University of Mumbai

वेबसाईट – mu.ac.in
ईमेल - आयडी - dr.aams@fort.mu.ac.in
aams3@mu.ac.in

विद्याविषयक प्राधिकरणे
सभा आणि सेवा विभाग(ए.ए.एम.एस)
रूम नं. १२८ एम.जी.रोड, फोर्ट,
मुंबई – ४०० ०३२
टेलिफोन नं - ०२२ - ६८३२००३३

(नॅक पुनर्मूल्यांकनाद्वारे ३.६५ (सी.जी.पी.ए.) सह अ++ श्रेणी
विद्यापीठ अनुदान आयोगाद्वारे श्रेणी १ विद्यापीठ दर्जा)

क.वि.प्रा.स.से./आयसीडी/२०२५-२६/३७ दिनांक : २७ मे, २०२५

परिपत्रक:-

सर्व प्राचार्य/संचालक, संलग्नित महाविद्यालये/संस्था, विद्यापीठ शैक्षणिक विभागांचे संचालक/ विभाग प्रमुख यांना कळविण्यात येते की, राष्ट्रीय शैक्षणिक धोरण २०२० च्या अमंलबजावणीच्या अनुषंगाने **शैक्षणिक वर्ष २०२५-२६** पासून **पदवी व पदव्युत्तर अभ्यासक्रम** विद्यापरिषदेच्या दिनांक २८ मार्च २०२५ व २० मे, २०२५ च्या बैठकीमध्ये मंजूर झालेले सर्व अभ्यासक्रम मुंबई विद्यापीठाच्या www.mu.ac.in या संकेत स्थळावर NEP २०२० या टॅब वर उपलब्ध करण्यात आलेले आहेत.

[illegible]
(डॉ. प्रसाद कारंडे)
कुलसचिव

मुंबई – ४०० ०३२
२७ मे, २०२५

क.वि.प्रा.स.से.वि/आयसीडी/२०२५-२६/३७ दिनांक : २७ मे, २०२५
Desktop/ Pritam Loke/Marathi Circular/NEP Tab Circular

Copy forwarded for information and necessary action to :-	
1	The Deputy Registrar, (Admissions, Enrolment, Eligibility and Migration Dept)(AEM), dr@eligi.mu.ac.in
2	The Deputy Registrar, Result unit, Vidyanagari drresults@exam.mu.ac.in
3	The Deputy Registrar, Marks and Certificate Unit,. Vidyanagari dr.verification@mu.ac.in
4	The Deputy Registrar, Appointment Unit, Vidyanagari dr.appointment@exam.mu.ac.in
5	The Deputy Registrar, CAP Unit, Vidyanagari cap.exam@mu.ac.in
6	The Deputy Registrar, College Affiliations & Development Department (CAD), deputyregistrar.uni@gmail.com
7	The Deputy Registrar, PRO, Fort, (Publication Section), Pro@mu.ac.in
8	The Deputy Registrar, Executive Authorities Section (EA) eau120@fort.mu.ac.in He is requested to treat this as action taken report on the concerned resolution adopted by the Academic Council referred to the above circular.
9	The Deputy Registrar, Research Administration & Promotion Cell (RAPC), rapc@mu.ac.in
10	The Deputy Registrar, Academic Appointments & Quality Assurance (AAQA) dy.registrar.tau.fort.mu.ac.in ar.tau@fort.mu.ac.in
11	The Deputy Registrar, College Teachers Approval Unit (CTA), concolsection@gmail.com
12	The Deputy Registrars, Finance & Accounts Section, fort draccounts@fort.mu.ac.in
13	The Deputy Registrar, Election Section, Fort drelection@election.mu.ac.in
14	The Assistant Registrar, Administrative Sub-Campus Thane, thanesubcampus@mu.ac.in
15	The Assistant Registrar, School of Engg. & Applied Sciences, Kalyan, ar.seask@mu.ac.in
16	The Assistant Registrar, Ratnagiri Sub-centre, Ratnagiri, ratnagirisubcentar@gmail.com
17	The Director, Centre for Distance and Online Education (CDOE), Vidyanagari, director@idol.mu.ac.in
18	Director, Innovation, Incubation and Linkages, Dr. Sachin Laddha pinkumanno@gmail.com
19	Director, Department of Lifelong Learning and Extension (DLLE), dlleuniversityofmumbai@gmail.com

Copy for information :-	
1	P.A to Hon’ble Vice-Chancellor, vice-chancellor@mu.ac.in
2	P.A to Pro-Vice-Chancellor pvc@fort.mu.ac.in
3	P.A to Registrar, registrar@fort.mu.ac.in
4	P.A to all Deans of all Faculties
5	P.A to Finance & Account Officers, (F & A.O), camu@accounts.mu.ac.in

To,

1	The Chairman, Board of Deans pvc@fort.mu.ac.in
2	**Faculty of Humanities,** **Offg. Dean** 1. Prof.Anil Singh Dranilsingh129@gmail.com **Offg. Associate Dean** 2. Prof.Manisha Karne mkarne@economics.mu.ac.in 3. Dr.Suchitra Naik Naiksuchitra27@gmail.com
	Faculty of Commerce & Management, **Offg. Dean,** 1 Prin.Ravindra Bambardekar principal@model-college.edu.in **Offg. Associate Dean** 2. Dr.Kavita Laghate kavitalaghate@jbims.mu.ac.in 3. Dr.Ravikant Balkrishna Sangurde Ravikant.s.@somaiya.edu 4. Prin.Kishori Bhagat kishoribhagat@rediffmail.com

	Faculty of Science & Technology **Offg. Dean** 1. Prof. Shivram Garje ssgarje@chem.mu.ac.in **Offg. Associate Dean** 2. Dr. Madhav R. Rajwade Madhavr64@gmail.com 3. Prin. Deven Shah sir.deven@gmail.com
	Faculty of Inter-Disciplinary Studies, **Offg. Dean** 1.Dr. Anil K. Singh aksingh@trcl.org.in **Offg. Associate Dean** 2.Prin.Chadrashekhar Ashok Chakradeo cachakradeo@gmail.com 3. Dr. Kunal Ingle drkunalingle@gmail.com
3	Chairman, Board of Studies,
4	The Director, Board of Examinations and Evaluation, dboee@exam.mu.ac.in
5	The Director, Board of Students Development, dsd@mu.ac.in DSW direcotr@dsw.mu.ac.in
6	The Director, Department of Information & Communication Technology, director.dict@mu.ac.in

AC – 20/05/2025
Item No. – 6.14 (N)

As Per NEP 2020

University of Mumbai

Syllabus for Major Vertical – 1 & 4		
Name of the Programme – B.Sc. (Computer Science)		
Faulty of Science		
Board of Studies in Computer Science		
U.G. Second Year Programme	**Exit Degree**	**U.G. Diploma in Computer Science**
Semester		**III & IV**
From the Academic Year		**2025-26**

University of Mumbai

(As per NEP 2020)

Sr. No.	Heading	Particulars
1	**Title of program** **O: _____________**	**B.Sc. (Computer Science)**
2	**Exit Degree**	**U.G. Diploma in Computer Science**
3	**Scheme of Examination** **R: ______________**	NEP **40% Internal** **60% External**, Semester End Examination **Individual Passing** in Internal and External Examination
4	**Standards of Passing** **R: ______________**	**40%** in **each component**
5	**Credit Structure** **Sem. III - R: SU-555C** **Sem. IV -R: SU- 555D**	**Attached herewith**
6	**Semesters**	Sem III & IV
7	**Program Academic Level**	5.0
8	**Pattern**	Semester
9	**Status**	New
10	**To be implemented from Academic Year**	From Academic Year: 2025-26

Sd/-
Sign of the BOS Chairman
Dr. Jyotshna Dongardive
Ad-hoc BOS (Computer Science)

Sd/-
Sign of the Offg. Associate Dean
Dr. Madhav R. Rajwade
Faculty of Science & Technology

Sd/-
Sign of Offg. Dean
Prof. Shivram S. Garje
Faculty of Science & Technology

Preamble

1) Introduction

The **Second Year** of the **B.Sc. Computer Science** program serves as a pivotal phase in the academic journey of students, deepening their understanding of both theoretical and applied aspects of computing. This stage is structured to strengthen their analytical thinking and programming abilities while introducing them to essential components of system-level programming, computational theory, and software development life cycles.

Core subjects such as **Data Structures, Operating Systems**, and the **Theory of Computation** form the backbone of this year, equipping students with skills to design efficient algorithms, understand system architecture, and explore the mathematical foundations of computing. Simultaneously, practical programming knowledge is enhanced through courses like **Java Programming**, which emphasizes object-oriented programming and GUI development, and **Computer Networks**, which provides an insight into data transmission, network protocols, and communication models.

An important aspect of this year is the inclusion of **Software Engineering** and **IoT Technologies**, which bridge theory with real-world application by guiding students through the structured development of software and embedded systems. Further, skill-oriented courses like **Mobile Application Development** and **MEAN Stack Development** expose students to industry-relevant tools and frameworks, fostering full-stack development and innovation.

Overall, this phase of the program is designed to shape students into competent problem solvers, capable of building scalable software solutions and understanding the internal workings of computing systems. It also encourages critical thinking, creativity, and a readiness to explore emerging technologies, laying a solid foundation for their final year of specialization and future career paths.

2) Aims and Objectives

Deepen Conceptual Understanding: To develop a strong conceptual understanding of intermediate-level topics such as Operating Systems, Theory of Computation, Computer Networks, and Software Engineering.

Hands-on Skill Development: To provide hands-on experience through practical labs in Data Structures, Java Programming, Mobile App Development, and IoT, encouraging students to apply their learning to solve real-world problems.

Design & Analytical Skills: To strengthen algorithmic and analytical skills, enabling students to design efficient data structures, evaluate system performance, and understand formal models of computation.

Software Project Exposure: To introduce students to structured software development processes, including requirement analysis, design, testing, and documentation using Software Engineering principles.

Technology Familiarity: To familiarize students with industry-relevant technologies such as MEAN stack and IoT to encourage innovation and interdisciplinary application.

Teamwork & Communication: To promote collaborative project work and improve communication skills through documentation and presentations, essential for industry-readiness.

3) Learning Outcomes

By the end of the second year (S.Y.B.Sc.), students will be able to:

- Design and implement efficient data structures and apply them in software applications.
- Explain the fundamental principles of Operating Systems, including process scheduling, memory management, and file systems.
- Demonstrate a clear understanding of formal languages, automata theory, and computational models in the Theory of Computation.
- Build dynamic and interactive Java-based applications, including GUI elements and object-oriented designs.
- Understand and describe how computer networks function, including the OSI and TCP/IP models, and apply this knowledge to network-based applications.
- Analyze and design software systems using Software Engineering methodologies such as the SDLC, UML diagrams, and requirement specifications.
- Develop IoT-based prototypes, integrating hardware and software for smart environments.
- Build simple mobile applications using industry-standard tools and frameworks.
- Implement full-stack web applications using MEAN Stack Development tools.
- Work collaboratively on lab assignments and mini-projects, exhibiting professionalism, communication skills, and teamwork.
- Prepare themselves for higher studies and internships by integrating theory with hands-on experience across varied domains of Computer Science.Formulate, model, and design solutions and procedures, utilizing software tools to address real-world problems effectively.

