prevention in cyber security. The project will involve implementing machine learning and deep learning models and addressing the challenges of traditional threat detection methods.	CO4

Online Learning Resources

- **MIT AI Cybersecurity Lab Resources:**
  https://www.csail.mit.edu/research/ai-cybersecurity
- **Stanford AI for Cybersecurity Lectures:**
  https://ai.stanford.edu/research/security
- **Kaggle:** Datasets for machine learning models in cyber security.
  https://www.kaggle.com/
- **Coursera:** Courses on AI in cyber security from leading universities.
  https://www.coursera.org/courses?query=ai%20cybersecurity

Social Media Security

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Social Media Security	ENSP307	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	None		

Course Perspective: This course introduces students to the critical concepts of social media security, addressing the growing need to understand and manage security and privacy issues in the digital age. Social media platforms have become integral to personal, professional, and commercial interactions, creating a complex landscape of potential security threats and privacy concerns. The course equips students with the knowledge and skills to mitigate these risks effectively. Students will explore the technical, legal, and social dimensions of social media security, developing strategies to safeguard personal information, ensure user trust, and comply with legal standards.

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Demonstrating an understanding of the different types of social media platforms, their features, and their impact on communication, marketing, and society.
CO 2	Acquiring knowledge and skills in social media monitoring techniques, including data collection, analysis, and the use of relevant tools and technologies.
CO 3	Analyzing and evaluate viral content on social media, understand the factors contributing to its spread, and recognize its implications for marketing and online engagement.
CO 4	Identifying the challenges, opportunities, and pitfalls associated with social media marketing, and formulate strategies for effective audience targeting, engagement, and brand promotion.
CO 5	Developing strategies to safeguard personal information, foster user trust, and mitigate associated risks.

Course Outline

Unit Number: 1	**Title: Social Media Overview**	**No. of hours: 10**
Content Summary:		

- Overview of Social Media.
- Types and Platforms.
- Social Media Monitoring and Analysis.
- Data Collection and Analysis Methods: BoW Model, TF-IDF.
- Network Analysis Basics: Node Centrality, Degree Distribution, Clustering Coefficient.
- Introduction to Synthetic Networks: Random Graphs, Preferential Attachment.

Unit Number: 2	**Title: Social Media Management and Marketing**	**No. of hours: 10**

Content Summary:

- Strategies for Recruitment and Employment Screening.
- Customer Engagement and Content Management.
- Evaluating Social Media Campaigns: Effective vs. Ineffective.
- Ethical Considerations and Privacy in Crowdsourcing.
- Managing and Promoting Social Media Presence.

Unit Number: 3	**Title: Privacy Issues in Social Media**	**No. of hours: 10**

Content Summary:

- Privacy Settings and Personal Identifiable Information (PII) Leakage.
- Types of Privacy Attacks: Identity Disclosure, Inference Attacks, De-anonymization.
- Privacy Metrics: k-anonymity, l-diversity, Differential Privacy.
- Balancing Personalization and Privacy.
- Impact of Privacy on User Trust.

Unit Number: 4	**Title: Social Media Security: Laws, Best Practices, and Case Studies**	**No. of hours: 10**

Content Summary:

- Legal Aspects: Posting and Content Regulations.
- Best Practices: Content Moderation, User Authentication.
- Security Awareness and Education.
- Case Studies: Facebook, Twitter, Instagram, YouTube, LinkedIn, and others.

Learning Experience

Classroom Learning Experience

- **Interactive Lectures and Discussions:** Use lecture PPTs and video lectures to introduce key topics in social media trends, followed by in-class discussions on current trends and their implications in the digital world. These sessions encourage critical thinking and engagement with the latest developments in social media.
- **Hands-On Data Analysis:** In-class practical sessions will allow students to collect, process, and analyze social media data using tools such as Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). These activities provide hands-on experience with data analytics in a social media context.
- **Group Social Media Campaign Projects:** Students will collaborate in teams to design, implement, and evaluate mock social media campaigns. These projects involve applying marketing strategies, content management principles, and performance evaluation techniques to develop and promote a campaign.
- **Case Study Analysis:** Students will analyze real-world case studies of social media platforms to understand how different platforms operate, their business models, and the ethical and operational issues they face.
- **Privacy and Security Workshops:** Workshops will provide interactive sessions on privacy settings, privacy metrics, and security measures related to social media. Students will engage with practical tools to explore how privacy and security are managed on different platforms.
- **Ethical Dilemma Debates:** Students will participate in debates on ethical issues and privacy concerns surrounding social media usage. These debates will encourage students to explore both sides of pressing issues such as data privacy, user manipulation, and platform responsibility.
- **Continuous Assessment and Feedback:** Quizzes, assignments, and peer reviews will be regularly conducted to assess student understanding. Continuous feedback will be provided to help students improve their analytical and project management skills.

Outside Classroom Learning Experience

- **Moodle LMS and ICT Integration:** Students will use Moodle LMS to access course materials, submit assignments, and participate in discussions. This

integration of ICT tools enhances learning by offering a centralized platform for communication and collaboration.

- **Practical Management Tools Sessions:** Outside the classroom, students will work on mastering social media management tools such as Hootsuite, Buffer, or Sprout Social. These tools will aid in content moderation, scheduling, and campaign promotion, providing valuable real-world experience.
- **Group Collaboration on Social Media Campaigns:** Teams will continue collaborating outside of class to design, implement, and manage their social media campaigns, applying theories and concepts learned during classroom sessions.
- **Independent Data Analysis:** Students will engage in self-directed learning by conducting data collection and analysis projects outside the classroom. They will use their knowledge of BoW, TF-IDF, and other analytical tools to examine social media trends.
- **Preparation for Ethical Debates:** Students will research and prepare for ethical dilemma debates on their own or in groups, investigating privacy and security issues in social media and forming arguments based on evidence and critical analysis.

Text Books & References

- Bonzanini Marco. *Mastering Social Media Mining*, Packt Publishing Limited.
- Mikhail Klassen and Matthew A. Russell. *Mining the Social Web*, O'Reilly Media, Inc.
- Zafarani, Reza, Mohammad Ali Abbasi, and Huan Liu. *Social Media Mining: An Introduction*, Cambridge University Press.
- Michael Cross. *Social Media Security: Leveraging Social Networking While Mitigating Risk*, Syngress.
- Daxton R. Stewart. *Social Media and the Law: A Guidebook for Communication Students and Professionals*, Taylor & Francis Ltd.
- Henry A. Oliver. *Security in the Digital Age: Social Media Security Threats and Vulnerabilities*, Create Space Independent Publishing Platform.

