

BTech - Computer Science and Engineering
Proposed Curriculum (2025 onwards)

Total credits – 160

Credit breakup –

- **Core – 79**
- **Electives- 21**
- **Experiential Learning – 8**
- **CCC/UWE – 40**
- **Project – 12**

Specializations offered –

- **Artificial Intelligence**
- **Data Science**
- **Cybersecurity**
- **Systems and Networks**

Program Structure | Computer Science and Engineering

Semester 1 (20 credits)				
Course Code	Title	Credits	Category	Prerequisites
CSD1001	Problem Solving Using Programming	4(3:0:1)	Major Core	None
MAT1003	Multivariate Calculus	4(3:1:0)	Major Core	None
PHY1011	Fields, Waves & Quanta	5(3:1:1)	Major Core	None
ECE1001	The Electron's Path: Fundamentals of EEE	3(2:0:1)	Major Core	None
SOE4801	Introduction to Engineering: Ideas to Impact	2(1:0:1)	Experiential Learning	None
CHD1001	Nature's Code: Chemistry & Biology	2(2:0:0)	Major Core	None

Semester 2 (18 credits)				
Course Code	Title	Credits	Category	Prerequisites
MAT1004	Linear Systems and Transforms	4(3:1:0)	Major Core	None
MED1001	The Matter of Materials	2(2:0:0)	Major Core	None
CED1001	Forces in Action	3(2:0:1)	Major Core	None
SOE4802	Design to Reality: CAD & 3D Printing	2(1:0:1)	Experiential Learning	None
ECE1002	Connected Intelligence: Sensors & IoT	3(2:0:1)	Major Core	None
CED1002	Environment & Sustainability ^s	4(3:0:1)	UWE	None

Semester 3 (23 credits)				
Course Code	Title	Credits	Category	Prerequisites
CSD2001	Data Structures and Algorithms	4(3:0:1)	Core	CSD1001
CSD2002	Discrete Mathematics	4(3:1:0)	Core	CSD1001
CSD2003	Computer Organization	5(3:1:1)	Core	CSD1001
CSD2004	Object-Oriented Programming	4(3:0:1)	Core	CSD1001
	CCC/UWE	6	CCC/UWE	NA

Semester 4 (22 credits)				
Course Code	Title	Credits	Category	Prerequisites
CSD2005	Operating Systems	4(3:0:1)	Core	CSD2001, CSD2003
CSD2006	Introduction to Probability and Statistics	4(3:1:0)	Core	NA
CSD2007	Applied Linear Algebra	4(3:1:0)	Core	NA
CSD4801	Introduction to Database Systems	4(3:0:1)	Experiential Learning	CSD2001
	CCC/UWE	3	CCC/UWE	NA
	Major Elective 1	3	Elective	NA

Semester 5 (23 credits)				
Course Code	Title	Credits	Category	Prerequisites
CSD4001	Computer Networks	4(3:0:1)	Core	CSD2005
CSD4002	Artificial Intelligence and Machine Learning	4(3:0:1)	Core	CSD2007
	Major Elective 2	3	Elective	
	Major Elective 3	3	Elective	

	CCC/UWE	9	CCC/UWE	
--	---------	---	---------	--

Semester 6 (24 credits)				
Course Code	Title	Credits	Category	Prerequisites
CSD4003	Formal Languages and Automata Theory	4(3:1:0)	Core	CSD2001
CSD4004	Advanced Algorithms	4(3:0:1)	Core	CSD2001
CSD4005	Advanced Computer Architecture	4(3:1:0)	Core	CSD2005, CSD2003
	Major Elective 4	3	Elective	
CSD4901	Software Engineering	3 (2:0:1)	Project Basket	
	CCC/UWE	6	CCC/UWE	

Semester 7 (21 credits)				
Course Code	Title	Credits	Category	Prerequisites
	Major Elective 5	3	Elective	
	Major Elective 6	3	Elective	
	Major Elective 7	3	Elective	
	CCC/UWE	6	CCC/UWE	
CSD4902	Project-1	6	Project Basket	CSD2001

Semester 8 (9 credits)				
Course Code	Title	Credits	Category	Prerequisites
CSD4903	Project-2/Internship	3	Project Basket	CSD2001
	CCC/UWE	6	CCC/UWE	

Appendix – I : List of courses (Core, Experiential, Project basket and Electives)

Appendix-II: List of specializations

Appendix-III: List of interdisciplinary specializations

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD1001
4. Course Title	Problem Solving Using Programming
5. Credits (L:T:P)	4(3:0:1)
6. Contact Hours (L:T:P)	3:0:2
7. Prerequisites	NA
8. Major Core for	CSE, ECE, Mechanical, Civil, Chemical
9. Major Elective for	NA

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This course introduces students to the fundamentals of problem-solving through programming using the C programming language. It emphasizes computational thinking, algorithmic design, and structured programming techniques for developing efficient software solutions to real-world problems. Students learn to analyze problems, design algorithms, and implement, test, and debug programs using programming constructs such as variables, data types, operators, control structures, functions, arrays, strings, pointers, structures, and file handling. The course also introduces modular programming, good coding practices, and basic problem-solving strategies that form the foundation for advanced courses in computer science and engineering. Through a combination of lectures, laboratory exercises, and programming assignments, students develop logical reasoning, analytical thinking, and practical programming skills essential for software development and engineering applications.

11. Course Objectives

- Develop computational thinking and systematic problem-solving skills using programming.
- Introduce the fundamental concepts of programming, including data types, operators, control structures, functions, arrays, pointers, structures, and file handling.
- Enable students to analyze computational problems, design appropriate algorithms, and implement efficient and correct solutions using the C programming language.
- Foster good programming practices, including modular program design, code readability, debugging, testing, and documentation.
- Build a strong foundation for advanced studies in computer science, software engineering, data structures, algorithms, and other programming-intensive courses.

12. Learning Outcomes

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

- Analyze computational problems and develop appropriate algorithmic solutions using logical reasoning and computational thinking.
- Apply fundamental programming concepts, including variables, data types, operators, expressions, control structures, functions, arrays, pointers, structures, and file handling, to develop C programs.
- Design, implement, test, and debug modular programs to solve computational problems efficiently and correctly.
- Use appropriate programming practices, including code documentation, modularization, and testing techniques, to produce readable, maintainable, and reliable software.
- Evaluate alternative programming approaches and select suitable data representations and algorithms for solving simple engineering and scientific problems.

13. Curriculum Content

- Introduction to Problem Solving and Programming- Introduction to computers and programming, problem-solving methodology, algorithms and flowcharts, structure of a C program, variables, constants, data types, operators, expressions, input/output statements, assignment statements,

type conversion, programming examples, logical expressions, conditional statements (if, if-else, nested if, switch).

- Iterative Programming and Arrays- Iteration constructs (while, do-while, for loops), nested loops, programming applications, one-dimensional arrays, array operations, searching techniques (linear search, binary search), elementary sorting techniques (bubble sort, selection sort), introduction to top-down program development and stepwise refinement.
- Modular Programming- Functions, function declaration and definition, parameter passing, call by value, recursion, scope and lifetime of variables, using arrays with functions, strings and string manipulation using standard library functions, programming applications.
- Pointers and User-Defined Data Types - Pointers, pointer arithmetic, pointers and arrays, parameter passing using pointers, dynamic memory allocation (malloc, calloc, realloc, free), structures, nested structures, arrays of structures, applications of structures and pointers.
- Advanced Programming Concepts- Two-dimensional arrays, file handling (text and binary files), file operations, command-line arguments (optional), debugging and testing techniques, integrated problem-solving using modular programming, top-down design, and stepwise refinement.

List of Labs

- **Introduction to the C Programming Environment:** Writing, compiling, executing, and debugging simple C programs. Input/output operations and arithmetic expressions.
- **Operators and Decision Making:** Programs using arithmetic, relational, logical, and conditional operators. Applications of if, if-else, nested if, and switch statements.
- **Iterative Programming:** Programs using for, while, and do-while loops. Pattern generation and numerical computations.
- **One-Dimensional Arrays:** Array operations, searching (linear and binary), and sorting (bubble and selection).
- **Functions:** User-defined functions, parameter passing, function decomposition, and arrays with functions.
- **Recursion:** Recursive solutions for mathematical and algorithmic problems.
- **Strings:** String manipulation using standard library functions and user-defined string processing functions.
- **Structures:** Structures, nested structures, arrays of structures, and applications using structures
- **Pointers:** Pointer variables, pointer arithmetic, pointers and arrays, and parameter passing using pointers.
- **Integrated Programming Assignment:** Design and implementation of a modular program that integrates functions, arrays, pointers, structures, dynamic memory allocation, and file handling using a top-down

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Labs Assessment – 30%
- Quizzes – 20%

15. Textbooks and References

- Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Pearson Education, 2nd Edition, 1988 (Reprint Edition).
- Yashavant Kanetkar, Let Us C, BPB Publications, 19th Edition, 2023.
- E. Balagurusamy, Programming in ANSI C, McGraw-Hill Education, 9th Edition, 2024.
- Reema Thareja, Programming in C, Oxford University Press, 2nd Edition, 2016.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD2001
4. Course Title	Data Structures and Algorithms
5. Credits (L:T:P)	4(3:0:1)
6. Contact Hours (L:T:P)	3:0:2
7. Prerequisites	CSD1001
8. Major Core for	CSE
9. Major Elective for	NA

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This course provides a comprehensive introduction to essential data structures, including Arrays, Stacks, Queues, Linked Lists, Trees, Graphs, and Hashing along with Searching and Sorting techniques. It covers key concepts of algorithms such as asymptotic notation and provides insights into both theoretical and empirical analysis of algorithms—covering iterative and recursive techniques.

11. Course Objectives

The purpose of this course is to provide the students with solid foundations in the basic concepts of data structures and algorithms as follows:

- To teach the students how to select and design data structures and algorithms that are appropriate for problems that they might encounter.
- To demonstrate methods for proving algorithm correctness and analyzing computational complexity.
- To provide a balanced combination of theoretical foundations and practical implementation experience.

12. Learning Outcomes

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

- Demonstrate the knowledge of how to measure the complexity of an algorithm, including best-case, worst-case, and average complexities as functions of the input size, as well as classification in terms of asymptotic complexity classes.
- Implement and analyze searching, sorting, insertion, deletion, and traversal operations on linear and non-linear data structures.
- Design and implement hash tables using appropriate collision-resolution techniques.
- Apply algorithmic design paradigms including recursion, divide-and-conquer, greedy methods, and dynamic programming to solve computational problems.
- Implement and analyze trees, priority queues, graphs, and graph algorithms for efficient problem solving.