Credit Structure of the Program
Under Graduate Diploma in Computer Science

Level	Semester	Major: Mandatory	Major: Electives	Minor	OE	VSC, SEC (VSEC)	AEC, VEC, IKS	OJT, FP, CEP, CC, RP	Cum. Cr. / Sem.	Degree/ Cum. Cr.
	R: SU-555 C									
5.0	III	**MJ5:** Principles of Operating Systems (TH) – 2 **MJ6:** Theory of Computation (TH) – 2 **MJ7:** Data Structures (TH) – 2 **MJP3:** Computer Science Practical 3 (PR) – 2 **8**	-	4	2	**VSC:2** Java Programming – 2	AEC:2	FP: 2 CC:2	22	**UG Diploma 88**
	R: SU-555 D									
	IV	**MJ8:** Computer Networks (TH) – 2 **MJ9:** Software Engineering (TH) – 2 **MJ10:** IoT Technologies (TH) – 2 **MJP4:** Computer Science Practical 4 (PR) – 2 **8**	-	4	2	**SEC:2** Mobile Application Development – 2 **OR** MEAN Stack Development – 2	AEC:2	CEP: 2 CC:2	22	
	Cum Cr.	**28**	**-**	**10**	**12**	**6+6**	**8+4+2**	**8+4**	**88**	
Exit option; Award of UG Diploma in Major and Minor with 80-88 credits and an additional 4 credits core NSQF course/ Internship OR Continuewith Major and Minor										

[Abbreviation - OE – Open Electives, VSC – Vocation Skill Course, SEC –Skill Enhancement Course, (VSEC), AEC – Ability Enhancement Course, VEC – Value Education Course, IKS – Indian Knowledge System, OJT – on Job Training, FP – Field Project, CEP – Community Engagement Project, CC – Co-Curricular, RP – Research Project]

Semester III

Component	Major		Minor	OE	VSC	SEC	AEC	FP	CC	Total
	Mandatory	Electives								
Credits	2+2+2+2	---	4	2	2	---	2	2	2	22

Component	Subject	Total Credits
Major	Principles of Operating Systems	**2**
Major	Theory of Computation	**2**
Major	Data Structures	**2**
Major	Computer Science Practical 3	**2**
VSC	Java Programming	**2**

Semester IV

Component	Major		Minor	OE	VSC	SEC	AEC	CEP	CC	Total
	Mandatory	Electives								
Credits	2+2+2+2	---	4	2	---	2	2	2	2	22

Component	Subject	Total Credits
Major	Computer Networks	**2**
Major	Software Engineering	**2**
Major	IoT Technologies	**2**
Major	Computer Science Practical 4	**2**
SEC (any one)	Mobile Application Development	**2**
	MEAN Stack Development	

Sem – III

Name of the Course: Principles of Operating Systems

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** The Principles of Operating Systems course introduces students to the fundamental role an operating system plays in managing hardware and software resources. It covers essential concepts like process management, memory handling, file systems, and CPU scheduling, providing a strong foundation in how computers function at a low level. **Relevance:** This course is highly relevant as it bridges the gap between computer hardware and user applications. Understanding the operating system's role helps students grasp how various programs run efficiently and how resources like memory and CPU are allocated and managed. **Usefulness:** The course equips students with practical knowledge that is directly applicable in configuring, using, and troubleshooting different operating systems such as Linux, Windows, and macOS. It also deepens their understanding of how applications interact with system resources, which is critical for developers, testers, and IT professionals. **Application:** Operating system principles are used extensively in fields such as embedded systems, mobile application development, cloud computing, and cybersecurity. From smartphones to servers and IoT devices, the knowledge of OS design and implementation is central to building and maintaining modern computing environments. **Interest:** This course excites students by uncovering the inner workings of the systems they use every day. The hands-on aspects, like working with commands, simulating scheduling algorithms, and exploring file structures, make it both intellectually engaging and practically rewarding. **Connection with Other Courses:** The concepts learned here support and are reinforced in

		several other subjects such as Computer Architecture, Data Structures and Algorithms, System Programming, and Cybersecurity. The interdisciplinary nature of the course strengthens students’ overall understanding of the computer science domain. **Demand in the Industry:** There is a consistent demand in the IT industry for professionals who understand the workings of operating systems. Tech companies developing system software, embedded platforms, or managing cloud infrastructure seek candidates who have a strong grasp of OS-level functioning. **Job Prospects:** Career opportunities after learning this subject include roles like system administrator, kernel developer, embedded systems engineer, and DevOps engineer. A deep knowledge of operating systems also opens doors to specialized fields such as cybersecurity, performance tuning, and system software development.
2	**Vertical:**	Major
3	**Type:**	Theory
4	**Credits:**	2 credits
5	**Hours Allotted:**	30 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives (CO):** **CO 1.** To learn basic concepts and structure of operating systems **CO 2.** To understand process communication techniques **CO 3.** To study various CPU scheduling algorithms **CO 4.** To learn about Memory management **CO 5.** To learn about File system management and implementation	
8	**Course Outcomes (OC):** After successful completion of this course, students would be able to - **OC 1.** Work with any type of operating system **OC 2.** Handle threads, processes, process synchronization **OC 3.** Implement CPU scheduling algorithms **OC 4.** Understand the background role of memory management **OC 5.** Design file system	
9	**Modules:-** **Module 1 (15 hours):**	
	Fundamentals of Operating systems – Definition of Operating System, Operating System's role, Operating-System Operations, Functions of Operating System, Computing Environments Operating-System Services, User and Operating-System Interface, System Calls, Types of System Calls, Operating-System Structure	

	Processes – Threads - Overview, Multicore Programming, Multithreading Models, Process Concept, Process Scheduling, Operations on Processes, Inter-process Communication **Process Synchronization** – General structure of a typical process, race condition, The Critical-Section Problem, Mutex Locks, Semaphores, Classic Problems of Synchronization, Monitors **CPU Scheduling** – Basic Concepts, Scheduling Criteria, Scheduling Algorithms (FCFS, SJF, SRTF, Priority, RR, Multilevel Queue Scheduling, Multilevel Feedback Queue Scheduling), Thread Scheduling	
	Module 2 (15 hours):	
	Deadlocks – System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock **Memory Management** – Main memory background, Logical address space, Physical address space, MMU, Swapping, Contiguous Memory Allocation, Segmentation, Paging, Structure of the Page Table **Virtual Memory** – Background, Demand Paging, Copy-on-Write, Page Replacement, Allocation of Frames, Thrashing, Mass-Storage Structure: Overview, Disk Structure, Disk Scheduling, Disk Management **File System Interface and Implementation** – File Concept, Access Methods, Directory and Disk Structure, File-System Mounting, File Sharing File-System Structure, File-System Implementation, Directory Implementation, Allocation Methods, Free-Space Management	
10	**Text Books** 1. Abraham Silberschatz, Peter Galvin, Greg Gagne, Operating System Concepts, Wiley, 9th Edition	
11	**Reference Books** 1. Achyut S. Godbole, Atul Kahate, Operating Systems, Tata McGraw Hill 2. Naresh Chauhan, Principles of Operating Systems, Oxford Press 3. Andrew S Tanenbaum, Herbert Bos, Modern Operating Systems, 4e Fourth Edition, Pearson Education, 2016	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

Name of the Course: Theory of Computation

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** The Theory of Computation explores the fundamental principles that define what problems can be solved using computers and how efficiently they can be solved. It lays the groundwork for understanding computation through abstract machines, formal languages, and logical reasoning, making it a cornerstone of theoretical computer science. **Relevance:** This course is highly relevant as it forms the theoretical basis for designing algorithms, compilers, and programming languages. It deepens the student's understanding of the limits of computation, which is critical for solving complex computational problems systematically and effectively. **Usefulness:** The subject helps students learn how to model computational problems using mathematical tools and abstract machines like automata and Turing machines. These skills are vital when optimizing algorithms, designing new computing systems, or developing efficient parsing tools in programming languages. **Application:** Theory of Computation finds applications in compiler design, artificial intelligence, natural language processing, software verification, and cryptography. It also plays a key role in understanding whether problems can be solved algorithmically and what resources are required for their solutions. **Interest:** Students often find this subject intellectually stimulating because it challenges their logical and mathematical thinking. It involves elegant problem-solving techniques and creative ways to model and classify computational problems, sparking curiosity about the power and limits of machines. **Connection with Other Courses:** The course connects closely with subjects like Compiler Design, Artificial Intelligence, Algorithms and Data

		Structures, and Programming Language Theory. It provides the mathematical and logical foundation required for developing more advanced computer science concepts and tools. **Demand in the Industry:** Industries focused on algorithm development, formal verification, secure systems, and artificial intelligence highly value the principles taught in this course. Companies working in automation, language processing, and software correctness increasingly seek professionals with strong theoretical backgrounds. **Job Prospects:** Career opportunities include roles such as compiler engineer, language designer, research scientist, software developer, algorithm designer, and systems programmer. It also serves as a gateway for advanced studies or research in computer science, especially in theoretical and mathematical computing domains.
2	**Vertical:**	Major
3	**Type:**	Theory
4	**Credits:**	2 credits (1 credit = 15 Hours for Theory)
5	**Hours Allotted:**	30 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives (CO):** **CO 1.** Introduce the fundamental concepts of theoretical computer science, including automata theory, formal languages, and computational models. **CO 2.** Understand the mathematical foundations necessary to describe and analyze computation. **CO 3.** Explore different models of computation, including finite automata, pushdown automata, linear bound automata, and Turing machines. **CO 4.** Analyze the decidability and complexity of computational problems. **CO 5.** Introduce the theory of computability and the limitations of algorithmic solutions. **CO 6.** Familiarize students with the classification of problems based on complexity classes such as P, NP, and NP-Complete.	
8	**Course Outcomes (OC):** After successful completion of this course, students would be able to - **OC 1.** Describe the fundamental concepts and significance of Theory of Computation in Computer Science. **OC 2.** Design deterministic and non-deterministic finite automata for regular languages and prove their equivalence.	