Additional Readings

Online Learning Resources for Social Media Security

1. Coursera - Social Media Marketing Specialization. **Provider:** Northwestern University. **Link:** https://www.coursera.org/specializations/social-media-marketing
2. edX - Cybersecurity Fundamentals. **Provider:** Rochester Institute of Technology. **Link:** https://www.edx.org/course/cybersecurity-fundamentals
3. Udemy - The Complete Cyber Security Course: Network Security! **Instructor:** Nathan House. **Link:** https://www.udemy.com/course/the-complete-cyber-security-cours

Social Media Security Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Social Media Security Lab	ENSP357	0-0-2	1
Type of Course:	Minor		

Defined Course Outcomes

COs	**Statements**
CO 1	Analyzing different social media platforms, their features, and the ethical and privacy considerations in crowdsourcing and data handling.
CO 2	Implementing data collection, text analysis, and network analysis techniques on social media datasets, demonstrating proficiency in using APIs and various models.
CO 3	Developing strategies and plans for using social media in various contexts such as employment screening, customer engagement, and small business promotion.
CO 4	Evaluating privacy settings, metrics, and security incidents on social media platforms, applying best practices for user authentication, access control, and content moderation.

S.N	**Experiment Title**	**Mapped CO/COs**
P1	Project Title: Comprehensive Analysis of Social Media Platforms Problem Statement: Conduct a comprehensive analysis of different social media platforms, their features, and the data they generate. The project will involve collecting and analyzing data from social media, performing content analysis, and understanding network properties.	CO1
P2	Project Title: Effective Social Media Management and Marketing Strategy Problem Statement: Develop an effective social media management and marketing strategy for a small business. The project will involve analyzing customer engagement, creating marketing strategies, and addressing ethical considerations in social media use.	CO2

S.N	Experiment Title	Mapped CO/COs
P3	Project Title: Privacy Protection in Social Media Problem Statement: Develop strategies and tools to protect user privacy on social media platforms. The project will involve analyzing privacy settings, simulating privacy attacks, and evaluating privacy metrics.	CO3
P4	Project Title: Enhancing Security and Compliance on Social Media Platforms Problem Statement: Develop a comprehensive approach to enhance security and ensure compliance with laws on social media platforms. The project will involve researching laws, developing best practices, and analyzing case studies of security incidents.	CO4

Online Learning Resources

- **Facebook Security and Privacy Documentation:**
  https://www.facebook.com/safety
- **Twitter API Documentation:**
  https://developer.twitter.com/en/docs
- **Kaggle:** Datasets for social media analysis.
  https://www.kaggle.com/
- **Coursera:** Courses on social media security and privacy.
  https://www.coursera.org/

Discipline Specific Elective - II (Full Stack Development)

Mobile Application Development using iOS

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Mobile Application Development using iOS	ENSP409	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	None		

Course Perspective: The objective of the course is to provide skills to develop applications for OS X and iOS. It includes an introduction to the development framework Xcode. Objective-C is used as a programming language to develop applications. Objective-C is the superset of the C programming language and provides object-oriented capabilities and a dynamic runtime. Objective-C inherits the syntax, primitive types, and flow control statements of C and adds syntax for defining classes and methods. The course is divided into 4 modules:

1. Introduction to IDE and SDK of iOS App Development
2. Swift Programming
3. Encapsulating Data
4. Developing iOS Applications

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the fundamental concepts of variables, constants, and basic data types in SWIFT.
CO 2	Analyzing the use of control flow statements such as for, if, and switch in various programming scenarios.
CO 3	Applying object-oriented concepts in SWIFT, including the use of classes, structures, and protocols.
CO 4	Creating functions, closures, and extensions to enhance code modularity and reuse.
CO 5	Evaluating error handling techniques and type checking mechanisms to develop robust SWIFT applications.

A student is expected to have learned concepts and demonstrated abilities or skills related to mobile application development using iOS at the end of the course.

Course Outline

Unit Number: 1	Title: Introduction to SWIFT Language	No. of hours: 10
Content:		
• Variables & Constants, Introduction to functions (methods), Arrays, Dictionaries, Data, Date and other basic data types, Enums, structures, closures • For, If, switch statement, Object-oriented concepts with SWIFT • Type check, AnyObject, Any Protocols, Extensions, Error handling, Working with classes		
Unit Number: 2	**Title: Working with Xcode**	**No. of hours: 8**
Content:		
• Introduction to XCODE, COCOA touch framework, iOS application architecture, Application lifecycle		
Unit Number: 3	**Title: Introduction to View Controllers and Views**	**No. of hours: 12**
Content:		
• View Controllers, view, view lifecycle, Basic Controls – Label, Buttons, Text field, image View, Table view with default cells and customized cells • Collection view with default cells and customized cells, Picker view, Date picker, scroll view, navigation and Tab bar controller • Understanding Interface builder, XIB files, Creating outlets and Actions, Handling touch and gesture events, Segment and Page control, switch view, UIAlertView		
Unit Number: 4	**Title: Integrating with Database**	**No. of hours: 10**
Content:		
• Introduction to data storage methods in iOS, Using Core Data, SQLite database, User Defaults, Property List		

Learning Experience

Classroom Learning Experience

- **Hands-On Development:** Students will actively engage in building iOS applications using Xcode, starting from the basics of Swift programming to developing

complex apps with integrated databases. This hands-on approach ensures that students gain practical experience in coding, debugging, and deploying iOS apps.

- **Exploration of iOS Ecosystem:** Students will explore various components of the iOS ecosystem, including the use of Apple's frameworks, design patterns, and best practices for app development. This exploration helps students create apps that are efficient, user-friendly, and aligned with industry standards.
- **Real-World Case Studies:** The course will incorporate case studies of successful iOS applications, analyzing their design, architecture, and development process. This analysis helps students learn from existing apps and apply these lessons to their own projects.
- **Guest Lectures and Industry Insights:** Industry experts will be invited to share their experiences and insights into mobile application development. These sessions provide valuable industry perspectives, keeping students updated on the latest trends and best practices in iOS development.
- **Continuous Feedback and Assessment:** Regular assessments through quizzes, assignments, and project presentations ensure that students are on track with their learning. Continuous feedback helps students improve their coding skills and understand the nuances of iOS app development.