13. Curriculum Content

- **Introduction to Data Structures (DS) and Algorithm:** Overview and Importance of DS, Asymptotic Notations, Time and Space Complexity Analysis
- **Arrays:** Implementation and Operations of an array, Single and Multi-dimensional Arrays, Applications
- **Linked Lists:** Implementation and Operations of linked lists, Doubly linked lists, and circular linked lists, Applications
- **Stacks:** Operations of the stack, Array and Linked list Representation of Stack, Applications of stack – Recursion, Polish Notation, Expression Evaluation.
- **Queues:** Operations of queue, Array and Linked list Representation of Queue, Circular Queue, Priority Queue, Applications
- **Sorting and Searching Algorithms:** Insertion sort, Selection sort, Bubble sort, Merge sort, Quick sort, Heap sort, Counting, Radix, and Bucket sort; Linear and Binary search

- **Trees:** Representation and Operations of binary tree, Binary tree traversals (Preorder, Inorder, Postorder), Binary Search Tree (BST), AVL tree, Multiway search tree: B-tree and B+ tree
- **Hashing:** Collision Resolution Techniques (Chaining, Open Addressing), Linear Probing.
- **Graphs:** Graph Representation: Adjacency Matrix and Adjacency List; Graph Traversal Algorithms: Depth-First Search (DFS) and Breadth-First Search (BFS); Shortest Path Algorithms, Minimum Spanning Trees.

List of Labs

- **Basics of C Programming** - A refresher session to ensure proficiency in fundamental C concepts necessary for data structures. Focuses on memory mapping via pointers, functional modularity, and understanding the call stack through recursion.
- **Time Complexity** - An experiential lab to bridge theoretical Big-O notation with practical execution time. Students will analyze how different algorithms scale as input sizes grow.
- **Singly Linked List (SLL)** - Introduction to dynamic data structures. Students will move away from contiguous array storage to node-based pointer chaining, mastering dynamic memory allocation and pointer manipulation.
- **Circular Linked List (CLL)** - An extension of the linked list where the last node points back to the first node. This session emphasizes handling edge cases where a null pointer doesn't exist to signify the end of a list.
- **Doubly Linked List (DLL)** - Introduces bi-directional traversal by maintaining two pointers (next and prev) per node. Focuses on the trade-off between increased pointer overhead and flexibility in movement.
- **Stack and Queue** - Implementation of restricted access linear data structures. Students will explore LIFO (Last-In-First-Out) and FIFO (First-In-First-Out) operations using arrays or linked lists.
- **Sorting Algorithms** – Comparative implementation and analysis of Insertion, Merge, Quick, and Heap Sort.
- **Binary Tree and Binary Search Tree (BST)** - **Description:** Moving into non-linear data structures. Students learn hierarchical data representation and the properties of a BST where left child < root < right child.
- **Hashing** - Focuses on achieving O(1) average-case search time. Students will explore hash functions and learn how to handle data collisions systematically.
- **Graph Representation** - **Description:** Introduces networking and relational data structures. Students will implement the two primary methods of storing non-linear, interconnected graph nodes in memory.
- **Breadth First Search (BFS) and Depth First Search (DFS)** - Practical application of graphs and linear data structures. Students will use queues and stacks (or recursion) to systematically traverse all vertices of a graph.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Labs and Assignments – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. *Introduction to Algorithms* (4th ed.). Cambridge, MA: MIT Press, 2022.
- Weiss, M. A. *Data Structures and Algorithm Analysis in C* (2nd ed.). Boston, MA: Addison-Wesley, 1997.
- Srivastava, S. K., & Srivastava, D. *Data Structures Through C in Depth* (2nd Revised & Updated ed.). New Delhi: BPB Publications, 2011.
- Karumanchi, N. *Data Structures and Algorithms Made Easy: Data Structure and Algorithmic Puzzles* (2nd ed.). Hyderabad: CareerMonk Publications, 2010.
- Goodrich, M. T., & Tamassia, R. *Data Structures and Algorithms in C* (2nd ed.). Hoboken, NJ: Wiley, 2015.
- Aho, A. V., Hopcroft, J. E., & Ullman, J. D. *Data Structures and Algorithms*. Reading, MA: Addison-Wesley, 1983.

- Langsam, Y., Augenstein, M. J., & Tanenbaum, A. M. *Data Structures Using C and C++* (2nd ed.). Noida: Pearson Education, 2015.
- Balagurusamy, E. *Data Structures Using C*. New Delhi: McGraw Hill Education, 2017.
- Kanetkar, Y. P. *Data Structures Through C* (2nd ed.). New Delhi: BPB Publications, 2017.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD2002
4. Course Title	Discrete Mathematics
5. Credits (L:T:P)	4 (3:1:0)
6. Contact Hours (L:T:P)	3:1:0
7. Prerequisites	CSD1001
8. Major Core for	CSE
9. Major Elective for	None

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This course is designed to equip students with knowledge about the fundamentals of Discrete Mathematics by being able to use mathematically correct terminology and notation, construct correct direct and indirect proofs, use counterexamples and apply logical reasoning to solve a variety of problems, and apply combinatorics and graph theory to many real-world situations.

11. Course Objectives

The objectives of this course are as follows:

- Develop a strong foundation in discrete mathematical structures, including logic, sets, relations, functions, and proof techniques, to support rigorous reasoning and problem solving in computer science.
- Apply principles of combinatorics, number theory, and abstract algebra to analyze computational problems, design efficient solutions, and understand their applications.
- Model and solve real-world and computer science problems using graph theory, trees, partial orders, and related discrete structures, enabling the analysis and design of complex systems and networks.

12. Learning Outcomes

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

- Understand an argument using logical notation and determine if the argument is or is not valid.
- Apply the basic principles of sets and operations in sets.
- Analyse the relations and functions and be able to determine their properties.
- Evaluate/Demonstrate different traversal methods for trees and graphs.
- Create/Model problems in Computer Science using graphs and trees.
- Understand fundamentals of abstract algebra, such as groups, rings, and fields, which are important in advanced research domains.

13. Curriculum Content

- **Set Theory:** Operations on sets, Cartesian product of sets, General proofs of some fundamental identities on sets.
- **Propositional Logic and Predicate Calculus:** Propositions and Logical operations, Conditional statements, First order predicate, Well-formed formula of predicate, quantifiers, Inference theory of predicate logic, Methods of proof, Mathematical induction
- **Relations and Functions:** Properties of Binary relations, Closure of relation, Warshall's Algorithm, Functions, Compositions of functions, Invertible functions, recursively defined functions, Growth of Functions, Recurrence relations
- **Relations and Digraphs:** Relations and Directed Graphs, Special properties of binary relations, Connection between relations and some data structures, Equivalence relation and partition, Partial order relations, Partial order sets: Definition, Partial order sets, Combination of partial order sets, Hasse diagram, Lattices: Definition, Properties of lattices – Bounded, Complemented and Complete lattice
- **Combinatorics:** Basics of Counting, Permutation & Combination, Derangement, Pigeonhole Principle, Binomial and Multinomial Theorems, Principle of Inclusion and Exclusion

- **Number Theory:** Modular Arithmetic, probability, application to cryptography, Algorithm complexity.
- **Graph Theory:** Basic of Graph, Euler paths and circuits, Hamiltonian paths and circuits, Isomorphic graphs, Connected Graph, Trees: Labelled trees, Tree searching, Undirected trees, Isomorphic trees, Minimal spanning trees, Prim's algorithm, Graphs: Planar Graph, Matching problems, Colouring graphs, Transport networks

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Tutorials/Assignments – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Kenneth H. Rosen, *Discrete Mathematics and Its Applications*, 8th Edition, McGraw-Hill Education, 2019.
- C. L. Liu and D. P. Mohapatra, *Elements of Discrete Mathematics: A Computer-Oriented Approach*, 4th Edition, McGraw-Hill Education, 2017.
- Susanna S. Epp, *Discrete Mathematics with Applications*, 5th Edition, Cengage Learning, 2018.
- Richard Johnsonbaugh, *Discrete Mathematics*, 8th Edition, Pearson Education, 2017.
- Norman L. Biggs, *Discrete Mathematics*, 2nd Edition, Oxford University Press, 2003.
- J. P. Tremblay and R. Manohar, *Discrete Mathematical Structures with Applications to Computer Science*, McGraw-Hill Education, Reprint Edition, 2017.
- Kolman, Busby and Ross, *Discrete Mathematical Structures*, 6th Edition, Pearson Education, 2018.
- Seymour Lipschutz and Marc Lars Lipson, *Schaum's Outline of Discrete Mathematics*, 3rd Edition, McGraw-Hill Education, 2009.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD2004
4. Course Title	Object Oriented Programming
5. Credits (L:T:P)	4 (3:0:1)
6. Contact Hours (L:T:P)	3:0:2
7. Prerequisites	CSD1001
8. Major Core for	CSE
9. Major Elective for	None

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

This course introduces the concepts and implementation of the Object-Oriented Programming (OOP) paradigm using the Java programming language. Topics include objects and classes, inheritance and polymorphism, string handling, GUI development, event-driven programming, exception handling, multithreading, JDBC, and the Java Collections Framework.

11. Course Objectives

This course enables students:

- to understand programming paradigms
- to learn the principles of object-oriented programming using Java,
- to apply object-oriented design and implementation techniques to develop software solutions.

12. Learning Outcomes

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

- Develop and execute Java programs using appropriate input/output mechanisms.
- Apply object-oriented programming concepts including classes, objects, encapsulation, and abstraction.
- Design and implement abstract classes and interfaces.
- Apply inheritance and polymorphism in software development.
- Manipulate and process strings using Java string-handling mechanisms.
- Design graphical user interfaces using Java GUI frameworks.
- Develop event-driven applications using appropriate event-handling techniques.
- Implement robust exception handling mechanisms in Java applications.
- Develop database applications using JDBC.
- Utilize Java Collection Framework components to manage and process data efficiently.

13. Curriculum Content

- **Object oriented thinking:** Need for oop paradigm, A way of viewing world – methods, classes and instances, class hierarchies (Inheritance), etc.
- **Java Basics:** concepts of classes, objects, constructors, methods, access control, this keyword, overloading methods and constructors, parameter passing, exploring string class.
- **Inheritance:** Hierarchical abstractions, forms of inheritance- specialization, benefits of inheritance. Member access rules, super uses, using final with inheritance, polymorphism- method overriding, abstract classes, the Object class.
- **Packages and Interfaces:** Defining, Creating and Accessing a Package, importing packages, differences between classes and interfaces, defining an interface, implementing interface, applying and extending interfaces.
- **JavaFX** – Introduction, difference between JavaFX, Swing, and AWT, exploring Color class, Font class, Image class, ImageView class, Pane class - StackPane, FlowPane, GridPane, BorderPane, HBox, and VBox, etc.