	OC 3. Apply regular expressions and grammars to define and generate formal languages. **OC 4.** Construct context-free grammars and analyse their ambiguity, simplification, and normal forms. **OC 5.** Differentiate between complexity classes such as P, NP, NP-Complete, and NP-Hard problems. **OC 6.** Identify and analyze undecidable problems.
9	**Modules** **Module 1 (15 hours):**
	Introduction to Theory of Computation: Basics of Computation, Importance of Theory of Computation in Computer Science, Mathematical Foundations (Sets, Relations, Functions, Proof Techniques) **Automata Theory:** Defining Automaton, Finite Automaton, Transitions and Its properties, Acceptability by Finite Automaton, Nondeterministic Finite State Machines, DFA and NDFA equivalence, Mealy and Moore Machines, Minimizing Automata. **Formal Languages:** Defining Grammar, Derivations, Languges generated by Grammar, Chomsky Classification of Grammar and Languages, Recursive Enumerable Sets, Operations on Languages, Languages and Automata **Regular Languages:** Regular Grammar, Regular Expressions, Finite automata and Regular Expressions, Pumping Lemma and its Applications, Closure Properties, Regular Sets and Regular Grammar Context Free Languages: Context-free Languages, Derivation Tree, Ambiguity of Grammar, CFG simplification, Normal Forms, Pumping Lemma for CFG
	Module 2 (15 hours):
	Pushdown Automata: Definitions, Acceptance by PDA, PDA and CFG **Linear Bound Automata:** The Linear Bound Automata Model, Linear Bound Automata and Languages. **Turing Machines:** Turing Machine Definition, Representations, Acceptability by Turing Machines, Designing and Description of Turing Machines, Turing Machine Construction, Variants of Turing Machine, Decidability and Undecidability, The Church-Turing thesis, Universal Turing Machine, Halting Problem, Introduction to Unsolvable Problems **Computability and Complexity:** Time Complexity and Space Complexity, Big-O Notation, Class P and Class NP, NP-Complete and NP-Hard Problems, Polynomial Reductions, Introduction to Complexity Hierarchies
10	**Text Books** 1. Theory of Computer Science, K. L. P Mishra, Chandrasekharan, PHI,3rd Edition 2. Introduction to Computer Theory, Daniel Cohen, Wiley,2nd Edition 3. Introductory Theory of Computer Science, E.V. Krishnamurthy,Affiliated East-West Press.

11	**Reference Books** 1. Theory of Computation, Kavi Mahesh, Wiley India 2. Elements of The Theory of Computation, Lewis, Papadimitriou, PHI 3. Introduction to Languages and the Theory of Computation, John E Martin, McGraw-Hill Education 4. Introduction to Theory of Computation, Michel Sipser, Thomson	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

Name of the Course: Data Structures

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** The Data Structures course introduces students to the foundational building blocks of programming and software development. It covers the systematic organization, management, and storage of data for efficient access and modification using various data structures. **Relevance:** Data structures are essential for solving computational problems efficiently, making this subject central to the study of computer science. Understanding how data is stored, manipulated, and accessed directly impacts the performance of applications. **Usefulness:** Knowledge of data structures allows students to choose the most appropriate structure for any problem, enabling optimal algorithm design and effective memory utilization. This leads to better program performance and scalability. **Application:** Data structures find wide applications in database indexing, compiler design, network routing, artificial intelligence, and graphics. Real-life scenarios such as job scheduling, expression evaluation, searching, and pathfinding extensively use stacks, queues, trees, and graphs. **Interest:** Students often find this subject intellectually stimulating as it challenges them to think logically and solve problems efficiently. Implementing structures like AVL Trees or graph traversals can be deeply engaging and rewarding. **Connection with Other Courses:** This course is closely linked with algorithms, operating systems, databases, and programming languages. Many advanced subjects assume a working knowledge of data structures, making it a prerequisite for deeper computer science learning. **Demand in the Industry:** Almost every tech role—from software development to system design—requires a strong understanding of data structures. Employers frequently test these concepts during technical interviews and coding assessments.

		Job Prospects: Proficiency in data structures opens doors to careers in software engineering, data science, system architecture, web development, and cybersecurity. It provides a solid base for roles that involve designing efficient and scalable solutions.
2	**Vertical:**	Major
3	**Type:**	Theory
4	**Credits:**	2 credits (1 credit = 15 Hours for Theory)
5	**Hours Allotted:**	30 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives (CO):** **CO 1.** To introduce students to the concept of Abstract Data Types (ADTs) and various data structures for efficient data representation and manipulation. **CO 2.** To develop an understanding of linked structures, including singly and doubly linked lists, and their applications. **CO 3.** To study stack and queue data structures, their implementation, and real-life applications like expression conversion and job scheduling. **CO 4.** To explore non-linear data structures such as trees and graphs, their operations, and their applications in problem-solving. **CO 5.** To understand the principles of priority queues, heaps, and hashing, and their role in efficient data access and management.	
8	**Course Outcomes (OC):** After successful completion of this course, students would be able to - **OC 1.** Define and implement various data structures using Abstract Data Types (ADTs) and understand their classifications and use cases. **OC 2.** Apply operations on linked lists, including traversal, insertion, deletion, and use them in practical applications like polynomial manipulations. **OC 3.** Implement stack and queue operations with array and linked representations, and apply them in real-world scenarios like delimiter checking and scheduling. **OC 4.** Design and traverse tree structures including binary trees, BSTs, AVL trees, and understand their applications in encoding and searching. **OC 5.** Implement graph structures, perform traversals using BFS and DFS, and solve shortest path and connectivity problems. **OC 6.** Use heaps and hashing techniques effectively for priority management, efficient searching, and collision handling in various applications.	
9	**Modules** **Module 1 (15 hours):**	
	Abstract Data Type: Different Data Types, different types of data structures & their classifications, Introduction to ADT, Creating user-specific ADT **Linked Structures**: ADT for linked list, Advantages & Disadvantages, Singly Linked List-Traversing, Searching, Prepending and Removing Nodes, applications of	

	linked list like polynomial equation, ADT of doubly linked list, Advantages & Disadvantages, Insertion and deletion of nodes at various positions **Stacks**: Stack ADT for Stack, Advantages & Disadvantages, Applications of stack like balanced delimiter, prefix to postfix notation **Queues**: Queue ADT, Advantages & Disadvantages, linked representations. Circular Queue operations, Dequeues, applications of queue like job scheduling queues	
	Module 2 (15 hours):	
	Trees: ADT for Tree Structure. Advantages & disadvantages, Binary Tree-Properties, Implementation and Traversals, Binary Search Tree, Balanced BST, Threaded Binary Trees, AVL Trees, Applications of Tree like Huffman Coding, **Priority Queues & Heaps:** Priority Queue, Priority Queue ADT, Advantages and Disadvantages, Applications, Heaps, types of heaps, Heapifying the element, **Graph:** Introduction, Graph ADT, Advantages and Disadvantages, Graph Representation using adjacency matrix and adjacency list, Graph operations like insertion and deletion of nodes, Graph Traversals using BFS & DFS, Applications of Graphs like shortest path algorithms, **Hashing:** Hash Table ADT, Advantages & Disadvantages, Concept of hashing, hash table, hash functions, collision, collision avoidance techniques, Applications of hashing	
10	**Text Books** 1. Introduction to Algorithm, Thomas H Cormen, PHI 2. Data Structures And Algorithms Made Easy, Narasimha Karumanchi, 2021	
11	**Reference Books** 1. Fundamentals of Computer Algorithms, Sartaj Sahni and Sanguthevar Rajasekaran Ellis Horowitz, Universities Press, 2018 2. Data Structures and Algorithms in Python, Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, Wiley, 2016	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

Name of the Course: Computer Science Practical 3

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** This course is a practical extension of theoretical concepts covered in Operating Systems and Data Structures. It introduces students to essential programming constructs required for system-level programming and structured data organization. Through carefully curated lab exercises, students gain insights into concurrency, resource management, scheduling, and structured data processing. **Relevance:** Operating Systems and Data Structures form the foundation of all computing systems and applications. Understanding how data is stored, accessed, manipulated, and how system resources are scheduled and synchronized is vital for every computer science graduate. This practical course equips students with essential skills that are applicable across all domains of software development and system programming. **Usefulness:** This course builds a strong foundation in system-level programming and abstract data handling, essential for any computer science graduate. It enhances the ability to understand and implement core concepts like scheduling, synchronization, and data abstraction through real-time coding tasks. Students gain valuable skills for debugging, memory management, and efficient algorithm design. These skills are crucial for both academic excellence and industry readiness. **Application:** The practical skills developed in this course are directly applicable in the design of operating system modules, file systems, and memory management tools. Data structures like trees, heaps, and graphs are widely used in building compilers, databases, and networking software. The understanding of multi-threading and process synchronization enables students to contribute to applications requiring concurrency, such as games, real-time simulations, and cloud systems. These applications form the backbone of modern software development. **Interest:** Students often find this course engaging due to its interactive and logic-based approach to solving real-world

		problems. Concepts like circular queues for task scheduling or graphs for social network analysis spark curiosity and hands-on involvement. The implementation of real-time synchronization and visual data traversals makes the learning process both stimulating and rewarding. This course encourages creative problem-solving through code. **Connection with Other Courses:** This practical course closely aligns with theoretical subjects like Operating Systems, Data Structures and Algorithms, and Object-Oriented Programming. It also forms the basis for advanced topics such as Distributed Systems, Artificial Intelligence, and Systems Programming. Concepts learned here are frequently reused and expanded upon in courses like Database Systems, Compiler Design, and Software Engineering. Hence, it serves as a vital link across the curriculum. **Demand in the Industry:** Industry consistently seeks professionals proficient in data structures and operating system fundamentals. Skills like process synchronization, memory optimization, and graph traversal are essential for roles in software development, cloud computing, and backend engineering. These competencies are tested during technical interviews at leading firms, including product-based and service-based companies. The ability to translate theoretical knowledge into efficient code is highly valued across tech domains. **Job Prospects:** Completing this course prepares students for roles such as Software Developer, System Programmer, Backend Engineer, or Technical Consultant. It opens opportunities in IT services, product development, and tech startups. With further specialization, students can also pursue careers in AI/ML, cybersecurity, and data engineering. The practical exposure provided here serves as a strong launchpad for core technical positions in the industry.
2	**Vertical:**	Major
3	**Type:**	Practical
4	**Credits:**	2 credits (1 credit = 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	60 hours
6	**Marks Allotted:**	50 Marks

7	**Course Objectives(CO):** **CO 1.** To develop hands-on skills in implementing core concepts of operating systems and data structures. **CO 2.** To simulate and solve real-world problems using process management, synchronization, and memory management. **CO 3.** To strengthen students' understanding of data abstraction and manipulation using linked structures, trees, graphs, and hashing. **CO 4.** To enable students to analyze and compare algorithmic strategies for CPU scheduling, buffer control, and structured data operations. **CO 5.** To foster problem-solving abilities through coding, debugging, and testing of system-level and data structure-oriented programs.
8	**Course Outcomes (OC):** After successful completion of this course, students would be able to - **OC 1.** Design and implement solutions using inter-process communication techniques such as shared memory and message passing. **OC 2.** Apply multithreading, synchronization mechanisms, and scheduling algorithms to solve operating system-related problems. **OC 3.** Construct and manipulate linear and non-linear data structures using custom implementations. **OC 4.** Demonstrate effective use of stack, queue, trees, graphs, and hash tables in algorithm development. **OC 5.** Analyze and evaluate the performance of memory and disk management techniques and abstract data operations. **OC 6.** Apply practical programming knowledge to develop efficient, real-time, and scalable system-level applications.
9	**Modules:-** **Module 1 (30 hours):**
	Practical based on Principles of Operating Systems **Process Communication using Shared Memory** • Understand shared memory concepts in inter-process communication. • Implement producer-consumer synchronization using shared memory and semaphores. • Explore issues of race conditions and how to avoid them. **Process Communication using Message Passing** • Use message queues/pipes to solve the producer-consumer problem. • Compare and contrast shared memory vs. message-passing approaches. • Analyze blocking vs. non-blocking communication. **Threading and Single Thread Control Flow** • Practice thread creation and basic thread lifecycle using standard libraries (e.g., pthreads or Java threads).