Outside Classroom Learning Experience

- **Project-Based Learning:** The course includes comprehensive projects where students design and develop complete iOS applications. These projects help students apply theoretical knowledge to real-world scenarios, such as creating user interfaces, managing app states, and working with data persistence.
- **Team Collaboration:** Students will work in teams to develop iOS apps, fostering collaboration, communication, and project management skills. These team projects simulate professional environments, preparing students for industry roles.
- **Advanced Topics and Emerging Trends:** Students will be introduced to advanced topics such as Core Data, integrating with cloud services, and using third-party libraries. Additionally, emerging trends in iOS development, such as SwiftUI and ARKit, will be explored, encouraging students to stay ahead in the rapidly evolving field of mobile development.
- **Self-Study and Research:** Encourage students to independently explore additional iOS development topics such as accessibility features, performance optimization, and app store submission guidelines through self-study and research.
- **Peer Collaboration and Review:** Organize peer review sessions where students provide feedback on each other's iOS projects, fostering a collaborative learning environment and helping students improve their design and coding skills.

Text Books

- "iOS 14 Programming for Beginners: Kickstart your iOS app development journey with the Swift programming language and Xcode 12, 6th Edition" by Ahmad Sahar and Craig Clayton

- "Mastering iOS 14 Programming: Build professional-grade iOS applications with Swift 5 and Xcode 12" by Ahmad Sahar and Craig Clayton

Reference Books

- "iOS 14 Programming for Beginners: Kickstart your iOS app development journey with the Swift programming language and Xcode 12, 6th Edition" by Ahmad Sahar and Craig Clayton
- "Mastering iOS 14 Programming: Build professional-grade iOS applications with Swift 5 and Xcode 12" by Ahmad Sahar and Craig Clayton

Additional Readings

Self-Learning Components:

1. Apple Developer Documentation
   Description: Comprehensive documentation and tutorials for iOS app development using Swift and Xcode.
   Link: https://developer.apple.com/documentation/
2. Ray Wenderlich: iOS and Swift Tutorials
   Description: A collection of high-quality tutorials and courses on iOS app development, covering Swift, Xcode, and various iOS frameworks.
   Link: https://www.raywenderlich.com/ios
3. GitHub: iOS Development Resources
   Description: A curated list of open-source projects, libraries, and resources for learning and improving iOS development skills.
   Link: https://github.com/vsouza/awesome-ios

Mobile Application Development using iOS Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Mobile Application Development using iOS Lab	ENSP459	0-0-2	1
Type of Course:	Minor		

Defined Course Outcomes

COs	**Statements**
CO 1	Understanding and apply fundamental concepts of iOS development using Xcode and the Cocoa Touch framework to build robust and user-friendly applications.
CO 2	Developing interactive and dynamic user interfaces in iOS applications using view controllers, views, and gesture recognizers.
CO 3	Creating and manage user interfaces and view controllers in iOS applications using Xcode, demonstrating proficiency in Interface Builder and UIKit components.
CO 4	Developing interactive and dynamic user interfaces in iOS applications using view controllers, views, and gesture recognizers.

S.N	**Experiment**	**Mapped CO/COs**
1	Set up the iOS development environment by installing Xcode. Create a simple "Hello, World!" iOS application to familiarize with the Xcode IDE and Swift programming basics.	CO1
2	Develop a basic iOS application that demonstrates the use of Swift syntax, variables, data types, and control flow. Create a simple calculator app to perform basic arithmetic operations.	CO1
3	Use Xcode and Interface Builder to design a user interface for an iOS app. Create a simple user interface with labels, buttons, and text fields, and handle user interactions.	CO3
4	Implement a simple iOS app to demonstrate the app lifecycle and navigation between view controllers. Create a multi-screen app that navigates between different views using navigation controllers.	CO1

S.N	Experiment	Mapped CO/COs
5	Design a responsive user interface using Auto Layout and the constraint system. Create an iOS app with a login screen that adjusts to different screen sizes and orientations.	CO2
6	Implement navigation between different views using storyboards and segues. Create a multi-screen app with a main menu and detailed views for each menu item.	CO2
7	Implement gesture recognition and touch event handling in an iOS app. Create an app that responds to tap, swipe, and pinch gestures to perform different actions.	CO2
8	Implement data persistence using Core Data. Create an iOS app that allows users to add, edit, and delete notes, and save them to a local database.	CO1
9	Use User Defaults and the file system to store and retrieve user preferences and data. Create an app that saves user settings and displays them when the app is reopened.	CO1
10	Implement offline data storage and synchronization. Create an iOS app that allows users to add data while offline and syncs with a remote server when the device is back online.	CO1
11	Implement advanced UI components and animations in an iOS app. Create a visually appealing app with custom views, animations, and transitions between screens.	CO3
12	Access and use iOS sensors and hardware features. Create an app that uses the camera to take photos, and the GPS to display the user's current location on a map.	CO2
13	Debug and test an iOS app using Xcode's debugging tools. Implement unit tests and UI tests to ensure the app functions correctly under different scenarios.	CO4

Online Learning Resources

- **Apple Developer Documentation:** Comprehensive guide on iOS development and using Xcode.
  https://developer.apple.com/documentation/
- **Swift Documentation:** Resources for learning and using Swift for iOS development.
  https://swift.org/documentation/
- **Ray Wenderlich:** Tutorials and guides on iOS app development.
  https://www.raywenderlich.com/
- **Coursera:** Courses on iOS app development from leading universities.
  https://www.coursera.org/courses?query=ios%20app%20development

DevOps & Automation

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
DevOps & Automation	ENSP411	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	None		

Course Perspective: Throughout the subject, students will engage in hands-on exercises and projects to gain practical experience with various DevOps tools and practices. By the end of the course, students will be well-equipped to embrace the DevOps culture and apply automation techniques to enhance software development, delivery, and operations processes. The course is divided into 4 modules:

1. Introduction to DevOps
2. Version Control and CI/CD
3. Containerization and Orchestration
4. Configuration Management and Monitoring

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding the principles and benefits of DevOps, and its role in enhancing collaboration and efficiency between development and operations teams.
CO 2	Acquiring hands-on experience with popular DevOps tools such as Git, Jenkins, Docker, Kubernetes, and Ansible for implementing continuous integration, continuous delivery, and automated deployment processes.
CO 3	Demonstrating proficiency in containerization and orchestration techniques using Docker and Kubernetes for efficient and scalable application deployment and management.
CO 4	Implementing configuration management and Infrastructure as Code (IaC) using Ansible and Terraform to automate the provisioning and management of infrastructure resources.
CO 5	Developing skills in monitoring, logging, and security practices in the context of DevOps, ensuring application performance, resilience, and adherence to security best practices.