- **Event Handling:** Events, Delegation event model, handling mouse and keyboard events, Adapter classes, user interface components- labels, button, text components, check box, check box groups, choices, layout manager – layout manager types – border, grid, flow, etc.
- **Exception handling:** Concepts of exception handling, benefits of exception handling, usage of try, catch, throw, throws and finally, built in exceptions, creating own exception sub classes.
- **Java Collection Framework:** Interfaces: Set, List, Queue, Deque and classes: Array List, Vector, Linked List, Priority Queue, Hash Set, etc

List of Labs

- Elementary Programming, Selections, Loops
- Methods, Single-Dimensional Arrays, Multidimensional Arrays
- Object-oriented programming: Classes, Objects
- Inheritance and Polymorphism
- String handling
- Abstract Classes and Interfaces
- Graphical User Interface
- Event-Driven Programming
- Exception Handling
- Java Collections Framework

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Labs – 20%
- Quiz and Assignments - 20%
- Class participation – 10%

15. Textbooks and References:

- Liang, Y. Daniel. Introduction to Java Programming and Data Structures, Comprehensive Version. 10th Edition. Pearson Education, 2015.
- Schildt, Herbert. Java: The Complete Reference. 11th Edition. McGraw-Hill Education, 2018.
- Deitel, Paul J., and Harvey M. Deitel. Java: How to Program. 11th Edition. Pearson Education, 2018. ISBN-10: 0134743350; ISBN-13: 9780134743356.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD2003
4. Course Title	Computer Organization
5. Credits (L:T:P)	5(3:1:1)
6. Contact Hours (L:T:P)	3:1:2
7. Prerequisites	CSD1001
8. Major Core for	CSE
9. Major Elective for	NA

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

This course provides a comprehensive understanding of the fundamental components that make up a computer system. It begins with the basics of Boolean algebra, logic gates, and the design of both combinational and sequential circuits, forming the foundation for digital logic. Students will explore how computers interpret and execute instructions, delving into machine language, instruction sets, and their processing. The course also covers key elements of computer architecture, including the control unit, arithmetic logic unit (ALU), registers, and pipelining, offering insight into how computers process information efficiently. Different types of memory systems, their organization, and data management techniques will be discussed to understand how memory plays a critical role in computing.

11. Course Objectives

This course will enable the students to understand the internal organization of a computer, and analyze its different modules in terms of structure & function. Additionally, the course examines how computers interact with external devices through input/output systems, interfaces, and various communication techniques. Students will gain practical knowledge of interfacing with external devices by learning how hardware and software enable external connections. Overall, this course offers an in-depth look into the internal workings of a computer system and its interaction with the outside world.

12. Learning Outcomes

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

- Understand the evolution and major architectural milestones and their impact on modern computing.
- Analyze the performance of computer systems using appropriate quantitative metrics.
- Apply computer arithmetic principles for representing and performing operations on data.
- Program using assembly language, with emphasis on MIPS programming.
- Analyze the structure and functionality of the CPU, including datapath design, control unit operation, instruction execution, and register organization.
- Design and model digital circuits using Verilog for basic computer organization components.
- Understand the concept of instruction pipelining.
- Analyze memory organization and hierarchy, including cache memory, main memory, secondary storage, and their impact on overall system performance.
- Understand and analyze input/output organization.

13. Curriculum Content

- **Definition of computer organization, computer architecture, and their relationship:** Basic Components: Input devices, output devices, CPU, memory, storage devices. Binary, octal, decimal, and hexadecimal systems; conversions between all four — binary $\leftrightarrow$ decimal, octal $\leftrightarrow$ binary, hex $\leftrightarrow$ binary, hex $\leftrightarrow$ decimal. IEEE 754 single and double precision; sign bit, biased exponent, mantissa/significand; normalized form.

- **Boolean Algebra:** Laws & Theorems: AND, OR, NOT operations; commutative, associative, distributive laws; De Morgan's theorems; duality principle. Logic Gates & Boolean Function Implementation: AND, OR, NOT, NAND, NOR, XOR, XNOR gates; truth table to gate circuit; gate-level implementations of Boolean functions. Karnaugh Maps & Boolean Minimization: Minterms and maxterms; K-maps; grouping 1s (SOP) and 0s (POS); don't-care conditions; minimal expressions. Adder Circuits, Decoders & Encoders, Multiplexers & Demultiplexers, Latches, Flip-Flops, Shift Registers, Counters, Integrated Review & Problem Solving: Number conversion edge cases; K-map with don't-cares; adder/decoder integration; flip-flop conversion problems; counter + register chain; exam-style MCQs. Programmable Logic Devices: PLA, PAL, CPLD, FPGA. Modelling combinational and sequential circuits using Verilog
- **Introduction to Computer Architecture:** VonNeumann Architecture, Classes of Computers, Moore's Law, Amdahl's Law, Overview of IC Manufacturing, Processor power consumption and performance trends Processor Performance measurement and Benchmarking ISA Fundamentals — Instruction Format & Addressing Modes: What is an ISA; instruction format; opcode and operands; instruction types; addressing modes. RISC vs. CISC & Registers, MIPS as a Representative RISC ISA Introduction to MIPS Assembly Language Programming
- **Processor:** Data and Control Path Introduction to the basic pipelining concepts Computer Arithmetic, Basic Adder, Multiplier, Fast Adder, ALU Design & CPU Pipelining.
- **Memory Systems:** Why multiple levels; registers, cache (L1/L2/L3), main memory, secondary storage; locality of reference; cost-speed-capacity tradeoff Cache Memory — Mapping Techniques: Cache hit and miss; hit ratio and average access time; direct mapping; fully associative mapping; set-associative mapping; worked numerical examples for all three techniques. Cache Replacement Algorithms & Write Policies: LRU; FIFO; Random replacement; optimal algorithm; write-through vs. write-back; write allocate vs. no-write allocate; dirty bit; cache coherence basics.
- **I/O Devices, Interfaces & Techniques:** Keyboard, mouse, monitor, disk drives; parallel & serial interfaces (UART, USB); programmed I/O; interrupt-driven I/O; DMA — controller registers, cycle stealing vs. burst mode. memory-mapped I/O vs. port-mapped I/O; bus timing; device driver structure; OS interaction; I/O request queues; blocking vs. non-blocking I/O. Case Study: 8085/8086 Processors, SoCs and Microcontrollers, GPUs

List of Labs

- **Combinational Circuits for Binary Arithmetic:** The objective of this experiment is to understand and implement basic combinational arithmetic circuits using CMOS logic gates. Students will verify De Morgan's Laws and design Half Adder and Full Adder circuits using standard ICs on a breadboard.
- **Binary Arithmetic Circuits – Multi-bit Addition and Subtraction:** The objective of this experiment is to design and verify multi-bit combinational arithmetic circuits. Students will first implement a Half Subtractor as a prerequisite and then design a 2-bit Full Adder and a 2-bit Full Subtractor using basic logic gates.
- **Verification of 4-bit Binary to Gray and Gray to Binary Code Conversion using Universal Gates:** The objective of this experiment is to understand code conversion techniques in digital systems. Students will design and verify 4-bit Binary to Gray code conversion and Gray to Binary code conversion using universal gates such as NAND and NOR gates.
- **Implementation of 2:1 Multiplexer using Universal Gates:** The objective of this experiment is to understand the working of data selection circuits. Students will design and implement a 2:1 multiplexer using universal gates and verify its truth table on a breadboard.
- **Implementation of 4:1 Multiplexer using Logic Gates:** The objective of this experiment is to understand the design of multi-input data selection circuits. Students will implement a 4:1 multiplexer using basic logic gates and verify its operation for all input and select-line combinations.
- **Implementation of 1:4 Demultiplexer using Logic Gates:** The objective of this experiment is to understand the working of data distribution circuits. Students will implement a 1:4 demultiplexer using logic gates and verify the output corresponding to different select-line combinations.

- **Implementation and Verification of Encoder using Logic Gates:** The objective of this experiment is to understand the design of combinational encoding circuits. Students will implement an encoder using basic logic gates and verify its truth table.
- **Implementation and Verification of Decoder using Logic Gates:** The objective of this experiment is to understand the design and operation of combinational decoding circuits. Students will implement a decoder using logic gates and verify its truth table.
- **Verification of Flip-Flops using NAND/NOR Gates:** The objective of this experiment is to understand the operation of basic sequential circuits. Students will verify the state tables of S-R, J-K, D, and T flip-flops using NAND/NOR gates.
- **Design and Verification of 4-bit Synchronous Counter:** The objective of this experiment is to understand the design of synchronous sequential circuits. Students will design and verify a 4-bit synchronous counter using flip-flops and logic gates.
- **Design and Verification of 4-bit Asynchronous Counter:** The objective of this experiment is to understand ripple counter operation. Students will design and verify a 4-bit asynchronous counter using flip-flops and observe the counting sequence.
- **Implementation of 3-Input, 2-Output Binary Adder using Verilog:** The objective of this experiment is to introduce hardware description language-based digital design. Students will write Verilog code for a 3-input, 2-output binary adder, simulate the design, and verify its functionality using appropriate test cases.
- **MIPS Programming:** MIPS programming involving register usage, branching, loops, memory access, and array handling.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Labs – 20%
- Quiz and Assignments - 20%
- Class participation – 10%

15. Textbooks and References

- Mano, M. M., Digital Logic and Computer Design, Pearson India, 2017.
- Patterson, D. A., and Hennessy, J. L., Computer Organization and Design MIPS Edition: The Hardware/Software Interface, 6th ed., Morgan Kaufmann/Elsevier, 2020.
- Sarangi, S. R., Basic Computer Architecture, Version 3.09, 2026.
- Hennessy, J. L., Patterson, D. A., and Kozyrakis, C., Computer Architecture: A Quantitative Approach, 7th ed., Morgan Kaufmann/Elsevier, 2025.
- Carter, N. P., adapted by Raj Kamal, Computer Architecture and Organization, 2nd ed., Schaum's Outline Series, McGraw Hill Education, 2017.
- Hamacher, C., Vranesic, Z., Zaky, S., and Manjikian, N., Computer Organization and Embedded Systems, 6th ed., McGraw Hill Education, 2023.

Operating Systems

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD2005
4. Course Title	Operating Systems
5. Credits (L:T:P)	4 (3:0:1)
6. Contact Hours (L:T:P)	3:0:2
7. Prerequisites	CSD2001, CSD2003
8. Major Core for	CSE
9. Major Elective for	None

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

The course provides a comprehensive exploration of the fundamentals of Operating System, mechanism, implementations, and its practical aspects. It explores how operating systems act as an intermediary between computer hardware and users, ensuring efficient and secure resource management. The core topics of the course include Linux operating system, process states & life cycle, process scheduling, Thread creation, synchronization, concurrency, multi-threaded programming, memory management, file system concepts, secondary storage structure, I/O systems, Security and Protection, Distributed OS, and influential Operating Systems. The course aims to build the foundation of Computer Science, and it will play an essential role in all the upcoming courses.