- Observe execution order, thread joining, and delays.
- Measure execution time for sequential vs threaded execution.

Multi-threading and Fibonacci Generation

- Implement multi-threading to generate and print Fibonacci sequences.
- Explore thread safety, synchronization when accessing shared variables.
- Introduce concepts of thread pooling and task delegation.

Process Synchronization and Bounded Buffer Problem

- Simulate producer-consumer bounded buffer using mutex and semaphores.
- Implement buffer control with synchronized access.
- Introduce circular queue techniques for managing shared buffers.

Readers-Writers Problem – Synchronization in Shared Access

- Implement reader and writer prioritization.
- Use semaphores to allow multiple readers or exclusive writer access.
- Extend to fairness in access and deadlock prevention.

CPU Scheduling Algorithms (Part 1) – FCFS and Non-preemptive Scheduling

- Simulate First-Come First-Serve scheduling.
- Extend implementation to general non-preemptive scheduling.
- Analyze waiting time, turnaround time, and Gantt chart generation.

CPU Scheduling Algorithms (Part 2) – Round Robin

- Implement Round Robin scheduling with configurable time quantum.
- Compare with FCFS: fairness, turnaround, response time.
- Track context switches and improve queue management.

Memory Management Techniques

- Simulate FIFO and LRU page replacement using page reference strings.
- Measure hit/miss ratios under different reference patterns.
- Extend to include frames and memory constraints.

Disk Scheduling and Simple File System Design

- Simulate FCFS, SSTF, C-SCAN, C-LOOK, RSS for disk head movement.
- Design a basic file system structure with block allocation, directory management, and file operations (create, read, delete).

Module 2 (30 hours):

Practical based on Data Structures

Exploring Abstract Data Types (ADT) & Custom Structures

- Create and manipulate structures to model ADTs like Student, Book, or

Employee.

- Implement basic operations (create, update, delete) using structures.
- Reflect on differences between primitive and abstract data types.

Building and Using Singly Linked Lists

- Construct a dynamic singly linked list with basic operations.
- Apply linked lists to simulate scenarios such as managing a playlist or to-do list.
- Compare static (array) vs dynamic (linked) approaches.

Polynomial Operations Using Linked Lists

- Represent polynomials using linked lists.
- Perform polynomial addition and subtraction by merging lists.
- Use structured representation to reinforce node manipulation.

Working with Doubly Linked Lists

- Create a doubly linked list with forward and backward traversal.
- Implement insertion/deletion at head, tail, and specific positions.
- Use in scenarios like browser history or undo-redo features.

Implementing and Using Stack ADT

- Implement push, pop, peek using arrays or linked lists.
- Solve problems like delimiter matching or undo mechanism.
- Convert expressions from prefix to postfix and evaluate them.

Understanding Queues and Circular Queues

- Develop linear and circular queues to simulate task scheduling.
- Perform enqueue and dequeue with wrap-around logic.
- Discuss memory utilization in linear vs circular queues.

Tree Traversals and Binary Search Trees

- Create a binary search tree (BST) from a dataset.
- Perform and visualize in-order, pre-order, and post-order traversals.
- Use traversal results to derive sorted sequences.

Balanced Trees & Priority Queues

- Insert values and observe AVL tree rebalancing.
- Construct min-heaps or max-heaps and simulate priority queues.
- Use priority queues to manage task priorities (e.g., patient triage, job

	scheduling). **Graph Representations and Traversals** • Represent graphs using adjacency matrices and lists. • Implement BFS and DFS to explore graph components. • Use graphs for mapping routes or exploring social networks. **Hashing Concepts and Collision Handling** • Implement a hash table with chaining or linear probing. • Simulate insertion, deletion, and search with collisions. • Discuss practical hashing applications (e.g., dictionary lookup, indexing).	
10	**Text Books** 1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2022). Operating system concepts (10th ed.). Wiley. 2. Aho, A. V., Ullman, J. D., & Lam, M. S. (2021). Data Structures and Algorithms in Python (Adapted by R. Rao). Pearson India.	
11	**Reference Books** 1. Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). Data Structures and Algorithms in Java (6th ed.). Wiley India. 2. Kanetkar, Y. (2020). Data Structures Through Python (1st ed.). BPB Publications.	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

Name of the Course: JAVA Programming

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** Java is one of the most widely-used, robust, and versatile programming languages in the software industry. It follows the object-oriented programming paradigm and is known for its platform independence, thanks to the Java Virtual Machine (JVM). This course introduces students to the foundational and advanced features of Java, covering core concepts, GUI development, multithreading, exception handling, web application development using Servlets and JSP, and database connectivity through JDBC. **Relevance:** In today's software-driven world, Java is a fundamental language that underpins many enterprise-level, web-based, mobile, and desktop applications. Its syntax and concepts are not only essential for programming in Java but also form the basis for understanding other languages such as C#, Kotlin, and Scala. Java's relevance is further reinforced by its use in Android development, big data (via tools like Hadoop), and large-scale enterprise applications. **Usefulness:** This course builds strong foundational skills in programming and software design. It enables students to construct modular programs using classes, interfaces, and packages. They also learn to handle exceptions, manage concurrent threads, design user-friendly graphical interfaces, and create dynamic, database-connected web applications using technologies like Servlets and JSP. **Application:** The practical aspect of the course allows students to build desktop tools, interactive GUI applications, and basic web portals. By working with JDBC, Swing, and multithreading, students gain hands-on experience in areas that mirror real-world software development scenarios. **Interest:** Java's simplicity, combined with its rich libraries and real-time problem-solving approach, makes learning

		engaging and rewarding. Students enjoy working on mini-projects, interactive applications, and seeing the immediate impact of their code, which deepens their interest and confidence in programming. **Connection with Other Courses:** The skills acquired in this course directly support and enhance learning in related subjects like Data Structures, Database Management Systems, Operating Systems, Web Development, and Software Engineering. It also forms a foundation for advanced electives in Mobile App Development and Enterprise Computing. **Demand in the Industry:** Java developers are in steady demand across industries such as finance, education, e-commerce, and healthcare. Its robustness, scalability, and extensive community support make it a preferred language for backend systems and enterprise-level applications. **Job Prospects:** After completing this course, students are well-prepared for internships and entry-level roles like Java Developer, Backend Developer, Software Engineer, Web Application Developer, and Android App Developer. Mastery of Java also positions students strongly for advanced certifications and competitive programming opportunities.
2	**Vertical:**	VSC
3	**Type:**	Practical
4	**Credits:**	2 credits (1 credit = 15 Hours for Theory or 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	60 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives (CO):** **CO 1.** Provide a strong foundation in Java programming concepts including object-oriented principles and core syntax. **CO 2.** Enable students to design and implement modular programs using classes, interfaces, packages, and exception handling. **CO 3.** Introduce multithreading and collections to manage concurrency and structured data efficiently. **CO 4.** Familiarize students with GUI development using Swing and event handling for desktop applications. **CO 5.** Train learners to build web-based applications using Servlets, JSP, and JDBC for database connectivity and dynamic content.	

8	**Course Outcomes (OC):** After successful completion of this course, students would be able to - **OC 1.** Apply object-oriented programming concepts to develop efficient and maintainable Java applications. **OC 2.** Implement exception handling and multithreading to build robust and concurrent programs. **OC 3.** Use Java Collection Framework to store, manipulate, and retrieve data effectively. **OC 4.** Design user interfaces using Swing components and handle user events in GUI applications. **OC 5.** Connect Java applications with databases using JDBC and perform CRUD operations. **OC 6.** Develop dynamic, session-managed web applications using Servlets and JSP.
9	**Modules:** **Module (30 hours):**
	Java Basics and OOP: History and Features of Java, JVM, JDK, JRE, Java Program Structure, Tokens, Data Types, Operators, OOP Principles: Class, Object, Constructor, this, static, Inheritance, Polymorphism (Overloading/Overriding), Abstraction, Encapsulation, Abstract Classes, Interfaces, Inner Classes, super, Anonymous Classes **Packages & Access Specifiers:** Predefined & user-defined packages, access specifiers **Exception Handling:** Pre-Defined Exceptions, try-catch-finally, throw, throws, custom (user defined) exceptions **Multithreading:** Thread creation, Thread life cycle, Synchronization, wait(), notify(), notifyAll() **Collection Framework:** java.util overview, Interfaces: List, Set, Map, Classes: ArrayList, LinkedList, HashSet, TreeSet, HashMap **Introduction to JSON:** Syntax, DataTypes, JSON with Java
	Module (30 hours):
	Java Foundation Classes & Swing: JFC Overview, Common Swing Components: JFrame, JPanel, JButton, JTextField, JLabel, Layouts, Event Handling using Delegation Event Model, Adapter classes, ActionListener **JDBC:** JDBC Architecture & Drivers, Connecting to DB, Statement, PreparedStatement, ResultSet, Navigating data, ResultSetMetaData, Transactions, Exception handling **Servlets:** Servlet Lifecycle & basic structure, Deployment Descriptor, ServletConfig, ServletContext, RequestDispatcher, Response redirection, Session tracking (Cookies, URL Rewriting, HttpSession), Introduction to Filter API