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

Course Outline

Unit Number: 1	Title: Introduction to DevOps	No. of hours: 12
Content:		

- DevOps Principles and Culture: Understand the core principles of DevOps and its cultural impact. Collaboration, automation, continuous integration, continuous delivery, and continuous deployment.
- DevOps Toolchain: Overview of tools and technologies used in DevOps practices. Introduction to popular DevOps tools like Git, Jenkins, Docker, Kubernetes, and Ansible.
- Version Control with Git: Branching, merging, and collaborative development using Git. Continuous Integration (CI): Setting up CI pipelines with Jenkins for automated building and testing.
- Continuous Delivery and Deployment: Implementing CD pipelines for deploying.

Unit Number: 2	Title: Version Control and CI/CD	No. of hours: 8
Content:		

- Version Control with Git: Version control concepts, Git workflows, and collaboration strategies.
- Continuous Integration with Jenkins: Setting up Jenkins pipelines, automated testing, and deployment.
- Maven Integration: Integrate Maven for dependency management and building projects.

Unit Number: 3	Title: Containerization and Orchestration	No. of hours: 8
Content:		

- Introduction to Docker: Docker concepts, container management, and Docker file creation.
- Container Orchestration with Kubernetes: Kubernetes architecture, deployment, scaling, and networking.
- Docker Compose: Managing multi-container applications with Docker Compose.

Unit Number: 4	Title: Configuration Management and Monitoring	No. of hours: 12
Content:		

- Configuration Management with Ansible: Ansible playbooks, roles, and infrastructure automation.
- Infrastructure as Code (IaC): Terraform for provisioning and managing infrastructure.
- Monitoring and Logging: Monitoring tools, log management, and application performance monitoring in DevOps.
- Security in DevOps: Implementing security best practices in CI/CD pipelines and containerized environments.

Learning Experience

Classroom Learning Experience

- **Hands-On Tool Usage:** Students will gain practical experience with a wide range of DevOps tools, such as Git, Jenkins, Docker, Kubernetes, Ansible, and Terraform. Through labs and assignments, they will learn to implement continuous integration, continuous delivery, and automated deployment processes.
- **Industry-Relevant Case Studies:** The course will feature case studies from leading tech companies, analyzing how they implement DevOps practices and automation in their operations. This will provide students with insights into industry standards and best practices.
- **Exploration of Automation Techniques:** Students will explore various automation techniques in DevOps, including the use of scripting, configuration management tools, and Infrastructure as Code. This will enable them to automate repetitive tasks and manage large-scale infrastructures efficiently.
- **Real-Time Monitoring and Logging:** Students will learn to implement monitoring and logging solutions to ensure application performance, availability, and security. They will work with tools like Prometheus, Grafana, and ELK Stack to gain insights into system health and troubleshoot issues effectively.
- **Continuous Feedback and Iterative Learning:** Regular assessments, including quizzes, assignments, and peer reviews, will be used to gauge student understanding and provide feedback. This iterative approach will allow students to continuously improve their skills and knowledge.

Outside Classroom Learning Experience

- **Project-Based Learning:** The course will include projects where students will work on real-world scenarios, such as setting up CI/CD pipelines, containerizing applications, and managing infrastructure as code. These projects will help students apply their knowledge in practical contexts, preparing them for industry roles.
- **Collaboration and Teamwork:** Students will engage in collaborative projects that mimic real-world DevOps practices, emphasizing the importance of teamwork, communication, and shared responsibility in software development and operations.

- **Advanced Topics and Emerging Trends:** The course will introduce advanced topics such as microservices architecture, serverless computing, and DevSecOps. Students will also stay updated on emerging trends in DevOps and automation, preparing them for future developments in the field.
- **Security Integration:** Emphasis will be placed on integrating security practices within the DevOps pipeline, ensuring that students are equipped to build secure, compliant, and resilient systems.
- **Self-Study and Peer Review:** Encourage students to explore additional DevOps practices and trends through independent research, while organizing peer review sessions to provide feedback on each other's projects and automation workflows.

Text Books

- "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation" by Jez Humble and David Farley, Pearson Education, Inc., 2011

Reference Books

- "The Kubernetes Book" by Nigel Poulton, Independently published, 2018
- "Building Microservices: Designing Fine-Grained Systems" by Sam Newman, O'Reilly Media, Inc., 2015
- "Microservices Patterns: With examples in Java" by Eberhard Wolff, Manning Publications, 2018
- "Terraform: Up & Running: Writing Infrastructure as Code" by Yevgeniy Brikman, O'Reilly Media, Inc., 2017

Additional Readings

Self-Learning Components:

1. Kubernetes Academy by VMware
   Description: Free courses provided by VMware on Kubernetes, covering everything from basic concepts to advanced orchestration techniques.
   Link: https://kubernetes.academy
2. HashiCorp Learn: Terraform
   Description: HashiCorp's official resource for learning Terraform, providing tutorials and hands-on labs for infrastructure as code.
   Link: https://learn.hashicorp.com/terraform
3. Docker: Docker for Developers
   Description: Docker's official training resources for developers, covering containerization, Docker Compose, and more.
   Link: https://www.docker.com/docker-developer

DevOps & Automation Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
DevOps & Automation Lab	ENSP461	0-0-2	1
Type of Course:	Minor		

Defined Course Outcomes

COs	Statements
CO 1	Implementing collaborative development and continuous integration using Git and Jenkins, demonstrating proficiency in version control, automated testing, and deployment processes.
CO 2	Developing and deploying microservices applications using Docker for containerization and Kubernetes for orchestration, managing multi-container applications efficiently.
CO 3	Managing automated infrastructure provisioning and configuration using Ansible and Terraform, demonstrating expertise in infrastructure as code and configuration management.
CO 4	Implementing continuous monitoring, logging, and security best practices in a DevOps environment, ensuring application performance, system health, and data integrity.