11. Course Objectives

The purpose of this course is to provide the students with solid foundations of operating system as follows:

- To develop the student's basic understanding of Linux commands and its implementation.
- To provide the students with in-depth knowledge about Operating System abstraction and mechanisms.
- To introduce the students with the concept of process synchronization, deadlock resolution, and efficient CPU scheduling using various algorithms.
- Practically create multiple threads using pthreads, concurrency, and multi-threaded programming to exploits the multi-core hardware architecture.
- To introduce the students with memory management, filesystems, and I/O systems.
- Enable students to implement, simulate, and analyze various components of operating systems, build a mindset among students about the core of computer science and its working principle.

12. Learning Outcomes

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

- Create bash files using various Linux commands and have full control over the operating system that they are using.
- Create an efficient program using multi-threading approach and understand the concept of process synchronization and CPU scheduling.
- Evaluate issues arising from concurrent process execution, such as synchronization, communication, and resource sharing. Design solutions to ensure correct and efficient concurrent execution.
- Understand memory and virtual memory management strategies which are quite relevant given the program size are increasing day by day.
- Implement their own custom file systems as per their requirements.

13. Curriculum Content

- Overview of Operating System: Introduction, System Structure
- Process Management: Process State and lifecycle, Process Scheduling Algorithms, Thread Creation
- Process Synchronization: Synchronization tools, Synchronization Example, pthread: Multi-threaded Programming, Deadlocks

- Memory Management: Memory Management Strategies, Page Replacement Algorithms, Virtual Memory Management
- Storage Management & File System: File System, Implementing File Systems, Secondary Storage Structure, Disk scheduling algorithms, I/O Systems
- Protection & Security: System Protection, System Security, UNIX/LINUX and Windows as an example.
- Special Purpose System: Real Time Systems, Distributed Systems

List of Labs

- **Lab Assignment - 1 Introduction to Linux Commands:** The goal of this assignment is to help students develop practical skills in exploring Linux system structures, executing essential shell operations, and extracting system and process information using both commands and C programs. It enables students to understand how the operating system exposes low-level details through interfaces like the /proc filesystem and observe real process behaviour through monitoring tools.
- **Lab Assignment 2 - Process control system calls:** The goal of this lab is to deepen the understanding of students on various concepts of process management in Linux, specifically on system calls such as OPEN(), READ(), WRITE(), CLOSE(), FORK(), EXEC(), WAIT(), and EXIT() along with zombie and orphan states.
- **Lab Assignment 3 - CPU Scheduling Algorithms:** This lab aims to deepen your understanding of CPU scheduling algorithms used in operating systems, such as First Come First Serve (FCFS), Round Robin (RR), Priority, and Shortest Remaining Time First (SRTF). Through this assignment, students will simulate process scheduling, analyze scheduling behaviour, and compute key performance metrics such as waiting time, completion time, and turnaround time. The objective is to understand how different scheduling strategies impact system performance and process execution.
- **Lab Assignment 4 - CPU Scheduling (Mixed burst time):** The goal of this assignment is to understand and analyse the behavior of CPU scheduling in an operating system when multiple processes compete for the CPU and perform both CPU and I/O operations. Using priority-based scheduling, the assignment focuses on evaluating key performance metrics such as waiting time, turnaround time, response time, CPU utilisation, and CPU idleness.
- **Lab Assignment 5 - Multithreading:** The goal of this assignment is to explore the fundamentals of multithreading using pthread in C/C++. These tasks will help students to understand how parallel processing can improve performance and where synchronisation mechanisms are required to avoid unexpected issues like race conditions.
- **Lab assignment 6 - Implementing TAS, CAS and bounded waiting CAS mutual exclusion algorithms:** The goal of this assignment is to implement TAS, CAS and Bounded Waiting with CAS mutual exclusion (ME) algorithms studied in the class. Implement these algorithms in C++/C.
- **Lab Assignment 7 - Classical IPC problems:** The goal of this assignment is to understand the concept of process or thread synchronization in Operating Systems by implementing real-world concurrency problems. Students will simulate resource-sharing scenarios such as an office system where employees require both a printer and a scanner (Dining Philosophers Problem) and a college mess order system where chefs produce orders and students consume them (Producer-Consumer Problem). Through these implementations, students will learn how multiple threads coordinate while accessing shared resources.
- **Lab Assignment 8 - Implementing Banker's & Memory Management Unit (MMU):** The goal of this assignment is to implement deadlock avoidance Banker's algorithm & address Translation via MMU unit using C/C++ studied in the class.
- **Lab Assignment 9 - Page replacement policies:** The goal of this assignment is to simulate and compare the performance of different page replacement algorithms studied in the class, such as First-In First-Out (FIFO), Least Recently Used (LRU), and Optimal, used in Operating Systems for Virtual Memory Management.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Labs/Assignment – 30%
- Quiz/Class participation – 20%

15. Textbooks and References:

- Abraham Silberschatz, Peter B. Galvin, and Greg Gagne, *Operating System Concepts*, 10th Edition, John Wiley & Sons, 2018.
- Andrew S. Tanenbaum and Herbert Bos, *Modern Operating Systems*, 5th Edition, Pearson Education, 2022.
- Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*, Arpaci-Dusseau Books, 2018.
- William Stallings, *Operating Systems: Internals and Design Principles*, 10th Edition, Pearson Education, 2018.
- D. M. Dhamdhere, *Operating Systems: A Concept-Based Approach*, 3rd Edition, McGraw-Hill Education, 2017.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD2006
4. Course Title	Introduction to Probability and Statistics
5. Credits (L:T:P)	4 (3:1:0)
6. Contact Hours (L:T:P)	3:1:0
7. Prerequisites	NA
8. Major Core for	CSE
9. Major Elective for	None

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

Uncertainty is ubiquitous in nature and probability theory provides a rational description and quantification of uncertainty. There is a growing realization that randomness is an essential component in modelling and analysis of a variety of systems. Probability has become an important conceptual framework for computer science, engineering, physical and biological sciences. Several problems in computer engineering and other disciplines arise, which require probabilistic modelling. The complete specification of the model requires statistical tools for analysis of data and inference.

11. Course Objectives

Upon successful completion of this course, students will be able to:

- Apply the principles of probability theory to model uncertainty and analyze random phenomena in engineering and computing systems.
- Analyze discrete and continuous probability distributions and use them to solve practical problems involving randomness and stochastic processes.
- Apply statistical methods, estimation techniques, hypothesis testing, and regression analysis for data-driven decision making and inference.

12. Learning Outcomes

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

- Apply probability axioms, conditional probability, independence, and Bayes' theorem to solve probabilistic problems.
- Analyze discrete and continuous random variables using probability distributions, expectation, variance, and moments.
- Apply joint, conditional, and marginal distributions to model relationships among random variables.
- Utilize the Law of Large Numbers and Central Limit Theorem in probabilistic analysis and simulation.
- Generate random numbers and perform simulation-based experiments for stochastic systems.
- Compute descriptive statistical measures and perform exploratory data analysis on real-world datasets.
- Apply parameter estimation, confidence interval estimation, and hypothesis testing techniques to draw statistical inferences.
- Develop regression models and interpret statistical relationships for predictive analysis.

13. Curriculum Content

- Axioms of probability, conditional probability and independence, Bayes theorem.
- Random variables, distribution function, discrete random variable, expectation, variance, Bernoulli and Binomial random variable, Poisson random variable, Negative binomial random variable, Geometric random variable.
- Continuous random variable: expectation of random variable, variance, distribution: Uniform, Normal, Exponential, Gamma and Cauchy, computing probabilities by conditioning, characteristic and moment generating function
- Jointly distributed random variables, Independent random variable, sum of independent random variable, conditional distribution, joint probability distribution, covariance, correlation coefficient.

- Inequalities, Law of Large Numbers, Central Limit Theorem
- Generation of random numbers. Elements of information theory: entropy as a measure of randomness.
- Exploratory data analysis, types of data, frequency tables, descriptive measures, variability measures.
- Elements of statistical inference, properties of estimators, estimation of parameters, interval estimation, hypothesis testing, Maximum Likelihood Estimation (MLE) and Maximum A Posteriori (MAP) Estimation, linear statistical model, least squares method, regression function and analysis.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Tutorial/Assignments – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Hossein Pishro-Nik, Introduction to Probability, Statistics, and Random Processes, Kappa Research LLC, 2014.
- Mor Harchol-Balter, Introduction to Probability for Computing, Cambridge University Press, 2023.
- Sheldon M. Ross, Introduction to Probability and Statistics for Engineers and Scientists, 5th Edition, Academic Press (Elsevier), 2014.
- Dimitri P. Bertsekas and John N. Tsitsiklis, Introduction to Probability, 2nd Edition, Athena Scientific, 2008.
- Sheldon M. Ross, Introduction to Probability Models, 12th Edition, Academic Press, 2019.
- Larry Wasserman, All of Statistics: A Concise Course in Statistical Inference, Springer, 2004.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD2007
4. Course Title	Applied Linear Algebra
5. Credits (L:T:P)	4(3:1:0)
6. Contact Hours (L:T:P)	3:1:0
7. Prerequisites	NA
8. Major Core for	CSE
9. Major Elective for	NA

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This course provides a comprehensive introduction to linear algebra and optimization techniques, with a focus on vectors, matrices, and least squares methods. The course is designed to equip students with a strong foundation in mathematical modelling and analysis, preparing them for advanced studies in various fields.

11. Course Objectives

The aim of the course is to equip students with a broad understanding of linear algebra and optimization techniques, and to prepare them for a wide range of careers that require strong mathematical skills.

12. Learning Outcomes

- As part of this course students will learn to apply linear algebra and optimization techniques to solve systems of equations, find the best fit for data, and optimize functions.
- Further, this course provides a foundation for further studies in fields such as data science, machine learning, computer science, engineering, economics, and statistics.

13. Curriculum Content

- **Systems of Linear Equations and Matrix Computations:** Representation of linear systems, Gaussian elimination and row reduction, Echelon forms and reduced row echelon form, Matrix multiplication and inverse matrices, LU decomposition and permutation matrices, Computational aspects of solving linear systems, Tridiagonal and finite difference matrices.
- **Determinants and Matrix Properties:** Properties of determinants, Cofactor expansion and minors, Product rule for determinants, Singular and nonsingular matrices, Geometric interpretation of determinants, Applications of determinants.
- **Vector Spaces and Linear Independence:** Vector spaces and subspaces, Span and linear combinations
- Linear independence and dependence, Basis and dimension, Rank and rank-deficient systems, Pivot variables and free variables, Rank-nullity theorem.
- **Fundamental Subspaces:** Column space, Row space, Nullspace, Left nullspace, Orthogonality relations among fundamental subspaces, Rank equality, and geometric interpretations.
- **Orthogonality and Least Squares:** Dot product and norms, Orthogonal vectors and orthogonal, Complements, Orthogonal projections, Projection matrices, Least-squares approximation, Normal equations, Residual vectors and geometric interpretation, Numerical aspects and stability.
- **Orthogonal Bases and Matrix Factorizations:** Orthonormal vectors and bases, Gram-Schmidt orthogonalization, Orthogonal matrices, QR decomposition, Least-squares solution using QR factorization
- Applications of QR decomposition.
- **Eigenvalues and Eigenvectors:** Eigenvalues and eigenvectors, Characteristic polynomial, Eigenspaces, Similarity transformations, Matrix diagonalization, Matrix powers using diagonalization, Linear independence of eigenvectors, Applications of eigenvalue decomposition.