	Java Server Pages (JSP): JSP Lifecycle & Architecture, Scripting Elements, Directives, Implicit Objects, JSTL basics, Expression Language, CRUD operation overview using JSP **JSON:** Syntax, DataTypes, JSON with Java	
10	**Text Books** 1. Herbert Schildt, Java The Complete Reference, Eleventh Edition, McGraw-Hill Education, 2020 2. Bryan Basham, Kathy Sierra, Bert Bates, Head First Servlets and JSP, O'reilly (SPD), 2018 3. Ivan Bayross, Web Enabled Commercial Applications Development Using Java 2, BPB Publications 4. Java XML and JSON: Document Processing for Java SE by Jeff Friesen January 2019, Apress	
11	**Reference Books** 1. E. Balagurusamy, Programming with Java- A Primer, Tata McGraw-Hill Education India, 2023 2. Programming in JAVA, 2nd Ed, Sachin Malhotra & Saurabh Choudhary, Oxford Press, 2018 3. Joe Wigglesworth and Paula McMillan, Java Programming: Advanced Topics, Thomson Course Technology (SPD) 4. Eric Jendrock, Jennifer Ball, D Carson and others, The Java EE 5 Tutorial, Pearson Education 5. Java Parsing Collection XML JSON: Map List XML JSON Transform by Yang Hu, 2019	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

Sem – IV

Name of the Course: Computer Networks

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** The Computer Networks course provides a foundational understanding of how computers and other devices communicate over various types of networks. It introduces key networking models like OSI and TCP/IP, explores the structure and functionality of different layers such as physical, data link, network, transport, and application, and dives into real-world protocols such as Ethernet, IP, TCP, HTTP, and DNS. The course also addresses modern developments like IPv6, Quality of Service (QoS), and secure communication practices. **Relevance:** n an era where connectivity drives innovation, understanding computer networks is essential for any computer science graduate. This course aligns with the growing need for professionals who can design, troubleshoot, and manage complex networked systems, including those used in cloud computing, data centers, mobile communications, and IoT environments. **Usefulness:** The course is highly useful for learners as it equips them with the theoretical background and practical insights needed to understand how data moves through networks. This knowledge is critical not only for roles in networking but also in software development, cybersecurity, systems administration, and IT infrastructure management. **Application:** Concepts learned in this course have direct application in configuring LANs and WANs, managing IP addresses, analyzing packet data, securing communications, and developing network-based applications. Students will be able to apply their knowledge in creating efficient, scalable, and secure communication systems. **Interest:** Students often find this course interesting due to its hands-on nature and immediate relevance to everyday technologies such as the internet, mobile phones, social media, and streaming services. Simulations and practical case studies help visualize how theoretical concepts apply

		in real-world network scenarios. **Connection with Other Courses:** This course connects closely with subjects such as Operating Systems, Database Management Systems, Web Programming, Cloud Computing, and Cybersecurity. A good understanding of networking is essential for understanding how distributed systems work, how servers handle requests, and how secure communication is maintained. **Demand in the Industry:** There is a consistent demand in the industry for professionals who understand networking principles and can manage network operations. Roles involving cloud platforms (like AWS, Azure), DevOps, and IT support all value networking skills. Knowledge of protocols and architectures is especially important for careers in network engineering, cybersecurity, and system integration. **Job Prospects:** Completing the Computer Networks course equips students for roles like network administrator, support technician, and system engineer. It builds a strong foundation for careers in cybersecurity, cloud computing, and IT infrastructure. The knowledge gained is valuable for industry certifications and in-demand across tech-driven sectors.
2	**Vertical:**	Major
3	**Type:**	Theory
4	**Credits:**	2 credits (1 credit = 15 Hours for Theory or 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	30 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives (CO):** **CO 1.** To introduce students to fundamental concepts of computer networks, network types, architecture, and models including OSI and TCP/IP. **CO 2.** To explain data transmission techniques, bandwidth utilization, switching methods, and transmission media used in modern networks. **CO 3.** To provide an understanding of data link layer functions including MAC, error detection/correction, and LAN protocols. **CO 4.** To equip learners with knowledge of network layer operations such as IP addressing, routing, and protocol analysis including IPv4 and IPv6.	

	CO 5. To introduce transport and application layer protocols and their real-world applications such as web communication, email, and DNS. **CO 6.** To make students aware of modern network trends, QoS mechanisms, and the transition to IPv6 and secure communications.
8	**Course Outcomes (OC):** After successful completion of this course, students would be able to - **OC 1.** Describe network architectures, types, models, and the layered approach in data communication. **OC 2.** Analyze the working of physical and data link layers including signal transmission, media, error detection, and MAC protocols. **OC 3.** Explain the role of switching techniques and multiplexing in efficient communication. **OC 4.** Configure and evaluate IPv4/IPv6 addressing schemes and understand packet forwarding and routing algorithms. **OC 5.** Compare and contrast TCP, UDP, and SCTP protocols and apply them to real-time applications. **OC 6.** Use knowledge of application layer protocols (HTTP, FTP, Email, DNS, etc.) to understand client-server interactions. **OC 7.** Assess Quality of Service (QoS) requirements and identify modern network challenges and solutions like 5G, satellite, and secure communication.
9	**Modules:-** **Module 1 (15 hours):**
	Introduction to Computer Networks: Networking standards and organizations (ISO, IEEE, IETF), Types of Networks: LAN, MAN, WAN, Network topologies and basic hardware **Network Models:** OSI Reference Model – layers, functions, TCP/IP Protocol Suite – layers and comparison with OSI **Physical Layer Concepts:** Data and signals: analog & digital, Signal impairments: attenuation, noise, distortion, Data transmission: bandwidth, throughput, latency, Digital transmission: line coding, analog-to-digital conversion, Transmission modes: simplex, half-duplex, full-duplex **Bandwidth Utilization & Transmission Media:** Multiplexing: FDM, TDM, WDM, Spread Spectrum techniques: DSSS, FHSS, Transmission Media: Guided (Twisted Pair, Coaxial, Fiber Optics) & Unguided Media (Radio, Microwave, Infrared) **Switching Techniques:** Circuit Switching, Packet Switching (connectionless and connection-oriented) **Data Link Layer and Error Handling:** Link layer addressing (MAC), framing concepts, Error detection: Parity, CRC, Checksum, Error correction: Hamming Code, Data link protocols: Stop-and-Wait, Go-Back-N, HDLC, Introduction to MAC: CSMA/CD, CSMA/CA

	Wired & Wireless LANs: Ethernet (standard, fast, gigabit). IEEE 802.11 Wi-Fi, Bluetooth, WiMAX, Cellular telephony: Generations overview (2G–5G), Satellite networks: types and applications	
	Module 2 (15 hours):	
	Network Layer Fundamentals: Packet Switching Concepts, IPv4 Addressing, Subnetting, IP Packet forwarding & routing, Overview of ICMPv4, Basics of Mobile IP **Routing Techniques:** Routing algorithms: Concepts of Distance Vector & Link State Routing, Unicast Routing Protocols: Basic overview of RIP & OSPF **IPv6:** IPv6 Addressing format, comparison with IPv4, ICMPv6, Transition strategies **Transport Layer Protocols,** Transport layer services, User Datagram Protocol (UDP), Transmission Control Protocol (TCP): Features, 3-way handshake **Application Layer & Protocols:** Client-Server architecture: Iterative vs Concurrent Servers, Standard Protocols: HTTP (with HTTPS), FTP, Email (SMTP, POP3, IMAP), TELNET, SSH (modern replacement), DNS **Quality of Service (QoS):** Concepts of Delay, Jitter, Bandwidth, Flow control techniques, Integrated Services (IntServ) and Differentiated Services (DiffServ), Current trends: QoS in video streaming and VoIP	
10	**Text Books** 1. Data Communications and Networking, Behrouz A. Forouzan, Fifth Edition, TMH, 2018. 2. Computer Network, Andrew S. Tanenbaum, David J. Wetherall, Fifth Edition, Pearson Education, 2018.	
11	**Reference Books** 1. Computer Network, Bhushan Trivedi, Oxford University Press, 2016 2. Data and Computer Communication, William Stallings, PHI, 2017	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

Name of the Course: Software Engineering

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** Software Engineering is the foundation for systematic software development. It introduces learners to the lifecycle of software—from planning and modeling to development and testing. The course provides a disciplined approach to creating software that meets user needs and performs reliably under real-world conditions. **Relevance:** In today's digital age, software is integral to almost every sector. This subject is crucial as it prepares students to handle increasing software complexity and ensures they understand the importance of process-driven development and quality standards. **Usefulness:** The course equips students with practical tools and methodologies for planning, estimating, designing, and testing software. It fosters analytical thinking and technical communication, which are vital for handling real-life software projects effectively and efficiently. **Application:** Software Engineering principles are applied in various domains—banking, healthcare, education, mobile app development, and more. The skills learned here help in managing full-scale projects, ensuring user satisfaction and product scalability. **Interest:** The course offers engaging content like UML modeling, agile methods, risk management, and testing strategies. Students enjoy applying these concepts through diagrams, real-world case studies, and collaborative project planning. **Connection with Other Courses:** This subject ties well with courses in Object-Oriented Programming, Database Management, and Final Year Projects. It bridges theoretical knowledge with project execution skills, enhancing overall technical competence. **Demand in the Industry:**

		There is a high demand for software professionals who understand both development and project management. Agile development, DevOps, and software quality assurance are key skills sought by employers globally. **Job Prospects:** Completing this course opens doors to roles such as software developer, quality analyst, system designer, project assistant, and scrum team member. With experience, one can grow into roles like project manager, solution architect, or process consultant.
2	**Vertical:**	Major
3	**Type:**	Theory
4	**Credits:**	2 credits (1 credit = 15 Hours for Theory or 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	30 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives (CO):** **CO 1.** Provide foundational knowledge of software engineering processes, models, and methodologies. **CO 2.** Introduce software requirements analysis and system modeling using UML. **CO 3.** Explain software design principles, project estimation, and scheduling techniques. **CO 4.** Familiarize students with software quality assurance, risk management, and configuration control. **CO 5.** Enable understanding of software testing principles and test-case design techniques.	
8	**Course Outcomes (OC):** After successful completion of this course, students would be able to - **OC 1.** Explain software process models and apply suitable models to project scenarios. **OC 2.** Analyze software requirements and create UML-based system models. **OC 3.** Apply design principles and estimation techniques for software development. **OC 4.** Plan, schedule, and manage software projects effectively using industry practices. **OC 5.** Demonstrate understanding of quality assurance and perform software testing using appropriate methods.	