S.N	Experiment	Mapped CO/COs
1	Set up a Git repository and practice branching, merging, and collaborative development. Create a small project and manage code versions using Git.	CO1
2	Install and configure Jenkins for continuous integration. Create a simple CI pipeline that automatically builds and tests a project whenever code changes are committed to the repository.	CO1
3	Implement a continuous delivery pipeline using Jenkins. Deploy a sample application to a staging environment automatically after successful builds and tests.	CO1
4	Implement different Git workflows (e.g., GitFlow, Feature Branch Workflow) for a collaborative project. Manage branches, merges, and resolve conflicts.	CO1
5	Set up a Jenkins pipeline for continuous integration. Configure automated testing and deployment for a sample project. Integrate with a version control system like Git.	CO1

S.N	Experiment	Mapped CO/COs
6	Install Docker and create Dockerfiles for a sample application. Build, run, and manage containers using Docker commands.	CO2
7	Use Docker Compose to manage multi-container applications. Create a Docker Compose file to run a web application with a database and other services.	CO2
8	Use Terraform to provision and manage cloud infrastructure. Create Terraform scripts to deploy a web application on a cloud provider (e.g., AWS, Azure).	CO3
9	Set up monitoring and logging for a sample application. Use tools like Prometheus, Grafana, and ELK Stack (Elasticsearch, Logstash, Kibana) to monitor and analyze application performance and logs.	CO4

Online Learning Resources

- **Git Documentation:** Comprehensive guide on using Git for version control.
  https://git-scm.com/doc
- **Jenkins Documentation:** Resources for learning and using Jenkins for CI/CD.
  https://www.jenkins.io/doc/
- **Docker Documentation:** Guide on using Docker for containerization.
  https://docs.docker.com/
- **Terraform Documentation:** Guide on using Terraform for infrastructure as code.
  https://www.terraform.io/docs/
- **Prometheus Documentation:** Resources for learning and using Prometheus for monitoring.
  https://prometheus.io/docs/introduction/overview/
- **ELK Stack Documentation:** Guide on using the ELK Stack for logging and monitoring.
  https://www.elastic.co/what-is/elk-stack

.NET Framework

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
.NET Framework	ENSP413	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	None		

Course Perspective: The ".NET Framework" syllabus covers introduction and components of .NET, programming languages, Visual Studio, OOP, exception handling, memory management, Windows Forms/WPF, ASP.NET, web services, .NET Core, Entity Framework, and WCF. Emphasis on practical application and development skills for building robust and secure applications. The course is divided into 4 modules:

1. .NET Framework
2. .NET Framework Fundamentals
3. Building Applications with .NET Framework
4. ASP.NET Framework

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding .NET Framework's architecture, CLR, and CTS for cross-language integration and platform independence.
CO 2	Applying OOP concepts in .NET for designing robust software solutions.
CO 3	Utilizing Visual Studio debugging for diagnosing and fixing errors in .NET applications.
CO 4	Demonstrating proficiency in memory management and garbage collection in .NET.
CO 5	Designing web applications using ASP.NET, incorporating best practices.

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

Course Outline

Unit Number: 1	Title: .NET Framework	No. of hours: 8
Content:		
• .NET Framework - Architecture, Common Language Runtime, Common Type System, Namespaces, Assemblies, Memory Management, Process Management, Class Libraries		
Unit Number: 2	**Title: .NET Framework Fundamentals**	**No. of hours: 8**
Content:		
• Object-Oriented Programming (OOP) in .NET, Classes, objects, and inheritance • Exception Handling and Debugging, Debugging techniques and tools in Visual Studio, Logging and error reporting in .NET applications • Memory Management and Garbage Collection, Automatic memory management in .NET, Garbage collection, Finalizers and the Dispose pattern		
Unit Number: 3	**Title: Building Applications with .NET Framework**	**No. of hours: 12**
Content:		
• .NET - Declaration, Expression, Control Structures, Function, String, Array, Encapsulation, Class, Property, Indexer, Delegate, Inheritance, Interface, Polymorphism, Exception Handling, Modules, Graphics, File handling and Data Access. • .NET – Form- Event–Form Controls – Containers – Menus - Data controls - Printing – Reporting – Dialogs – Components - Single and Multiple Document Interfaces.		
Unit Number: 4	**Title: ASP.NET Framework**	**No. of hours: 12**
Content:		
• ASP.NET – Web Pages, Web Forms, Web Site Design, Data Controls, Validation Controls, HTML, Navigation Controls, Login Controls, Reports - Master Pages – Web Service Architecture - Basic Web Services – Web Reference – Standards		

Learning Experience

Classroom Learning Experience

- **Comprehensive Understanding of .NET Architecture:** Students will develop a deep understanding of the .NET Framework, including its architecture, components, and functionalities such as the Common Language Runtime (CLR)

and the Common Type System (CTS). This knowledge is essential for building cross-platform and language-integrated applications.

- **Hands-On OOP in .NET:** Through practical exercises, students will apply Object-Oriented Programming (OOP) concepts within the .NET environment. They will design and implement classes, objects, inheritance, and polymorphism, which are crucial for creating robust and maintainable software solutions.
- **Debugging and Exception Handling:** The course emphasizes the importance of debugging and error handling in software development. Students will gain hands-on experience with Visual Studio's debugging tools and learn how to manage exceptions effectively, ensuring the reliability of their applications.
- **Memory Management Proficiency:** Understanding and implementing memory management techniques in .NET is a critical skill. Students will learn about garbage collection, finalizers, and the Dispose pattern, which are essential for optimizing application performance and resource management.
- **Continuous Feedback and Iterative Learning:** Regular assessments, peer reviews, and instructor feedback will help students refine their skills and knowledge. This iterative learning process ensures that students can continuously improve and adapt to the complexities of .NET development.

Outside Classroom Learning Experience

- **Application Development with Visual Studio:** The course includes extensive hands-on sessions where students will build both desktop and web applications using Visual Studio. They will learn to use various controls, event handling, data access methods, and form designs to create fully functional applications.
- **Web Development with ASP.NET:** Students will explore the ASP.NET framework, gaining the ability to design and develop dynamic web pages and services. They will work with web forms, data controls, validation controls, and master pages to build secure and scalable web applications.
- **Real-World Project Development:** The course culminates in a capstone project where students will apply all the concepts learned to develop a complete .NET application. This project-based learning approach ensures that students are industry-ready and can confidently develop enterprise-level applications.
- **Exploration of Advanced .NET Features:** Students will explore advanced features of the .NET Framework, such as working with Windows Presentation Foundation (WPF) and Windows Communication Foundation (WCF). These features allow for the creation of rich user interfaces and reliable communication services.
- **Self-Study and Research:** Encourage students to independently explore additional .NET topics such as asynchronous programming, LINQ, and secure coding practices to deepen their knowledge of the framework.