- **Symmetric and Positive Definite Matrices:** Symmetric matrices, Spectral decomposition, Quadratic forms, Positive definite matrices, Tests for positive definiteness, Positive pivots and principal minors
- Sylvester's criterion, Energy interpretation and applications,
- **Complex Matrices and Hermitian Matrices:** Complex vectors and inner products, Hermitian matrices
- Orthogonality of complex eigenvectors, Spectral theorem for Hermitian matrices, Positive definite, Hermitian matrices.
- **Fourier Matrices and Structured Matrices:** Roots of unity, Fourier vectors and Fourier matrices, Orthogonality of Fourier basis, Cyclic permutation matrices, Circulant matrices, Diagonalization using Fourier matrices, Introduction to Fast Fourier Transform (FFT).
- **Singular Value Decomposition:** Singular values and singular vectors, Singular Value Decomposition (SVD)
- Geometric interpretation of SVD, Matrix rank and low-rank approximation, Rayleigh quotient, Variational characterization of singular values, Principal singular vectors, Orthogonality constraints.
- **Data Science and Machine Learning Foundations:** Vector and matrix representation of datasets, Feature matrices and data transformations, Linear regression and least-squares estimation, Covariance matrices and correlation analysis, Principal Component Analysis (PCA), Dimensionality reduction techniques, Matrix factorization methods, Low-rank approximations, Numerical stability in high-dimensional problems, Foundations of optimization in machine learning, Matrix computations in deep learning, Applications in computer vision, signal processing, and data analytics

14. Assessment Scheme

- Assignments – 25%
- Quiz – 25%
- Examination – 50%

15. Textbooks and References

- Strang, Gilbert. Introduction to Linear Algebra. 6th Edition. Wellesley-Cambridge Press, 2023.
- Boyd, Stephen, and Lieven Vandenberghe. Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares. Cambridge University Press, 2018.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4801
4. Course Title	Introduction to Database Systems
5. Credits (L:T:P)	4 (3:0:1)
6. Contact Hours (L:T:P)	3:0:2
7. Prerequisites	CSD2001
8. Major Core for	None (Experiential Learning for CSE)
9. Major Elective for	None

PART B: OBJECTIVES AND PRACTICES**10. Course Summary**

This course is designed to equip students with a comprehensive understanding of the fundamentals of Database Management Systems (DBMS). In addition to theoretical concepts, the course includes a significant lab component, where students will gain practical experience with SQL and procedural SQL. Through hands-on exercises, they will learn the process of real-world database design, from requirement analysis to actual development. Key topics covered include an introduction to DBMS, the Entity-Relationship (ER) model, relational data models, relational algebra, normalization, indexing, transaction processing, concurrency control, and recovery mechanisms.

11. Course Objectives

To enable students:

- Understand key concepts, architecture, and data models in database management systems.
- Develop proficiency in SQL for effective data manipulation and retrieval.
- Learn to design and normalize database schemas using ER diagrams.
- Explore techniques for ensuring data integrity.
- Apply database concepts to practical scenarios through projects and case studies.

12. Learning Outcomes

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

- Understand database architecture and database system environment.
- Understand data models and schemas.
- Write database requirements analysis for building a DBMS for a given organization or case study.
- Design ER model or EER to satisfy database requirements.
- Design a relational database using SQL, while satisfying relational model constraints.
- Understand the importance of indexing and design indexes based on application requirements.
- Understand transaction processing, concurrency control and recovery techniques.
- Develop the database designs for a range of simple applications using open-source software.

13. Curriculum Content

- **Introduction to Databases:** Overview of database systems, DBMS, and database applications, data abstraction, data independence, and database architecture.
- **Understanding Data Models through Real-life Examples:** The importance of data models, basics of the ER model, entities, attributes, relationships, ER diagram, constraints, ERD issues, weak entity sets, and EER model.
- **Case Studies:** Preparing Conceptual design/ER Model for a company, conference review System, University Management System, etc.
- **Database Design:** Relational schemas, relational database model, logical view of data, keys, integrity rules. Relational Database design: features of good relational database design, Domain and data dependency, functional dependency, Armstrong's axioms, normal forms, dependency preservation, lossless design.

- **Hands-on:** Converting ER Model to Relational model for the case studies discussed, identifying FDs and checking for normal forms.
- **Relational Algebra:** Introduction, selection and projection, set operations, renaming, joins, division operators, grouping and ungrouping, relational comparison.
- **Hands-on:** Understanding mapping of Relational Algebra to SQL queries.
- **Indexing:** Primary index, secondary, clustered index, multi-level index, B-tree index
- **Hands-on:** Creating various kind of indexes and understanding their need.
- **Constraints, SQL, and Views:** constraints, types of constraints, Integrity constraints. SQL: data definition language, data manipulation language, aggregate function, null values, nested subqueries, joined relations, procedural SQL and triggers. Views: Introduction to views, data independence, updates on views, comparison between tables and views. **Hands-on:** Lab sessions for understanding and learning Constraints, SQL and Views.
- **Transaction management, Concurrency control, and Recovery:** Transaction management: ACID properties, serializability and concurrency control, lock based concurrency control (2PL, Deadlocks), time stamping methods, multi-version and optimistic concurrency control schemes, database recovery management techniques. **Hands-on:** Lab sessions to understand the various properties of Transactions (ACID properties), Concurrency and Recovery techniques (rollback, checkpoint, commit).
- **Advanced topics:** Introduction to some advanced topics such as Object-oriented and object relational databases, logical databases, web databases, distributed databases and big data.
- **Project:** The scope of work includes the development and integration of the frontend and backend for the specified case study.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Labs/Assignment – 30%
- Quiz/Class participation – 20%

15. Textbook and References

- Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems (7th ed.). Pearson Education.
- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). Database System Concepts (7th ed.). McGraw-Hill Education.
- Ramakrishnan, R., & Gehrke, J. (2002). Database Management Systems (3rd ed.). McGraw-Hill Education.
- Ullman, J. D. (1988). Principles of Database and Knowledge-Base Systems, Volume I. Galgotia Publications.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4001
4. Course Title	Computer Networks
5. Credits (L:T:P)	4(3:0:1)
6. Contact Hours (L:T:P)	3:0:2
7. Prerequisites	CSD2005
8. Major Core for	CSE
9. Major Elective for	NA

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

This course will provide a basis in understanding different kinds of computer networks, particularly wired networks. Starting from the very basic of TCP/IP protocol suite, protocols operating from different layers will be discussed. Apart from theoretical and mathematical understanding of different network protocols, basic applications of socket programming, OMNET++ simulator and Wireshark will be introduced. Fundamental differences in network protocol design between wired and wireless networks will also be discussed.

11. Course Objectives

- To understand the TCP/IP protocol suite.
- To understand different networking protocols operating from different layers of TCP/IP protocol suite.
- To explore the basic applications of socket programming, Wireshark and OMNET++.
- To understand the medium access control mechanisms of wireless networks (only WIFI will be covered).

12. Learning Outcomes

- **Understand** the functions of different protocols of TCP/IP protocol suite.
- **Apply** socket programming in different networking problems.
- **Analyse** network performance based on Wireshark.
- **Understand** the operation of different network protocol using OMNET++ simulator.
- **Design** new network protocols for the forthcoming networking scenarios.

13. Curriculum Content

- Introduction to Computer Networking: Types of Networks (LAN, MAN, WAN, PAN), Network Standards, Protocols, Protocol Stack, OSI vs TCP/IP protocol suite, Physical Layer: Analog vs Digital transmissions, Guided and Unguided Media, Internetworking Devices: Hubs, Switches, Routers, Gateways, Multiplexing: FDM, TDM, Statistical TDM, Circuit Switching, Packet Switching, Message Switching, Synchronous vs Asynchronous Transmission.
- Data Link Layer: Data Link Layer Services, Link Layer Addressing, Error Detection Techniques: Parity, CRC, Checksum, Forward Error Correction, Flow Control & ARQ Protocols: Stop & Wait, Go-back-N, Selective Repeat, HDLC and PPP Protocols, ARQ Performance in Error-Free & Noisy Channels, MAC Protocols: ALOHA (Pure & Slotted), CSMA, CSMA/CD, Ethernet: Frame Format and Working, Wireless MAC: CSMA/CA, RTS/CTS, Backoff Algorithm.
- Network Layer: Network Layer Design Issues, Routing Concepts: Forwarding vs Routing, Subnetting, Supernetting, CIDR Notation, IPv4 Addressing, ARP, RARP, Routing Algorithms: Distance Vector (with examples), Count-to-Infinity problem, Split Horizon, Link State Routing (with examples), OSPF, BGP, RIP.
- Transport Layer: Transport Layer Services, UDP: Header, Working, Use Cases, TCP: Header Format, Connection Establishment (3-Way Handshake), Flow Control: Sliding Window Mechanism, TCP Congestion Control: Tahoe, Reno, Nagle's Algorithm, Cumulative ACK, Duplicate ACK, Socket Programming.

- Application Layer: Application Layer Protocols: HTTP, FTP, TELNET, Electronic Mail (SMTP, POP, IMAP), DNS and SNMP, Client-Server Programming Concepts, Demonstration through Wireshark.
- Network Security: Introduction to Network Security, Working Principle of IPSec, SSL/TLS (basic idea), Physical layer security.

14. Assessment Scheme

- Midterm/Final: 50%
- Lab/Quiz/Assignments: 50%

15. Textbooks and References

- James F. Kurose and Keith W. Ross, *Computer Networking: A Top-Down Approach*, 8th Edition, Pearson, 2021.
- W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff, "UNIX Network Programming, Volume I: The Sockets Networking API", 3rd edition, 2003.
- Data and Computer Communications, William Stallings, PHI.
- Computer Networks, Andrew S. Tanenbaum, PHI
- Beginning Linux Programming by N. Matthew and R. Stones, 4th Edition.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4002
4. Course Title	Artificial Intelligence and Machine Learning
5. Credits (L:T:P)	4(3:0:1)
6. Contact Hours (L:T:P)	3:0:2
7. Prerequisites	CSD2007
8. Major Core for	CSE
9. Major Elective for	NA

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

The goal of the Artificial Intelligence (AI) course is to provide hands-on knowledge on applying popular AI techniques to handle data. The students of this course will be given opportunities to do one research project and a set of assignments. The course will bring the students towards research level. The course curriculum is designed to equip the students with the recent advances in AI.