9	**Modules:-** **Module 1 (15 hours):**
	Introduction to Software Engineering: Nature of Software, Software Engineering: Principles and Practice, Software Process Framework, Layered Technology, Process Framework, Process Patterns, Capability Maturity Model (CMM), Process Assessment **Software Development Models:** Prescriptive Models: Waterfall, Incremental, Rapid Application Development (RAD), Evolutionary Models: Prototyping, Spiral Model, Specialized Models: Component-Based Development, Aspect-Oriented Development, Agile Development: Agile Manifesto, Extreme Programming (XP), Scrum Overview **Requirements Engineering and Analysis:** Requirements Engineering Process, Elicitation Techniques (Interviews, Workshops, Use Cases), Components & Characteristics of a Software Requirements Specification (SRS), Validation of Requirements **System and Object-Oriented Modeling (using UML):** Use Case Diagram, Class Diagram, Sequence Diagram, Activity Diagram, State Chart Diagram, Component & Deployment Diagram
	Module 2 (15 hours):
	Software Design and Architecture: Design Principles: Coupling and Cohesion, Functional-Oriented vs. Object-Oriented Design, System Architecture Design, Design Verification and Validation, Monitoring and Control in Design **Software Metrics and Estimation:** Software Measurement: LOC, Function Point, and Use Case-Based Estimations, Object-Oriented Metrics, Empirical Estimation Models, Introduction to COCOMO II, Estimation in Agile Development, Make/Buy Decision **Software Project Management:** Project Planning: Scope, Feasibility, Resource Estimation, Project Scheduling: Effort Estimation, Time-Line Charts, Gantt Charts, Risk Management: Identification, Projection, RMMM Plan **Software Quality Assurance & Testing:** SQA Activities, Software Reviews, Formal Technical Reviews (FTR), Software Reliability and SQA Plan, Verification & Validation, Testing Principles and Objectives, Test Oracles, Levels of Testing, White-box and Black-box Testing, Test Plan and Test Case Design
10	**Text Books** 1. Software Engineering, A Practitioner's Approach, Roger S, Pressman, 2019 2. Software Engineering: principles and Practices, Deepak Jain, OXFORD University Press, 2008
11	**Reference Books** 1. Software Engineering, Ian Sommerville, Pearson Education, 2017 2. Fundamentals of Software Engineering, Fourth Edition, Rajib Mall, PHI, 2018

	3. Software Engineering: Principles and Practices, Hans Van Vliet, John Wiley & Sons, 2010 4. A Concise Introduction to Software Engineering, Pankaj Jalote, Springer	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

Name of the Course: IoT Technologies

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** The Internet of Things (IoT) is an evolving field where devices connect and communicate to automate tasks and share data. This course lays the groundwork by explaining IoT’s structure, history, key components, and its interaction with the internet and physical objects. **Relevance:** As industries shift toward automation and real-time analytics, understanding IoT has become vital for developing intelligent and connected systems. Its relevance spans domains like healthcare, agriculture, logistics, and smart cities. **Usefulness:** This course equips students with practical knowledge to create prototypes, interface sensors and actuators, and understand protocols. The hands-on exposure enhances the ability to design, implement, and troubleshoot IoT systems. **Application:** Students apply their knowledge in real-life scenarios such as environmental monitoring, smart homes, precision farming, and intelligent transportation systems. The course provides foundational skills to build IoT-based solutions from scratch. **Interest:** IoT offers an exciting blend of hardware, software, and communication, making it engaging for students who enjoy practical problem-solving. Working with microcontrollers and sensors adds a creative, hands-on dimension to learning. **Connection with Other Courses:** This course connects well with subjects like Embedded Systems, Computer Networks, Artificial Intelligence, Cloud Computing, and Mobile Application Development. It forms a practical bridge between theoretical concepts and real-world implementations. **Demand in the Industry:**

		The IoT market is booming with demand for professionals skilled in embedded programming, cloud integration, sensor interfacing, and data security. Industries are looking for people who can build, maintain, and scale IoT systems. **Job Prospects:** Students completing this course can explore roles like IoT Developer, Embedded Systems Engineer, Firmware Developer, IoT Solution Architect, and Application Developer. With further specialization, it opens doors to R&D, automation, and innovation labs in industries across the globe.
2	**Vertical:**	Major
3	**Type:**	Theory
4	**Credits:**	2 credits (1 credit = 15 Hours for Theory or 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	30 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives (CO):** **CO 1.** To introduce the fundamentals and evolution of IoT, including architecture and design principles. **CO 2.** To explore various IoT platforms, sensors, actuators, and communication protocols. **CO 3.** To impart knowledge of system-on-chip (SoC) architecture and interfacing techniques. **CO 4.** To develop skills in building IoT systems and integrating them with web and cloud technologies. **CO 5.** To create awareness about real-world IoT applications and current trends like Edge and Fog Computing.	
8	**Course Outcomes (OC):** After successful completion of this course, students would be able to - **OC 1.** Understand the core concepts, design, and architecture of IoT systems. **OC 2.** Identify and use various sensors, actuators, and IoT development boards like Raspberry Pi, Arduino, and NodeMCU. **OC 3.** Apply appropriate protocols for communication and ensure secure data exchange. **OC 4.** Design simple IoT applications involving data collection, processing, and visualization. **OC 5.** Analyze IoT use cases and appreciate the role of Edge, Fog, and Cloud in modern applications.	

9	**Modules:-** **Module 1 (15 hours):**
	Introduction to IoT: Definition, Characteristics & Scope of IoT, History & Evolution of IoT, IoT vs M2M, IoT Architectures (Three-layer, Five-layer), Physical & Logical Design of IoT Systems, Enabling Technologies in IoT: Cloud Computing, Big Data, AI, Embedded Systems **IoT Components & Frameworks:** Smart "Things" and their identifiers, Overview of IoT Frameworks (Amazon AWS IoT, Google Cloud IoT, Azure IoT Hub) **System on Chip (SoC):** What is SoC? Structure & Characteristics, SoC Elements: FPGA, GPU, APU, Compute Units, Introduction to ARM and atmega328 Architectures **IoT Hardware Platforms:** Overview and comparison: Raspberry Pi, Arduino, NodeMCU, IoT board capabilities, selection criteria for applications **Hardware Interfacing & Communication Protocols:** Basic components: LED, Button, Camera, Motor, 8×8 LED Grid, Communication protocols: PWM, UART, GPIO, I2C, SPI
	Module 2 (15 hours):
	Sensors & Actuators: Digital and Analog Sensors: Temperature, Humidity, Motion, Light, Gas, Ultrasonic, Interfacing Relay Switch, Servo Motor **IoT Protocols & Security:** Protocols: HTTP, MQTT, CoAP, XMPP, UPnP, Privacy and Security Issues in IoT **Web & Cloud Integration in IoT:** Web server setup for IoT, Data exchange with IoT device, Node-RED basics, Introduction to Cloud Platforms for IoT **Wireless Sensor Networks (WSNs):** Basics, Architecture, Types, Role in IoT communication **Edge & Fog Computing:** Definition, Purpose and Use Cases, Edge vs Fog vs Cloud comparison, Edge architectures and communication models **IoT Applications:** Case Studies in Healthcare, Agriculture, Transportation, Smart Cities
10	**Text Books** 1. Introduction to IoT Paperback by Sudip Misra , Anandarup Mukherjee , Arijit Roy , Cambridge Press, 2022 2. Jain, Prof. Satish, Singh, Shashi, "Internet of Things and its Applications", 1st Edition, BPB, 2020. 3. Shriram K Vasudevan, Abhishek S Nagarajan, RMD Sundaram, Internet of Things, Wiley, India, 2019 4. IoT and Edge Computing for Architects - Second Edition, by Perry Lea, Publisher: Packt Publishing, 2020
11	**Reference Books** 1. Internet of Things by Vinayak Shinde, SYBGEN Learning India Pvt. Ltd, 2020

	2. Internet of things, Dr. Kamlesh Lakhwani, Dr. Hemant kumar Gianey, Josef Kofi Wireko, Kamalkant Hiran, BPB Publication, 2020 3. Arduino, Raspberry Pi, NodeMCU Simple projects in easy way by Anbazhagan k and Ambika Parameswari k, 2019. 4. IoT based Projects: Realization with Raspberry Pi, NodeMCU Paperback – February 2020, by Rajesh Singh Anita Gehlot, 2020 5. Mastering the Raspberry Pi, Warren Gay, Apress, 2014	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

Name of the Course: Computer Science Practical 4

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** This major practical course is a blend of two rapidly evolving areas in computing — Computer Networking and the Internet of Things (IoT). While the first part equips students with hands-on experience in network creation, routing, and traffic analysis, the second part transitions them into the world of intelligent connected devices. Students explore the design and deployment of smart systems using Raspberry Pi and Arduino kits, integrate sensors/actuators, and implement cloud-based interactions. **Relevance:** This course is highly relevant in today's digital world where interconnected devices and reliable communication networks form the backbone of every industry. The integration of Computer Networks and IoT ensures students gain comprehensive knowledge about both data communication and real-time embedded systems. As businesses and governments adopt automation and smart systems, professionals with expertise in both domains are in increasing demand. **Usefulness:** The course equips students with practical skills to configure networks, interface hardware, and use protocols for communication and control. It bridges the gap between theory and practice by allowing learners to build, simulate, and test complete systems. These hands-on experiences enhance understanding and build a strong foundation for advanced learning or employment. **Application:** The knowledge gained in this course can be directly applied to developing IoT systems like smart homes, environmental monitoring, health devices, and automated machinery. Students also learn to simulate network infrastructures and deploy routing protocols, which are essential in setting up enterprise-level networks. Real-world applications of this course span domains such as agriculture, transportation, education, and urban planning. **Interest:** This course captures students' interest through interactive hardware activities like blinking LEDs, controlling motors, and streaming sensor data. Tools like Cisco

		Packet Tracer, Node-RED, and Wireshark provide engaging visual and analytical insights into networking and IoT systems. Students are encouraged to innovate and experiment, keeping the learning experience dynamic and enjoyable. **Connection with Other Courses:** The course complements core subjects such as Computer Networks, Microprocessor and Embedded Systems, Operating Systems, and Web Programming. It allows students to apply concepts from these theory courses in a practical setup, promoting interdisciplinary learning. The use of programming and cloud communication also ties into courses like Data Science and Cloud Computing. **Demand in the Industry:** There is a significant and growing demand for professionals skilled in IoT and network technologies across industries such as healthcare, smart infrastructure, manufacturing, and IT services. Companies look for candidates who can develop, deploy, and maintain systems involving both physical hardware and software networking components. This course provides the skillset that aligns with industry trends and emerging technologies. **Job Prospects:** Students completing this course are better prepared for roles such as IoT Developer, Network Administrator, Embedded Systems Programmer, and Cloud-IoT Integrator. They can work in sectors including telecom, automation, R&D, and smart technologies. The practical skills gained make them suitable for both core technical roles and interdisciplinary project teams.
2	**Vertical:**	Major
3	**Type:**	Practical
4	**Credits:**	2 credits (1 credit = 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	60 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives(CO):** **CO 1.** Develop hands-on proficiency in computer networking tools, commands, and simulation using Cisco Packet Tracer. **CO 2.** Explore network topologies, routing protocols (RIP, OSPF, BGP), and IP addressing techniques (static/dynamic). **CO 3.** Understand and apply various IoT communication protocols, platforms, and hardware interfaces using Raspberry Pi/Arduino.	