Text Books

- "Pro C# 8 with .NET Core: Foundational Principles and Practices in Programming" by Andrew Troelsen and Philip Japikse, Apress, 9th Edition, 2020

- "Pro ASP.NET Core 3" by Adam Freeman, Apress
- "ASP.NET Core in Action" by Andrew Lock

Reference Books

- "Pro C# 8 with .NET Core: Foundational Principles and Practices in Programming" by Andrew Troelsen and Philip Japikse, Apress, 9th Edition, 2020
- "Pro ASP.NET Core 3" by Adam Freeman, Apress
- "ASP.NET Core in Action" by Andrew Lock

Additional Readings

Self-Learning Components:

1. Microsoft .NET Documentation
   Description: Direct students to the official Microsoft documentation for .NET Framework, which provides comprehensive guides and resources.
   Link: https://docs.microsoft.com/en-us/dotnet/

2. LeetCode
   Description: Assign coding exercises from platforms like LeetCode that focus on implementing concepts of .NET Framework.
   Link: https://leetcode.com/

3. HackerRank
   Description: Assign coding exercises from platforms like HackerRank that focus on implementing concepts of .NET Framework.
   Link: https://www.hackerrank.com/

4. GitHub
   Description: Encourage students to work on small projects using different aspects of the .NET Framework. Provide examples of project ideas and resources like GitHub repositories for inspiration.
   Link: https://github.com/

.NET Framework Lab

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
.NET Framework Lab	ENSP463	0-0-2	1
Type of Course:	Minor		

Defined Course Outcomes

COs	Statements
CO 1	Understanding and applying object-oriented design principles, exception handling, memory management, and debugging techniques to develop robust .NET applications.
CO 2	Developing graphical user interfaces and handle events in .NET applications to create interactive and user-friendly software solutions.
CO 3	Implementing web development techniques in ASP.NET, including web forms, user authentication, master pages, and web services to build secure and dynamic web applications.
CO 4	Analyzing and utilizing data handling, reporting, and visualization techniques to create comprehensive and functional software systems for various domains.

S.N	Experiment	Mapped CO/COs
1	Explore the architecture of the .NET Framework. Create a simple console application to understand the basic structure and components of a .NET project.	CO1
2	Demonstrate the functionality of the Common Language Runtime (CLR). Create a .NET application that uses various data types and namespaces to show how the CLR manages execution.	CO1
3	Implement a .NET application that showcases the Common Type System (CTS). Define and use various data types, and demonstrate type conversion and interoperability.	CO1
4	Create and manage assemblies in a .NET application. Demonstrate how to build, reference, and use assemblies in a multi-project solution.	CO1
5	Implement a simple object-oriented application in .NET. Define classes, create objects, and demonstrate inheritance and polymorphism.	CO2
6	Implement a .NET application that logs errors and handles exceptions gracefully. Use a logging framework (e.g., NLog, log4net) to record application events and errors.	CO2

S.N	Experiment	Mapped CO/COs
7	Build a .NET application demonstrating advanced OOP concepts such as encapsulation, properties, indexers, delegates, interfaces, and polymorphism.	CO3
8	Create a .NET application that handles graphics and file I/O. Implement functionality to draw shapes, handle images, and perform file read/write operations.	CO3
9	Implement a .NET application with a rich user interface. Use forms, event handling, form controls, containers, menus, data controls, printing, and reporting functionalities to create a feature-rich application.	CO3
10	Create a basic ASP.NET web application. Design web pages and web forms to understand the structure and components of an ASP.NET project.	CO4
11	Develop an ASP.NET application with user authentication. Use login controls to implement user authentication and authorization, and create a simple reporting feature to display user data.	CO4
12	Create and consume a basic web service in ASP.NET. Implement a web service that provides data to a client application, and demonstrate how to use web references to integrate the web service with an ASP.NET project.	CO4

Online Learning Resources

- **Microsoft .NET Documentation:** Comprehensive guide on using .NET for application development.
  https://docs.microsoft.com/en-us/dotnet/
- **ASP.NET Documentation:** Resources for learning and using ASP.NET for web development.
  https://docs.microsoft.com/en-us/aspnet/core/
- **Microsoft Learn:** Interactive tutorials and learning paths for .NET and ASP.NET.
  https://docs.microsoft.com/en-us/learn/dotnet/
- **Pluralsight:** Training and certification courses for .NET and ASP.NET development.
  https://www.pluralsight.com/paths/dotnet

New Age Programming Languages

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
New Age Programming Languages	ENSP415	4-0-0	4
Type of Course:	Minor		
Pre-requisite(s):	None		

Course Perspective: New-Age programming languages (GO, F#, Clojure, Kotlin) provide an introduction to the concepts and applications of modern programming languages. It explores the features and benefits of GO, F#, Clojure, and Kotlin, and develops practical skills in programming using these languages. The course will cover language syntax, data types, control structures, functional programming concepts, concurrency, and integration with other technologies. The course is divided into 4 modules:

1. GO Programming Language
2. F# Programming Language
3. Clojure Programming Language
4. Kotlin Programming Language

The Course Outcomes (COs)

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

COs	**Statements**
CO 1	Understanding principles and paradigms of modern programming languages.
CO 2	Developing proficiency in syntax, data structures, and control flow of each language.
CO 3	Exploring unique features and strengths of each language.
CO 4	Applying development tools to improve code quality and productivity.
CO 5	Designing and implementing projects integrating multiple programming languages.

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

Course Outline

Unit Number: 1	Title: GO Programming Language	No. of hours: 10
Content:		

- Overview and Comparison: Overview of GO, F#, Clojure, and Kotlin, Comparison with traditional programming languages, Installation and setup of development environment
- GO Programming Basics: Introduction to GO syntax and data types, Control structures in GO, Functions and packages, Arrays, slices, and maps, Structs and custom data types, Pointers and memory management

Unit Number: 2	Title: F# Programming Language	No. of hours: 10
Content:		

- Introduction to F# syntax and functional programming concepts, Data Types, Variables, Operators, Decision Making, Loops, Functions, Strings, Options, Immutable data types and pattern matching
- Higher-order functions and currying, Asynchronous and parallel programming in F#, Object-Oriented Programming with F#
- Database access with F#, Querying and manipulating data using F#, Integration with relational and NoSQL databases

Unit Number: 3	Title: Introduction to Clojure Programming	No. of hours: 10
Content:		

- Introduction to Clojure: Overview of Clojure and its features, Setting up the development environment
- Basic Syntax and Functional Programming, Basic syntax and data structures, Functional programming concepts, Immutable data and pure functions, Higher-order functions and recursion, Collections and sequence operations, Restructuring and pattern matching
- Error Handling and Testing: Exception handling and error management in Clojure, Testing strategies and frameworks in Clojure
- Data Manipulation and Transformation: Data manipulation with Clojure's sequence functions, Data transformation with transducers, Data-driven development with data literals and data readers

Unit Number: 4	Title: Introduction to Kotlin Programming	No. of hours: 10
Content:		