11. Course Objectives

This course will enable the students to acquire a deep understanding on the recent techniques for AI. On successful completion of the course, students will be able to achieve the following:

- Understand the basic and traditional ideas of AI problems and algorithms.
- Implement different popular AI techniques.
- Have a deep knowledge about the recent advances of AI.
- Solve real life problems on AI.

12. Learning Outcomes

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

- Understand the basic and traditional ideas of AI problems.
- Implement different popular AI techniques, and apply them in real life problems.
- Have a deep knowledge about the recent advances of AI.
- Solve real life problems on AI.

13. Curriculum Content

- Introduction to Artificial Intelligence: Intelligent agents, rational agents, deductive and inductive approaches to AI, applications of AI.
- Search and Problem Solving: Search spaces, problem formulation, uninformed search algorithms, informed search algorithms, heuristic search, constraint satisfaction problems, genetic algorithms, Tabu search, simulated annealing.
- Logic and Rule-Based Methods: Knowledge representation, propositional logic, predicate logic, inference mechanisms, rule-based systems, forward and backward chaining.
- Probabilistic Reasoning: Probability fundamentals, Bayesian inference, conditional independence, Bayesian networks, reasoning under uncertainty.
- Machine Learning Fundamentals: Learning paradigms, data preprocessing, feature engineering, model evaluation, overfitting and regularization.
- Neural Networks: Artificial neurons, perceptrons, multi-layer neural networks, activation functions, loss functions, gradient descent, backpropagation.
- Classification and Clustering algorithms
- Deep Learning Architectures: Convolutional neural networks, recurrent neural networks, Long Short-Term Memory (LSTM) networks, introduction to generative models.
- Introduction to Reinforcement Learning
- Introduction to Agentic AI

- AI and ML using Python: Python-based AI ecosystem, machine learning libraries, neural network libraries, model development, training, evaluation, and applications.

List of Labs:

- Lab of search algorithms
- Lab on Bayesian networks
- Lab on learning fundamentals – understand overfitting
- Lab on Neural networks
- Labs on Deep Learning models
- Lab on clustering

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Labs/Assignments/Project – 30%
- Quiz/Class Participation – 20%

15. Textbooks and References

- Aggarwal, C. C. Artificial Intelligence: A Textbook. Springer International Publishing, Cham, Switzerland, 2021.
- Russell, S. J., & Norvig, P. Artificial Intelligence: A Modern Approach. 4th Edition. Pearson Education Limited, 2021.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4003
4. Course Title	Formal Languages and Automata Theory
5. Credits (L:T:P)	4 (3:1:0)
6. Contact Hours (L:T:P)	3:1:0
7. Prerequisites	CSD2001
8. Major Core for	CSE
9. Major Elective for	None

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

This course introduces the mathematical foundations of computation through the study of formal languages, automata, grammars, and computability. It covers the representation and analysis of languages using finite automata, regular expressions, context-free grammars, and pushdown automata. Students learn techniques for proving language properties using closure properties and pumping lemmas, converting between equivalent computational models, and classifying languages within the Chomsky hierarchy. The course also explores Turing machines, computability, decidability, and the fundamental limits of computation, providing a theoretical basis for compiler design, programming languages, artificial intelligence, and computer science research.

11. Course Objectives

- Develop a strong foundation in the theory of formal languages, automata, grammars, computability, decidability, and computational complexity.
- Introduce the mathematical models and algorithms used to describe and analyze formal languages and abstract machines.
- Enable students to understand the theoretical foundations of computer science from the perspective of formal languages and automata theory.
- Explain the role of formal languages and grammars in compiler design, language processing, and related areas of computer science.
- Familiarize students with the concepts of computability and complexity, enabling them to distinguish between solvable and unsolvable computational problems and to analyze their computational limits.
- Prepare students for advanced study in areas such as compiler construction, programming language theory, cryptography, artificial intelligence, and theoretical computer science.

12. Learning Outcomes

- Provide an introduction to some of the central ideas of theoretical computer science from the perspective of formal languages.
- Introduce the fundamental concepts of formal languages, grammars, and automata theory.
- Classify machines by their power to recognize languages.
- Employ finite state machines to solve problems in computing.
- Understand deterministic and non-deterministic machines.
- Understand the differences between decidability and undecidability.

13. Curriculum Content

- **Introduction:** Alphabet, languages and grammars, productions and derivation, Chomsky hierarchy of languages.
- **Regular languages and finite automata:** Regular expressions and languages, deterministic finite automata (DFA) and equivalence with regular expressions, nondeterministic finite automata (NFA) and equivalence with DFA, regular grammars and equivalence with finite automata, properties of regular languages, pumping lemma for regular languages, minimization of finite automata.

- **Context-free languages and pushdown automata:** Context-free grammars (CFG) and languages (CFL), Chomsky and Greibach normal forms, nondeterministic push-down automata (PDA) and equivalence with CFG, parse trees, ambiguity in CFG, pumping lemma for context-free languages, deterministic pushdown automata, closure properties of CFLs.
- **Context-sensitive languages:** Context-sensitive grammars (CSG) and languages, linear bounded automata and equivalence with CSG.
- **Turing machines:** The basic model for Turing machines (TM), Turing-recognizable (recursively enumerable) and Turing-decidable (recursive) languages and their closure properties, variants of Turing machines, nondeterministic TMs and equivalence with deterministic TMs, unrestricted grammars and equivalence with Turing machines, TMs as enumerators.
- **Undecidability:** Church-Turing thesis, universal Turing machine, the universal and diagonalization languages, reduction between languages and Rice's theorem, undecidable problems about languages.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Tutorial/Assignment – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Harry R Lewis and Christos H Papadimitriou, Elements of The Theory Of Computation, Second Edition, Prentice Hall, 1998.
- John E Hopcroft, Rajeev Motwani, Jeffrey D Ullman, Introduction To Automata Theory, Languages, And Computation, Third Edition, Prentice Hall, 2007.
- Michael Sipser, Introduction To The Theory Of Computation, Third Edition, Course Technology, 2005.
- John Martin, Introduction To Languages And The Theory Of Computation, Third Edition, Tata McGraw-Hill, 2003.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4004
4. Course Title	Advanced Algorithms
5. Credits (L:T:P)	4 (3:0:1)
6. Contact Hours (L:T:P)	3:0:2
7. Prerequisites	CSD2001
8. Major Core for	CSE
9. Major Elective for	None

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

The course discusses some of the important approaches to design and analysis of algorithms emphasizing methods useful in practice. Different algorithms for a given computational task are presented and their relative merits evaluated based on performance measures. The following important computational problems will be discussed: sorting, searching, elements of dynamic programming and greedy algorithms, advanced data structures, graph algorithms, string matching, and NP completeness.

11. Course Objectives

The course aims to teach design and analysis of algorithms emphasizing methods useful in practice.

12. Learning Outcomes

- Formulate algorithms for arithmetic and logical problems and compute the computational time complexity.
- Write an algorithm to implement conditional branching, iteration and recursion.
- Decompose a problem into functions and synthesize a complete program using divide and conquer approach.
- For a given algorithms analyze worst-case running times of algorithms based on asymptotic analysis and justify the correctness of algorithms.
- Describe the greedy paradigm and explain when an algorithmic design situation calls for it. For a given problem develop the greedy algorithms.
- Describe the dynamic-programming paradigm and explain when an algorithmic design situation required it. For a given problems of dynamic-programming and develop the dynamic programming algorithms, and analyze it to determine its computational complexity.
- Design and Apply graph algorithms for real-life problems.
- For a given model engineering problem model it using graph and write the corresponding algorithm to solve the problems.
- Write the method of analyze randomized algorithms and calculate the expected running time and probability of error.

13. Curriculum Contents

- Models of computation: RAM model and its logarithmic cost.
- Formal introduction to algorithmic paradigms: divide and conquer, recursion, dynamic programming, greedy, branch and bound.
- Advanced data structures: Skip lists, Fibonacci heap, union find.
- Amortized complexity analysis.
- Randomized algorithms: Las Vegas and Monte Carlo
- Graph algorithms: Network Flows, Bipartite Matching, Stable Marriage
- String matching algorithms
- Reducibility between problems and NP completeness
- Approximation algorithms: Constant ratio approximation algorithms.
- Introduction to external memory algorithms, parallel algorithms.

List of Labs

- Implement the sorting algorithms and compare the performance.
- Implementation and performance analysis of Skip Lists for search, insertion, and deletion operations.
- Implementation of Fibonacci Heap operations and comparison with Binary Heap for priority queue applications.
- Design and implementation of the Union-Find (Disjoint Set Union) data structure.
- Implementation and comparative analysis of Dynamic Programming algorithms for the 0/1 Knapsack and Longest Common Subsequence problems.
- Implementation of Randomized QuickSort and experimental evaluation against deterministic QuickSort.
- Design and implementation of the Ford-Fulkerson Algorithm for computing maximum network flow.
- Implementation of the Gale-Shapley Stable Marriage Algorithm and analysis of stable matching outcomes.
- Design and implementation of a 2-Approximation Algorithm for the Vertex Cover Problem and evaluation of approximation quality.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Quiz/Class participation – 30%
- Lab/Assignment – 20%

15. Textbooks and References

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. *Introduction to Algorithms* (4th ed.). Cambridge, MA: MIT Press, 2022.
- Motwani, R., & Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4901
4. Course Title	Software Engineering
5. Credits (L:T:P)	3 (2:0:1)
6. Contact Hours (L:T:P)	2:0:2
7. Prerequisites	CSD2001
8. Major Core for	None (Project Basket)
9. Major Elective for	None

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

Software engineering is the branch of computer science that creates practical, cost-effective solutions to computing and information processing problems, preferentially by applying scientific knowledge, developing software systems in the service of mankind. This course covers the fundamentals of software engineering, including understanding system requirements, finding appropriate engineering compromises, effective methods of design, coding and testing, team software development, and the application of engineering tools.

11. Course Objectives

The aim of the course is that the students are able to:

- Design software requirement specifications of a problem analyze the requirement specification based on software requirement analysis and develop the SRS (Software Requirement Specification) document.
- Draw the Data Flow Diagrams (DFDs) and structure charts for given software program and develop the program based on software design principle.
- For a given Software Requirement Specification develop the software design using UML (Unified Modeling Language).
- For a given program code, design the black box, white box and performance test suites.

12. Learning Outcomes

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

- Understand various phases of the software development lifecycle.
- Analyze the requirements systematically and develop the model using standard tools and techniques.
- Apply key aspects of software engineering processes for the development of a complex software system.
- Develop a quality software project through effective team building, planning, scheduling, and risk assessment.