	CO 4. Interface sensors, actuators, and edge devices for real-world IoT applications. **CO 5.** Integrate hardware with software tools like Node-RED and cloud platforms for complete IoT solutions.
8	**Course Outcomes (OC):** After successful completion of this course, students would be able to - **OC 1.** Use network diagnostic and configuration commands effectively on Windows and Linux systems. **OC 2.** Design and simulate wired and wireless networks using Cisco Packet Tracer with IP configurations and routing protocols. **OC 3.** Analyze network traffic using Wireshark and identify protocol layers and data flow. **OC 4.** Configure and test IoT hardware platforms for device communication and data acquisition. **OC 5.** Implement real-time IoT applications using sensors, actuators, and cloud communication. **OC 6.** Develop integrated solutions using web technologies, IoT protocols, and dashboarding tools.
9	**Modules:-** **Module 1 (30 hours):**
	Practical based on Computer Networks **Exploring Networking Commands via Windows CMD / LINUX Terminal** • Execute and observe the output of: ping, traceroute / tracert, netstat, arp, ipconfig / ifconfig, getmac, hostname, nslookup, pathping, systeminfo • Discuss the purpose and interpretation of each command's output **Learning Focus:** Understanding diagnostic and configuration commands. **Building a Basic Peer-to-Peer Network** Create a network with **two PCs** connected using a **crossover cable** using Cisco Packet Tracer • Assign static IP addresses • Test connectivity with ping and Packet Tracer simulation **Static IP Setup with One Server and Two Clients** Connect **1 server** and **2 computers** using a switch (Cisco Packet Tracer) • Use **static IP addresses** • Configure server services (e.g., HTTP or FTP) and test connectivity from clients **Dynamic IP Allocation with Server and Clients** Use **DHCP service** from a server to assign IPs to **two PCs** (Cisco Packet Tracer) • Enable and configure DHCP on the server

- Verify IP allocation and connectivity using ipconfig

Creating a Mixed Network with Wired and Wireless Devices
One server, two wired PCs, and two **mobile/wireless devices (**Cisco Packet Tracer**)**

- Use appropriate cabling and access points
- Assign IPs and test cross-device communication

RIP Version 1 Routing Across Three Routers

Three routers, each connected to at least **three PCs (**Cisco Packet Tracer**)**

- Implement **RIPv1** routing between routers
- Verify inter-network connectivity using ping and route tables

RIP Version 2 Implementation

Three routers, each connected to at least **three PCs (**Cisco Packet Tracer**)**

- Enable **RIPv2** and observe subnet mask handling
- Use Packet Tracer's simulation mode to observe routing updates

OSPF Routing and Network Hierarchies

Three routers and their networks using **OSPF**
(Three routers, each connected to at least **three PCs (**Cisco Packet Tracer**)**

- Assign area IDs, router IDs, and enable OSPF
- Monitor OSPF neighbour relationships and path selections

BGP for Inter-domain Routing

Three autonomous systems (AS) with routers running **BGP** Three routers, each connected to at least **three PCs (**Cisco Packet Tracer**)**

- Configure BGP with different AS numbers
- Establish peerings and test inter-AS communication

Protocol Analysis with Wireshark

- Set up network transactions for each protocol:
  ICMP (ping), TCP (web browsing), HTTP (via browser), UDP (DNS), FTP (file transfer)
- Apply filters and observe packet contents

Module 2 (30 hours):

Practical based on IoT Technologies

Preparing the IoT Hardware

- Set up Raspberry Pi OS / Arduino IDE
- Configure GPIO settings and test basic connectivity
- Demonstrate pin layout and onboard peripherals

	GPIO – Light the LED (with and without Button) • Blink LED using Python (Raspberry Pi) or C++ (Arduino) • Add a push button to toggle LED ON/OFF **SPI Interface – Camera Module Integration** • Connect a Pi camera module (or SPI camera for Arduino) • Capture an image or short video • Store file or stream it locally **8x8 LED Grid Control (Matrix LED Programming)** • Connect an 8×8 LED matrix module • Program animations or scrolling text patterns • Explore logical formulas for patterns **PWM – Stepper Motor Control** • Interface a stepper motor using a motor driver • Control direction and vary speed using PWM signals • Observe effect of duty cycle changes on motor movement **Node-RED for IoT Dashboard** • Install and configure Node-RED on Raspberry Pi • Create a flow to turn LED ON/OFF via browser • Add visual interface for sensor data (e.g., temperature) **Sensor Integration – Analog & Digital Sensors** • Interface multiple sensors (LDR, DHT11, Gas) • Collect and display data on serial monitor / OLED / LCD • Trigger actions (e.g., fan ON if temp > threshold) **Web Trigger – Control GPIO from Web Server** • Host a simple Flask web app (Raspberry Pi) or ESP Web Page (Arduino) • Control a set of LEDs via buttons on a webpage • Ensure real-time response and feedback **IoT Protocol – Send Sensor Data Online** • Use HTTP or MQTT to push sensor values to a cloud server (e.g., Thingspeak) • Implement publishing logic with timestamps • Use Arduino/NodeMCU or Raspberry Pi as the publisher

	Integration – Smart Monitoring System • Combine sensors, actuators, communication protocol, and web/cloud • Example: A Smart Weather Station that logs temp/humidity online and triggers fan/LED alerts ***Note: The above practicals can be performed on Raspberry Pi Kits and / or Arduino kits as per the need***	
10	**Text Books** 1. Kurose, J.F. & Ross, K.W. (2021). *Computer Networking: A Top-Down Approach* (7th ed.). Pearson Education India. 2. Ramya, V., & Shanmuga Priya, K. (2019). *Practical Internet of Things: Concepts, applications and security*. Chennai: Wiley India Pvt. Ltd.	
11	**Reference Books** 1. Forouzan, B.A. (2017). *Data Communications and Networking* (5th ed.). McGraw Hill Education India. 2. Bahga, A. & Madisetti, V. (2014). *Internet of Things: A Hands-on Approach*. Universities Press India.	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

Name of the Course: Mobile Application Development

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** Mobile devices are ubiquitous in modern life, and with them comes the need for innovative and functional mobile applications. This course introduces students to the world of Android app development using Kotlin, the officially recommended programming language by Google. Learners get hands-on exposure to building real-time applications using Android Studio, enabling them to create robust, secure, and user-friendly mobile applications. **Relevance:** This course bridges academic programming knowledge with practical industry skills. In a world where mobile-first solutions dominate, Android development is one of the most sought-after technical skills, with Kotlin leading the trend due to its expressive syntax and reliability. **Usefulness:** This course provides hands-on experience with real-time Android app development using Kotlin, helping students bridge theoretical knowledge and practical application. Learners gain exposure to essential skills like UI design, database integration, and deployment practices, which are critical in the modern software development cycle. **Application:** Students can build interactive apps for domains like education, health, business, and entertainment. Through the use of features like Firebase, media handling, and location services, learners can design complete, data-driven mobile solutions suitable for real-world implementation. **Interest:** The course is engaging due to its visual and interactive nature — students see their code come to life in the form of functioning apps. Working with media, animations, sensors, and camera access makes the learning process exciting and creatively fulfilling. **Connection with Other Courses:** This course builds directly on programming concepts

		taught in Object-Oriented Programming and integrates well with Database Management Systems through SQLite and Firebase. It also relates to Software Engineering principles for structured app development and UI/UX Design for front-end aesthetics. **Demand in the Industry:** Android dominates the global mobile market, making Android app development a high-demand skill. With Google's backing of Kotlin, there is a rising need for developers proficient in this language, especially in the mobile-first product ecosystem. **Job Prospects:** Successful completion opens doors to roles like Android Developer, Mobile App Engineer, UI/UX Developer, and Firebase Backend Integrator. Students can also work as freelancers, contribute to startups, or launch their own apps on the Google Play Store.
2	**Vertical:**	SEC
3	**Type:**	Practical
4	**Credits:**	2 credits (1 credit = 15 Hours for Theory or 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	60 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives (CO):** **CO 1.** Understand the fundamentals of mobile app development using Kotlin and Android Studio. **CO 2.** Design and build interactive and responsive user interfaces for Android devices. **CO 3.** Utilize core Android components such as Activities, Intents, and Fragments. **CO 4.** Store, retrieve, and manipulate data using SQLite and Firebase Realtime Database. **CO 5.** Implement multimedia, location-based, and background services in Android apps. **CO 6.** Deploy and publish Android apps with proper versioning and signing protocols. **CO 7.**	
8	**Course Outcomes (OC):** After successful completion of this course, students would be able to - **OC 1.** Set up Android Studio and develop basic Kotlin applications with UI interaction. **OC 2.** Apply object-oriented programming concepts using Kotlin for mobile application logic.	

	OC 3. Use core Android components to develop modular and multi-screen applications. **OC 4.** Create dynamic UIs using layouts, fragments, menus, and handle user interactions efficiently. **OC 5.** Store and retrieve data using local databases and cloud services like Firebase. **OC 6.** Integrate media, camera, GPS, and background services into functional applications. **OC 7.** Package, sign, and deploy Android applications to the Google Play Store.
9	**Modules:-** **Module 1 (30 hours):**
	Getting Started with Android Studio & Kotlin: Setting up Android Studio, AVD, and first Kotlin-based app, Kotlin basics: variables, data types, type conversion, operators, Simple user input/output using TextView, EditText, Button **Kotlin Control Flow & OOP Basics in Action:** Control statements: if, when, loops, Functions, default arguments, extension functions, OOP concepts: classes, objects, inheritance, companion object **Android Core Components:** Activities, Intents (explicit/implicit), Activity lifecycle: demo with logs and state changes, Fragments: modular UI creation, Toasts, Dialogs, and simple navigation **Layout Design & UI Interactions:** Layouts: Linear, Relative, ConstraintLayout, Views and UI Controls: TextView, EditText, Button, ImageView. Event handling: onClickListener, simple data validation, Styling UI: themes, styles, and manifest configuration
	Module 2 (30 hours):
	Working with Lists & Menus: ListView, RecyclerView, Adapter usage, Menus: options, context, and popup menus, Fragments with RecyclerView navigation **Data Persistence & Firebase Integration:** SharedPreferences, SQLite basics and CRUD operations, Firebase Realtime Database: read/write, rules, testing, Dynamic UI based on data **Multimedia, Animations & Camera Access:** Using ImageView, switching images, Playing audio using MediaPlayer, Simple animations with XML, Accessing device camera (capture & display) **Location, Background Tasks & App Deployment:** Accessing location (GPS), Background tasks using JobScheduler, App signing and versioning, Deploying to Google Play (demo or mock submission)
10	**Text Books** 1. How to Build Android Apps with Kotlin: A hands-on guide to developing, testing, and publishing your first apps with Android, Alex Forrester, Packt Publishing, 2021 2. Android Programming: Crafting UI/UX using Kotlin, SYBGEN Learning, 2020