- Overview of Kotlin and its advantages, Setting up the development environment, Basic syntax and data types in Kotlin, Conditional statements and loops, Function declarations and parameters, Lambda expressions and higher-order functions
- Object-Oriented Programming in Kotlin: Classes, objects, and inheritance, Properties and access modifiers, Interfaces and abstract classes, Understanding nullable and non-nullable types, Safe calls and the Elvis operator, Type inference and smart casting
- Collections and Functional Programming: Working with lists, sets, and maps in Kotlin, Collection operations and transformations, Introduction to functional programming concepts in Kotlin, Creating extension functions in Kotlin, Using DSLs for domain-specific problems, Builder pattern and DSL implementation

Learning Experience

Classroom Learning Experience

- **Exploration of Modern Programming Paradigms:** Students will gain exposure to a diverse range of programming paradigms, including functional, concurrent, and object-oriented programming, by exploring languages such as GO, F#, Clojure, and Kotlin. This broadens their understanding of different approaches to software development.
- **Hands-On Coding with New-Age Languages:** Through practical assignments, students will become proficient in the syntax, control structures, and unique features of each language. They will build applications that demonstrate their ability to write clean, efficient, and effective code using GO, F#, Clojure, and Kotlin.
- **Functional Programming Proficiency:** The course places a strong emphasis on functional programming, particularly in F# and Clojure. Students will learn to implement functional constructs such as higher-order functions, immutability, and recursion, which are essential for developing scalable and maintainable code.
- **Concurrency and Parallelism in GO and F#:** Students will explore the concurrency models in GO and F#, gaining hands-on experience in writing concurrent applications. This includes understanding goroutines in GO and asynchronous workflows in F#, preparing them to tackle performance-critical tasks in modern software systems.
- **Integration with Modern Technologies:** The course will teach students how to integrate these modern programming languages with contemporary tools and technologies, such as databases, web frameworks, and cloud services. This ensures that students are equipped to use these languages in real-world development environments.

Outside Classroom Learning Experience

- **Project-Based Learning:** Students will apply their learning by developing projects that integrate multiple programming languages. These projects will demonstrate

their ability to select the appropriate language for a given task and to combine the strengths of various languages in a single application.

- **Advanced Programming Techniques:** The course will introduce advanced topics such as metaprogramming in Clojure, domain-specific languages (DSLs) in Kotlin, and memory management in GO. Students will gain insights into how these techniques can be used to optimize and extend the capabilities of their applications.
- **Continuous Feedback and Collaboration:** The learning experience will be enhanced by regular peer reviews, code walkthroughs, and instructor feedback. This collaborative environment will encourage students to refine their skills and learn from their peers, fostering a deeper understanding of the course material.
- **Development of a Language-Agnostic Perspective:** By studying multiple languages, students will develop a language-agnostic mindset, allowing them to adapt quickly to new programming languages and paradigms in the future. This adaptability is crucial in the ever-evolving field of software development.
- **Self-Study and Research:** Encourage students to independently explore additional language-specific features, libraries, and best practices, enhancing their understanding of each language's unique capabilities and applications.

Text Books

- "The Go Programming Language" by Alan A. A. Donovan and Brian W. Kernighan, Addison-Wesley Professional.
- "An Introduction to Programming in Go" by Caleb Doxsey, CreateSpace Independent Publishing.

Reference Books

- "Real-World Functional Programming: With Examples in F# and C#" by Tomas Petricek and Jon Skeet, Manning.
- "Programming F# 3.0: A Comprehensive Guide for Writing Simple Code to Solve Complex Problems" by Chris Smith, O'Reilly Media.
- "Getting Clojure: Build Your Functional Skills One Idea at a Time" by Russ Olsen, O'Reilly.
- "The Joy of Clojure" by Michael Fogus and Chris Houser, Manning Publication.
- "Atomic Kotlin" by Bruce Eckel and Svetlana Isakova, Mindview LLC.
- "Kotlin in Action" by Dmitry Jemerov and Svetlana Isakova, Manning Publication.

Additional Readings

Self-Learning Components:

1. Go (Golang)

(a) Coursera: Programming with Google Go
Description: An introductory course to Go programming, covering language syntax, data structures, and more.
Link: https://www.coursera.org/learn/golang-programming

(b) Go by Example
Description: A hands-on introduction to Go using annotated example programs.
Link: https://gobyexample.com/

2. F#

(a) Microsoft Learn: Introduction to F#
Description: A series of modules introducing the F# language, its syntax, and functional programming concepts.
Link: https://docs.microsoft.com/en-us/learn/modules/fsharp-introduction/

3. Clojure

(a) ClojureBridge
Description: Free Clojure workshops for beginners, including resources and exercises.
Link: https://clojurebridge.org/

(b) Learn Clojure: Clojure for the Brave and True
Description: A beginner-friendly book that teaches Clojure through real-world projects and examples.
Link: https://www.braveclojure.com/clojure-for-the-brave-and-true/

4. Kotlin

(a) Kotlin Lang: Kotlin Documentation
Description: Official Kotlin documentation and tutorials by JetBrains.
Link: https://kotlinlang.org/docs/reference/

(b) Udacity: Kotlin for Android Developers
Description: A course by Udacity focusing on Kotlin for Android development.
Link: https://www.udacity.com/course/kotlin-for-android-developers--ud888

Semester: 6

Major Project/Indusrial Training/Startup

Program Name:	B.Sc (Hons.) Cyber Security		
Course Name:	**Course Code**	**L-T-P**	**Credits**
Major Project/Indusrial Training/Startup	SIBC352	0-0-0	12
Type of Course:	Proj		
Pre-requisite(s):	None		

Preface

The **B.Sc (Hons.) Cyber Security Final Semester Major Project/Industrial Training/Startup** is a culmination of the academic journey for students in this program. This Standard Operating Procedure (SOP) is designed to provide guidance, ensuring a comprehensive, practical, and outcome-driven approach. The SOP provides a framework for students to choose from three types of projects—Industrial Projects, Research & Development (R&D) Projects, and Start-up Projects—emphasizing experiential learning, problem-solving, and interdisciplinary collaboration. Students will be mentored by both internal faculty and external industry or academic experts.

1. Introduction

The **B.Sc (Hons.) Cyber Security Final Semester Major Project/Industrial Training/Startup** is an essential academic requirement that enables students to apply theoretical knowledge to practical challenges. This project fosters critical thinking, problem-solving, innovation, and research-oriented learning with a focus on real-world problems in industrial, research, and entrepreneurial domains. Students may choose from:

- **Industrial Project:** Solving real industrial problems in collaboration with an industry partner.
- **Research & Development (R&D) Project:** Contributing to academic and applied research, with external guidance from academic or research institutions.
- **Start-up Project:** Developing and launching innovative start-up ideas with entrepreneurial mentors.