13. Curriculum Content

- Importance of Software Engineering, Phases of software development lifecycle, SDLC case study, Software process model, Waterfall model, V Model, Prototyping model, Incremental model, RAD Model, Spiral model, choosing a model, Lifecycle documents, Version Control System, Distributed VC, Git repository, Git online and Desktop, check in, and check out code in repository, Create branch and merging branch, Git Eclipse/IDE.
- Agile model, need of agile, Agile manifesto, Agile principles, Agile development methods, Extreme programming, Test first development, Refactoring, Pair programming, Scrum, Product backlog, Sprint cycle, Requirements engineering, Issues in capturing requirements, Requirement elicitation, Requirement analysis, Functional and Non-functional requirements, Requirement specification, Requirement prioritization, User stories, Acceptance criteria, Requirement validation and verification; Overview of design process: High-level and detailed design, Cohesion and coupling, Modularity and layering, Function-Oriented software design: Structured Analysis using DFD Structured Design

- UML, Use case, Use case Diagram, Include and extend relationship, Generalization in use Case, Top down and bottom up approach in use case diagram, Guidelines for creating use case diagrams, UML behavioral diagrams, Activity diagram, Activity diagram with swimlane UML structural diagrams, Class diagram, Relationships in class diagram, Sequence diagram, Rules of creating sequence diagram, Description of relationship between use case, activity, and sequence diagram, Architectural design, Cohesion, Coupling, Early locking of architecture, Architectural pattern, MVC pattern, Layered architecture, Repository architecture, Client server architecture, Software architect, roles, and responsibilities, Pipe and filter architecture.
- Software quality assurance and testing, Software testing for competitive advantage, Testing strategies, Designing test cases, Black box testing, Equivalence partitioning, Boundary value analysis, White box testing, Black box vs white box testing, Control flow testing technique, Control flow graph, Cyclometric complexity, Levels of coverage, Integration testing, Top down and bottom up integration, Continuous Integration testing, Regression testing, Acceptance testing, System testing, Performance testing, Load testing

List of Labs

- Prepare a Software Requirements Specification (SRS) document following the IEEE standard template.
- Design Data Flow Diagrams (DFDs) to represent system processes and data movement.
- Develop UML diagrams like Use Case Diagram, Class Diagram, Activity Diagram, utilizing online modeling tools such as Visual Paradigm and Draw.io.
- Implement the design using Java.
- Perform software testing and code coverage analysis using tools such as Junit, gcov, and Cobertura.
- Conduct web application testing using Selenium to validate functionality and user interactions.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Labs/Assignment – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Mall, R. (2024). Fundamentals of Software Engineering (6th ed.). PHI Learning Pvt. Ltd., New Delhi.
- Sommerville, I. (2011). Software Engineering (9th ed.). Pearson Education Limited, Harlow, England.
- Jalote, P. (2008). Software Engineering: A Precise Approach. Wiley India Pvt. Ltd., New Delhi.
- Berzins, V., & Luqi. (1991). Software Engineering with Abstractions. Addison-Wesley Publishing Company, Reading, MA.
- Shooman, M. L. (1983). Software Engineering: Design, Reliability, and Management. McGraw-Hill Book Company, New York.
- Aggarwal, K. K., & Singh, Y. (2005). Software Engineering (3rd ed.). New Age International (P) Limited, New Delhi.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4005
4. Course Title	Advanced Computer Architecture
5. Credits (L:T:P)	4 (3:1:0)
6. Contact Hours (L:T:P)	3:1:0
7. Prerequisites	CSD2003, CSD2005
8. Major Core for	CSE
9. Major Elective for	None

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

The course provides students with a fundamental knowledge of computer hardware and computer systems, with an emphasis on system design and performance. The course concentrates on the principles underlying systems organisation, issues in computer system design, and contrasting implementations of modern systems.

11. Course Objectives

This course aims to provide students with a comprehensive understanding of modern computer architecture and organisation. Students will explore the fundamental principles and techniques used to design and optimise high-performance computer systems. Topics will span from the microarchitectural level (pipelining, instruction-level parallelism) to the microarchitecture level (multiprocessors, memory hierarchies). The course emphasises the interplay between hardware and software and will equip students with the ability to analyse and evaluate the performance implications of various design choices. Through a combination of lectures, assignments, and projects, students will develop the skills to apply theoretical concepts to real-world scenarios, fostering a deep appreciation for the intricacies of computer systems engineering.

12. Learning Outcomes

- Draw a functional block diagram of the organisation of computer-based systems and discuss how applications influence the range of design choices.
- For a given memory configuration details compute response time, throughput, miss-rate, miss penalty
- For a given multi-core processor architecture, develop multi-threaded programs and determine the speed-ups.
- Analyse the performance of a given cache memory organisation and its optimization techniques.
- Given a RAID configuration, determine the data reliability and data throughput

13. Curriculum Content

- **Introduction:** Review of basic computer architectures, quantitative techniques in computer design, measuring and reporting performance. CISC and RISC processors. Instruction-level parallelism - data dependency, Instruction Pipeline and Performance.
- **Pipelining:** Basic concepts, instruction and arithmetic pipeline, hazards: data hazards, control hazards, and structural hazards, techniques for handling hazards. Branch prediction, Dynamic instruction scheduling, Speculative Execution, Compiler techniques for hazard resolution.
- **Advanced Processors:** Superscalar processors, super-pipelined and VLIW processor architectures, Exploiting Data Level Parallelism: Vector and GPU Architectures, Multi-threaded and multicore processors, memory synchronization, consistency, and coherence
- **Cache Memories:** Block Replacement Techniques and Write Strategy, Design Concepts in Cache Memory, Cache memory organisations, Basic and Advanced Optimization Techniques in Cache Memory, Techniques for reducing cache misses.
- **Memory system:** DRAM memory, Secondary Storage Systems, Hard Disk and Flash memory.

14. Assessment Scheme:

- Examination (Midterm/Final) – 50%
- Assignment/Tutorial – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Shen, J. P., & Lipasti, M. H. (2013). *Modern Processor Design: Fundamentals of Superscalar Processors* (2nd ed.). Waveland Press.
- Michael J. Flynn, *Computer Architecture: Pipelined and Parallel Processor Design*, Narosa Publishing House, 1995.
- Kai Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, McGraw-Hill, 1993.
- Hennessy, J. L., Patterson, D. A., and Kozyrakis, C., *Computer Architecture: A Quantitative Approach*, 7th ed., Morgan Kaufmann/Elsevier, 2025. ‘
- Vincent P. Heuring and Harry F. Jordan, *Computer Systems Design and Architecture*, 2nd Edition, Addison-Wesley, 2003.
- William Stallings, *Computer Organization and Architecture: Designing for Performance*, 10th Edition, Pearson, 2016.
- Dezső Sima, Terence Fountain, and Peter Kacsuk, *Advanced Computer Architectures: A Design Space Approach*, Addison-Wesley, 1997.

Course Description

Elective Courses

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4201
4. Course Title	Advanced Database Management System
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD4801
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICES

10. Course Summary

This course covers advanced topics in database management systems (DBMS). It is designed for students with prior knowledge of relational databases, SQL, and basic database theory. Topics will include advanced query optimization, distributed databases, NoSQL databases, data warehousing and OLAP, and special purpose databases. Practical assignments will include hands-on work with various database management systems.

11. Course Objectives

To enable students:

- To learn the query optimization techniques.
- Explore indexing types and apply them in case studies To design Search Engine
- Understand distributed databases and NoSQL systems through hands-on labs.
- Learn ETL processes and OLAP operations in data warehousing.
- Investigate special-purpose databases, including temporal and spatial databases.

12. Learning Outcomes

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

- Differentiate among major data models such as relational, spatial, and NoSQL
- Understand advanced query optimization techniques.
- Study and implement distributed databases and replication.
- Understand and use NoSQL database systems.
- Understand ETL processes and OLAP operations in data warehousing.
- Explain methods suitable for particular types of data such as temporal, multimedia or spatial data.

13. Curriculum Content

- **Review of Database Concepts:** Review of relational models and SQL, Recap of normalization and database design principles, transactions and recovery techniques, Introduction to advanced database topics.
- **Query Optimization and Execution:** Query parsing, rewriting, and execution, Cost-based and heuristic-based query optimization, Join optimization strategies.
- **Hands-on:** exercises on query optimization using SQL Server/MySQL.
- **Indexing and Advanced Data Structures:** Types of indexes (B-trees, Hash indexes, Bitmap indexes), Spatial and temporal indexing.
- **Case study:** Using indexes in real-world scenarios.
- **Distributed Databases:** Distributed database architecture, Fragmentation, replication, and allocation, Distributed query processing, Distributed transactions and commit protocols, Distributed concurrency control, Consistency models in distributed systems (CAP theorem).
- **Hands-on project:** Setting up a basic distributed database.
- **NoSQL Databases:** Introduction to NoSQL databases and their types (Key-Value, Document, Column-family, Graph databases), Comparison of relational vs. NoSQL systems. Use cases and applications of NoSQL. Schema design and data modelling in NoSQL, Query languages in NoSQL.
- **Practical lab:** Working with MongoDB or Cassandra and handling Complex queries in NoSQL.

- **Data Warehousing and OLAP:** Data warehousing concepts and architecture, ETL processes (Extract, Transform, Load), OLAP (Online Analytical Processing) operations and cube queries.
- **Hands-on lab:** Creating a data warehouse using PostgreSQL.
- **Special purpose databases:** Temporal, spatial and multimedia databases.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.
- Özsu, M. T., & Valduriez, P. (2020). *Principles of Distributed Database Systems* (4th ed.). Springer.
- Sadalage, P. J., & Fowler, M. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley Professional.
- Seidman, J., & Malaska, T. (2024). *Foundations for Architecting Data Solutions: Managing Successful Data Projects*. O'Reilly Media.
- Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson.

PART A: COURSE IDENTIFIERS

1. School	SoE
2. Department	CSE
3. Course Code	CSD4237
4. Course Title	AI Applications
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	Knowledge of Probability and Statistics & Machine Learning
8. Major Core for	NA
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This course introduces the fundamentals of Artificial Intelligence (AI) and Machine Learning (ML), including deductive and inductive reasoning approaches, core ML concepts, and applications of AI and ML across diverse domains.

11. Course Objectives

This course aims to:

- Learn to distinguish between AI and ML and understand their applications and use
- Examine AI and ML frameworks with a focus on what is working and what is not
- Gain valuable new insights into how AI can be used for public good
- Learn applications of AI in Data Science, Healthcare, and several other areas.

12. Learning Outcomes

After successful completion, the student will be able to:

- Explain the fundamental concepts of Artificial Intelligence and Machine Learning.
- Apply deductive and inductive reasoning approaches to solve real-world problems.
- Analyze supervised, unsupervised, and deep learning methods.
- Evaluate the suitability of AI and ML techniques for different application domains.
- Apply AI and ML concepts to address societal and public-good challenges.