11	**Reference Books** 1. Head First Android Development: A Learner's Guide to Building Android Apps with Kotlin Dawn Griffiths, 3rd Edition, O'Reilly Media, 2021 2. Android Studio 4.2 Development Essentials - Kotlin Edition: Developing Android Apps Using Android Studio 4.2, Kotlin and Android Jetpack, Neil Smyth, Payload Media, 2021 3. Android Programming with Kotlin for Beginners, John Horton, Packt Publishing, 2019	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

Name of the Course: MEAN Stack Development

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** MEAN Stack is a popular JavaScript-based technology stack used for developing full-stack web applications. It combines MongoDB (database), Express.js (backend web framework), Angular (frontend framework), and Node.js (runtime environment). The MEAN stack allows developers to use JavaScript throughout the application—on the front-end, back-end, and database layer—making development efficient and consistent. **Relevance:** With the increasing demand for cross-platform, real-time, and scalable applications, MEAN stack has emerged as a go-to technology for startups, enterprises, and product-based companies. This course addresses the industry need for developers skilled in full-stack JavaScript development, ensuring that learners stay up-to-date with modern tools and practices. **Usefulness:** The MEAN Stack course equips learners with the ability to build end-to-end web applications using JavaScript across all layers of development. It fosters an understanding of client-server communication, data flow, and modern design patterns, making learners self-sufficient developers. The unified use of JavaScript simplifies the learning curve and enhances productivity in real-world projects. **Application:** This course enables students to create responsive web applications such as e-commerce platforms, online booking systems, admin dashboards, social networking sites, and real-time chat apps. Learners can implement full-stack functionalities including database connectivity, routing, authentication, and RESTful services. It encourages building portfolio projects that demonstrate industry-relevant skills. **Interest:** The integration of hands-on labs, live coding sessions, and real-time application building keeps learners engaged and motivated. Learners experience immediate

		output and interactivity, which fuels curiosity and creative problem-solving. The modularity of the stack allows learners to explore each component deeply or specialize in their area of interest. **Connection with Other Courses:** This course complements earlier foundational subjects like HTML, CSS, JavaScript, Web Programming, and Database Management Systems. It serves as a bridge to advanced areas like Mobile App Development (using Ionic or React Native), Cloud Deployment, and DevOps. Students familiar with object-oriented programming and REST APIs will find a smooth transition into this full-stack course. **Demand in the Industry:** MEAN Stack development is widely adopted by startups and established firms alike due to its cost-efficiency and scalability. Employers are actively seeking developers with cross-functional abilities who can handle both frontend and backend tasks. With JavaScript being one of the most in-demand languages globally, MEAN Stack developers are highly valued in the job market. **Job Prospects:** Learners can pursue roles such as MEAN Stack Developer, Full Stack JavaScript Developer, Node.js Developer, Angular Frontend Developer, and API Developer. Opportunities are available in product development companies, IT consulting firms, SaaS platforms, and freelance markets. The skillset also serves as a foundation for launching independent software products or services.
2	**Vertical:**	SEC
3	**Type:**	Practical
4	**Credits:**	2 credits (1 credit = 15 Hours for Theory or 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	60 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives(CO):** **CO 1.** To provide a strong foundation in the MEAN (MongoDB, Express.js, Angular, Node.js) Stack for full-stack web development. **CO 2.** To develop skills to build dynamic, data-driven web applications with server-side and client-side integration.	

	CO 3. To introduce students to RESTful API development, routing, middleware, and database operations. **CO 4.** To promote real-time application development using modern JavaScript frameworks and tools. **CO 5.** To offer hands-on experience through mini-projects and practical implementation.
8	**Course Outcomes (OC):** **OC 1.** Design, develop, and deploy full-stack web applications using the MEAN stack. **OC 2.** Build secure and scalable back-end APIs using Node.js and Express.js. **OC 3.** Develop responsive and dynamic front-end interfaces using Angular. **OC 4.** Perform CRUD operations in MongoDB using Mongoose. **OC 5.** Integrate the front-end and back-end for seamless data flow and user experience.
9	**Modules:-** **Module 1 (30 hours):**
	Introduction to MEAN Stack: Understanding Full Stack Web Development, Overview of MEAN Architecture & Workflow, Setting up the MEAN Development Environment **Node.js Fundamentals:** Introduction & Installation of Node.js, Understanding package.json & npm, Node Modules & Module Exports, Event Loop and Asynchronous Programming, Creating Basic Node.js HTTP Server, Handling Routes and Requests **Express.js Framework:** Introduction to Express, Creating Express Applications, Using Middleware (Built-in, Custom, Third-party), RESTful Routing in Express, Creating API Endpoints (GET, POST, PUT, DELETE), Organizing Express Apps with Router Modules, Error Handling in Express, Environment Configuration using dotenv **MongoDB with Mongoose:** Installing MongoDB and MongoDB Compass, Understanding Collections, Documents, Using Mongo Shell and MongoDB Compass. Introduction to Mongoose ODM, Defining Schemas & Models, Performing CRUD Operations, MongoDB Validation & Indexing Basics **Introduction to Angular:** Installing Angular CLI & Creating Angular App, Angular Project Structure, Anatomy of an Angular Component, One-way and Two-way Data Binding, *ngIf, *ngFor Directives, Using Services & Dependency Injection, Component Communication (Input/Output decorators)
	Module 2 (30 hours):
	Angular Modules & Routing: Creating Feature Modules, Angular Routing: Setup and Navigation. RouterLink, RouterOutlet, Route Params, Lazy Loading of Modules

	Forms in Angular: Template-driven vs Reactive Forms, Building Forms using Reactive Forms Module, Form Validation and Error Handling, Submitting Form Data to Backend APIs **Connecting Angular to REST API:** HttpClientModule and Services, Making GET, POST, PUT, DELETE Requests, Displaying Server Data in Templates, Handling Observables and Async Pipe, Error Handling in HTTP Requests **CRUD Application with Angular + Node + MongoDB:** Building a Complete CRUD App (e.g., Student Records / Notes App), Integrated Frontend and Backend, Form Validation and User Notifications, Deploying to Cloud (Vercel/Render for Backend, Netlify for Frontend) **Introduction to Deployment:** Preparing Angular Build, Serving Angular App with Express (Single Deployment), Environment Variables and Production Setup, Introduction to Git and Version Control	
10	Text Books 1. Node.js, MongoDB and Angular Web Development: The definitive guide to using the MEAN stack to build web applications by Brad Dayley, Brendan Dayley, Caleb Dayley, Pearson, 2018. 2. Full Stack Javascript Development with Mean - MongoDB, Express, AngularJS, and Node.JS by Adam Bretz, Colin J Ihrig, Shroff/SitePoint, 2015	
11	**Reference Books** 1. Pro Mean Stack Development, Elad, Elrom, Apress, 2016 2. Mastering MEAN Stack, Pinakin Ashok Chaubal, bpb, 2023	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**

EVALUATION SCHEME

A. Evaluation for Theory Courses (2 Credit Courses)

I. Internal Evaluation for Theory Courses – 20 Marks

a) for all courses other than Software Engineering course

Sr. No.	Component
1	**Class Tests** Class Test 1 on Module 1: 10 marks Class Test 2 on Module 2: 10 marks **Average of 2 Class Tests: 10 marks**
2	**Assignments** Assignment on Module 1: 5 marks Assignment on Module 2: 5 marks **Total of 2 Assignments: 10 marks**
	Total: 20 Marks

b) for Software Engineering course

Select any one sample project* and prepare the following deliverables:

1. Problem Statement and SRS
2. DFD & Structured Chart
3. Use Case Diagram
4. Class Diagram, Object Diagram
5. State-chart & Activity Diagrams
6. Sequence Diagram, Collaboration Diagram
7. Component & Deployment Diagrams
8. Using Function Point (FP) Method
9. Gantt Chart / PERT Chart
10. Test Cases for Unit, Integration, White Box & Black Box Testing

Documentation: 10 marks Presentation: 10 marks
Total: 20 marks

**** List of sample projects**:*

a. Student Result Management System
b. Library management system
c. Inventory control system
d. Accounting system
e. Fast food billing system
f. Bank loan system
g. Blood bank system
h. Railway reservation system
i. Automatic teller machine
j. Video library management system
k. Hotel management system
l. Hostel management system
m. Share online trading
n. Hostel management system
o. Resource management system
p. Court case management system

II. External Examination for Theory Courses – 30 Marks

A **Semester End Theory Examination** of **1 hour duration** for **30 marks** as per the paper pattern given below:

Total Marks: 30 **Duration: 1 Hour**

Question	**Based On**	**Options**	Marks
Q. 1	Module 1	*Any 2 out of 4*	10
Q. 2	Module 2	*Any 2 out of 4*	10
Q. 3	Module 1 & 2	*Any 2 out of 4*	10

B. Evaluation for Practical Courses (2 Credit Courses)

I. Internal Evaluation for Practical Courses – 20 Marks

Sr. No.	**Component**
1	**Practical Assignments / Experiments / Hands-On Tests / Presentations / Demonstrations / Online Class Test / Case Studies: 15 marks**
2	**Journal: 5 marks**
	Total: 20 Marks

II. External Examination for Practical Courses – 30 Marks

A **Semester End Practical Examination** of **2 hours duration** for **30 marks** as per the paper pattern given below.

Total Marks: 30 **Duration: 2 Hours**

Question	**Practical Question Based On**	**Marks**
Q. 1	Module 1	15
Q. 2	Module 2	15

1. **Certified Journal** is **compulsory** for appearing at the time of Practical Exam
2. Minimum **80% practical** are required to be completed.

Letter Grades and Grade Points:

Semester GPA/ Programme CGPA Semester/ Programme	**% of Marks**	**Alpha-Sign/ Letter Grade Result**	**Grading Point**
9.00 - 10.00	90.0 – 100	O (Outstanding)	10
8.00 - < 9.00	80.0 - < 90.0	A+ (Excellent)	9
7.00 - < 8.00	70.0 - < 80.0	A (Very Good)	8
6.00 - < 7.00	60.0 - < 70.0	B+ (Good)	7
5.50 - < 6.00	55.0 - < 60.0	B (Above Average)	6
5.00 - < 5.50	50.0 - < 55.0	C (Average)	5
4.00 - < 5.00	40.0 - < 50.0	P (Pass)	4
Below 4.00	Below 40.0	F (Fail)	0
Ab (Absent)	----	Ab (Absent)	0

Sd/-
Sign of the BOS Chairman
Dr. Jyotshna Dongardive
Ad-hoc BOS (Computer Science)

Sd/-
Sign of the Offg. Associate Dean
Dr. Madhav R. Rajwade
Faculty of Science & Technology

Sd/-
Sign of Offg. Dean
Prof. Shivram S. Garje
Faculty of Science & Technology