The SOP ensures that these projects, emphasizing interdisciplinary, practical, and outcome-based learning.

2. Objectives

The primary objectives of the full-time project are:

- **Application of Theoretical Knowledge:** Enable students to apply their academic learning to practical problems.
- **Holistic Development:** Promote interdisciplinary learning, critical thinking, creativity, and problem-solving.
- **Research and Innovation:** Encourage innovative solutions, leading to publications, patents, or prototypes.
- **Industry Collaboration:** Foster partnerships with industries for real-world problem-solving.
- **Entrepreneurship Development:** Develop entrepreneurial skills and create viable start-ups.
- **Global Competency:** Equip students with the skills required to excel in global environments through research, innovation, and collaboration.

3. Types of Projects

a) Industrial Project

Students working on Industrial Projects will:

- Collaborate with an industry partner.
- Identify specific, real-world challenges faced by the company.
- Propose and implement a solution that provides value to the industry.
- Develop a final product or prototype that can be implemented in the industrial setting.

Project Proposal:

- **Problem Statement and Objectives:** Identify the industrial problem and outline the objectives.
- **Proposed Solution:** Present a detailed methodology for solving the problem.
- **Deliverables:** Define tangible deliverables, including prototypes, software, or hardware.
- **Expected Impact:** Outline the expected impact on the industry.

Evaluation Criteria:

- Practical implementation and solution viability (40%)
- Project innovation (20%)
- Industrial applicability and impact (20%)
- Final presentation and report quality (20%)

b) Research & Development (R&D) Project

The R&D Project focuses on creating innovative research outcomes through collaborations with academic or research institutions. This can result in publications, research reports, or new discoveries.

Project Proposal:

- **Literature Review:** Detailed research on existing work related to the chosen topic.
- **Hypothesis/Research Questions:** Define the specific research problem or question.
- **Methodology:** Include data collection, experimental design, and analysis techniques.
- **Research Timeline:** Step-by-step phases of research with milestones.

Mentorship:

- **External Mentor:** Collaboration with an external academic expert is mandatory for research projects.
- **Internal Mentor:** Each student will also be assigned an internal faculty member who will supervise the project.

Evaluation Criteria:

- Quality of Research and Novelty (30%)
- Research Methodology (25%)
- Contributions to the field (20%)
- Final Report, Presentation, and Publication (25%)

c) Start-up Project

The Start-up Project involves developing a business model or creating a start-up venture. Students work on a product/service idea that addresses a significant market need or societal problem.

Project Proposal:

- **Start-up Idea:** Explain the business or product idea.
- **Market Research:** Detailed research on the market, target customers, competitors, and potential revenue streams.
- **Business Plan:** Define the steps needed to take the idea to market, including funding, development phases, marketing, and operational plans.
- **Product Prototype:** If applicable, develop a working prototype.

Mentorship:

- **External Mentor:** An industry/start-up expert will guide the student.

- **Internal Faculty Mentor:** An internal mentor will provide academic guidance and ensure the start-up idea is feasible and innovative.

Evaluation Criteria:

- Start-up viability and market potential (30%)
- Product or service innovation (30%)
- Prototype/Business Model Development (20%)
- Final Pitch/Presentation and Start-up Plan (20%)

4. Roles and Responsibilities

a) Student's Responsibilities

- Select a suitable project topic based on interests (industrial, R&D, or start-up).
- Draft and submit a detailed proposal with objectives, methodology, timelines, and deliverables.
- Coordinate with both external and internal mentors regularly for feedback and guidance.
- Maintain a weekly progress report for both mentors.
- Submit a final comprehensive report and present the project.

b) Internal Supervisor

- Guide the student throughout the project.
- Provide academic input and ensure the project aligns with program outcomes.
- Conduct progress reviews and ensure timelines are adhered to.
- Evaluate the project at the mid-term and final stages.

c) External Mentor

- Offer specialized industrial, research, or entrepreneurial guidance.
- Provide real-world problem insights for industrial and start-up projects.
- Ensure the project is relevant to the chosen industry, research domain, or start-up ecosystem.
- Participate in the final evaluation of the project.

5. Project Phases

Phase 1: Proposal Submission and Approval

- Students must submit a project proposal during the first two weeks of the final semester.
- The proposal must include the problem statement, objectives, literature review (for R&D projects), methodology, and expected outcomes.
- The proposal is subject to review and approval by the internal supervisor and external mentor.

Phase 2: Planning and Resource Allocation

- Once approved, the student will develop a project plan that includes:
  - **Project Milestones:** Break down the project into smaller tasks with defined milestones.
  - **Resource Requirements:** Identify any software, hardware, lab resources, or tools required for the project.
  - **Team Roles:** For group projects, define the roles of each team member.
  - **Risk Assessment:** Highlight potential risks and corresponding mitigation strategies.

Phase 3: Mid-term Review

- A mid-term review will be conducted halfway through the project to assess progress.
- Students will present their work to a committee consisting of the internal supervisor, external mentor, and department head.
- The review will assess the progress against the timeline and suggest course corrections if needed.

Phase 4: Final Execution and Evaluation

- **Industrial Projects:** Students must submit a prototype or industrial report.
- **R&D Projects:** Students must submit a final research report or publish findings.
- **Start-up Projects:** Students must present a business plan along with a prototype, market analysis, and revenue model.

Phase 5: Final Report Submission and Presentation

- **Final Report:** The project report should contain a title page, abstract, introduction, problem statement, objectives, methodology, results, discussion, conclusions, future scope, references, and appendices.
- **Presentation:** Students will deliver a final presentation to a panel of evaluators, showcasing their work, findings, or product.

6. Collaboration and Mentorship

For Research Projects, mentorship will involve both:

- **External Mentor:** An academic expert outside the institution, preferably from a reputed university or research institute.
- **Internal Mentor:** A faculty member from the student's department to provide academic and administrative guidance.

For Industrial Projects:

- External mentorship will come from industry professionals, preferably from the partnering company.

For Start-up Projects:

- External mentorship will involve experienced entrepreneurs, start-up founders, or investors.

7. Documentation and Submission Requirements

Students are required to:

- Submit their proposal, mid-term report, final report, and any supporting documents via the Learning Management System (LMS).
- Maintain detailed project logs and weekly reports.