13. Curriculum Content

- AI Search Techniques
- Decision Modelling Approaches
- AI Methods with a Focus on Applications
- Machine Learning Methods: Supervised and Unsupervised Learning
- Perceptrons and Neural Networks
- Deep Learning Fundamentals
- Semi-Supervised, Self-Supervised, and Weakly Supervised Learning
- Continual Learning
- Active Learning
- Applications of AI and ML in Various Domains

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Aggarwal, C. C. (2021). *Artificial Intelligence: A Textbook*. Springer.

- Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson Education.
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press.
- Prince, S. J. D. (2023). *Understanding Deep Learning*. MIT Press.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Bishop, C. M., & Bishop, H. (2024). *Deep Learning: Foundations and Concepts*. Springer.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4203
4. Course Title	Algorithms for Big Data
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

This course will introduce students to how to design and analyse algorithms in the streaming data. The algorithms will be analysed mathematically, so it is intended for a mathematically mature audience with prior knowledge of algorithm design and basic probability theory.

11. Learning Objectives

- To introduce the fundamental concepts of randomized algorithms, probabilistic analysis, and probabilistic counting techniques.
- To develop an understanding of Markov chains, random walks, and their applications in algorithm design and analysis.
To familiarize students with the streaming model and efficient algorithms for processing massive data streams.
- To study sketching, hashing, and approximate computation techniques for large-scale data analysis.
- To introduce property testing models and randomized methods for testing graph and combinatorial properties.
- To explore the role of randomness in designing scalable algorithms with provable performance guarantees.

12. Learning Outcomes

Upon successful completion of this course, students will be able to:

- Analyze probabilistic processes using Markov chains, random walks, and probabilistic counting techniques.
- Apply randomized algorithms and adversarial models to solve computational problems efficiently.
- Design and implement streaming algorithms for frequency moments, distinct element estimation, and approximate counting.
- Utilize pairwise independence, universal hashing, Bloom filters, and sketch-based techniques for large-scale data processing.
- Analyze and apply Flajolet-Martin and Alon-Matias-Szegedy (AMS) sketch algorithms for data stream analytics.
- Evaluate graph and combinatorial properties using property testing techniques, including bipartiteness, connectivity, and triangle-freeness testing.
- Apply randomized testing methods such as the BLR linearity test and other property testing algorithms for Boolean functions.
- Compare exact and approximate approaches for large-scale computational problems and justify the use of randomized techniques in resource-constrained environments.

13. Curriculum Content

- Probabilistic counting, Markov Chains and Random Walks
- Randomized Algorithms against an Oblivious Adversary
- Intro to frequency moments in streaming, Distinct element estimation
- Pairwise Independence and Universal Hashing

- The Streaming Model
- Approximate Counting and approximate Median
- Flajolet Martin algorithm
- Alon-Mattias-Szegedy Sketch algorithm
- Bloom Filters
- Property Testing Model
- Local search and testing connectivity Enforce and Test Technique: Biclique and Bipartiteness Testing
- Random Walks and Testing Bipartiteness & Expansion
- Regularity Lemma and Testing Triangle Freeness
- Boolean Functions, BLR test for Linearity

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Mitzenmacher, M., & Upfal, E. (2017). *Probability and Computing: Randomized Algorithms and Probabilistic Analysis* (2nd ed.). Cambridge University Press.
- Ron, D. (2001). *Property Testing: A Learning Theory Perspective*. Foundations and Trends in Machine Learning, Now Publishers.
- Cormode, G., Garofalakis, M., Haas, P. J., & Jermaine, C. (2012). *Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches*. Foundations and Trends in Databases, Now Publishers.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4204
4. Course Title	Applied Cryptography
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD1001, CSD2002
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This course introduces modern cryptographic primitives and selected cryptographic protocols used in secure digital systems. Topics include symmetric-key and public-key cryptography, cryptographic hash functions, message authentication codes, digital signatures, authenticated encryption, key establishment, and secure communication protocols. The course also provides an overview of post-quantum cryptography and quantum-resistant cryptosystems for securing future digital systems.

11. Course Objectives

- To introduce the fundamental concepts and modern cryptographic primitives used in secure digital systems.
- To develop an understanding of cryptographic protocols and their application in solving real-world security and privacy challenges.
- To provide hands-on experience through assignments and programming exercises involving core cryptographic primitives, implementation techniques, and analysis of common cryptographic attacks.

12. Learning Outcomes

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

- Explain the fundamental concepts of cryptography and the standard cryptographic algorithms used to provide confidentiality, integrity, authentication, and non-repudiation.
- Apply modern cryptographic primitives, including symmetric-key and public-key encryption, cryptographic hash functions, message authentication codes, digital signatures, certificates, and key management techniques, to secure digital systems.
- Analyze the design and application of cryptographic protocols for addressing real-world security and privacy challenges.
- Implement and evaluate basic cryptographic techniques through programming assignments and practical case studies.
- Describe the principles of post-quantum cryptography and evaluate quantum-resistant cryptographic approaches for emerging secure communication systems.

13. Curriculum Content

- **Unit-I:** Course Introduction and terminology, Conventional Cryptography: Definitions, Classical encryption techniques, One-time pad, Perfect Secrecy, DES, Triple DES, Finite fields, AES, Modes of Encryption.
- **Unit-II:** Stream Ciphers, LFSR based stream ciphers, Message Authentication Codes, Cryptographic Hash functions, Hash algorithms.
- **Unit-III:** Asymmetric Cryptography: Number Theory, public key cryptography: RSA, ElGamal, and Elliptic Curve Cryptography, Diffie Hellman Key Exchange Protocol, Digital Signatures and Authentication Protocols. Digital Certificates: X.509.
- **Unit-IV:** Email Security, IPsec, SSL/TLS, Password Hashing.
- **Unit-V:** Basic overview of Post-quantum Cryptosystem: approaches and existing algorithms.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Text Books and References

- Jonathan Katz, Yehuda Lindell, "Introduction to Modern cryptography", 2nd Ed., CRC Press.
- Douglas R. Stinson, "Cryptography: Theory and Practice 3/e", CRC Press, 2006.
- W Stallings, "Cryptography and Network Security: Principles and Practice, 6/e", Prentice Hall.
- Christof Paar, Jan Pelzl, "Understanding Cryptography", Springer.
- A. Menezes, P. van Oorschot, S. Vanstone. "Handbook of Applied Cryptography", CRC press, 1997.
- B. Schneier. "Applied Cryptography". Second Edition. John Wiley & Sons, Inc., 1996.
- Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, "An Introduction to Mathematical Cryptography", Springer.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4205
4. Course Title	Blockchain Technology
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

From this course student will learn the basics of blockchain technology and its application mainly as cryptocurrency. The topics to be covered include how cryptocurrency like Bitcoin and Ethereum work, different decentralized consensus mechanisms, Smart Contracts, Decentralized Apps, cryptographic techniques, security and privacy in blockchain, scalability and efficiency aspects such as concurrency of transactions in a block, public and private networks, introduction to Hyperledger Fabric architecture and different use case scenarios. Students will later explore the efficient execution of smart contracts on blockchain using parallel computing.

11. Course Objectives:

- Enable students to understand the fundamentals of blockchain technology and its security implications.
- Identify common vulnerabilities and security challenges in blockchain systems.
- Analyze real-world security incidents in blockchain applications.
- Implement best practices for securing blockchain networks and applications considering efficiency aspects of implementation.

12. Learning Outcomes

- Understand how bitcoin or other cryptocurrency works in real world.
- Understand how cryptography contributes to developing trust, transparency at the same time security in the blockchain network.
- Create Decentralized Applications (DApps) i.e., applications that can be deployed on blockchain network.
- Understand how efficiency of smart contracts through concurrency can make real-time applications more efficient.

13. Curriculum Content

- Introduction to cryptography, blockchain & cryptocurrencies, Bitcoin, Merkle tree, P2P networks, transactions, blocks, Proof of work, wallets.
- Foundation of consensus, network models, corruption tolerance, different decentralized consensus mechanisms such as proof of work, proof of stake.
- Ethereum basics, Decentralized Apps, EVM, and the Ethereum blockchain, Introduction of programming in solidity.
- Introduction to private blockchain: Hyperledger fabric architecture, use case scenarios.
- Security and privacy in public blockchain, anonymity, privacy enhancing cryptography, ring signature, overview of Monero crypto currency.
- Basics of parallel and concurrent programming, Pthreads, thread synchronization, Locking; Efficient execution of Smart contracts, Concurrent execution of smart contracts, Sharding, Blockchain applications; and future directions in Blockchain technology.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%

- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press.
- Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Available at: <https://bitcoin.org/bitcoin.pdf>
- Wood, G. (2025). *Ethereum: A Secure Decentralised Generalised Transaction Ledger (Ethereum Yellow Paper)*. Available at: <https://ethereum.github.io/yellowpaper/paper.pdf>
- *Solidity Documentation*. Ethereum Foundation. Available at: <https://docs.soliditylang.org>
- *Hyperledger Fabric Documentation: Tutorials*. Linux Foundation. Available at: <https://hyperledger-fabric.readthedocs.io/en/latest/tutorials.html>
- Alonso, K. M. (2018). *Zero to Monero: First Edition – A Technical Guide to a Private Digital Currency for Beginners, Amateurs, and Experts* (Version 1.0.0). Available at: <https://www.getmonero.org/library/Zero-to-Monero-1-0-0.pdf>
- Shukla, S., Dhawan, M., Sharma, S., & Venkatesan, S. (2019). *Blockchain Technology: Cryptocurrency and Applications*. Oxford University Press.
- Thompson, J. (2017). *Blockchain: The Blockchain for Beginners – Guide to Blockchain Technology and Blockchain Programming*. CreateSpace Independent Publishing Platform.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4206
4. Course Title	Cloud Computing
5. Credits (L:T:P)	3 (2:1:0)
6. Contact Hours (L:T:P)	2:1:0
7. Prerequisites	CSD1001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This course includes the introductory concepts about Cloud computing, cloud computing models, virtualization techniques, the building blocks, technology and tools. It would also enable the students to understand the various cloud services and handling Big Data in detail. They course would enable them to be able to understand the design concepts of the cloud computing as well as foundations of cloud management. It also elaborates cloud compliances and security at fine-grained level by following a shared responsibility model. It enables students to think through the process of designing and implementing cloud infrastructure and optimal IT solutions.

11. Course Objectives

The aim of the course is to:

- Understand the architecture and design of Cloud Computing.
- Learn the usage of cloud services.
- Understand how the platform is managed.

12. Learning Outcomes

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

- Understand and explain fundamental concepts of cloud computing.
- Understand the concept of virtualization.
- Understand Cloud management.
- Demonstrate the use of HDFS for data management.

13. Curriculum Content

- Cloud Computing, Adoption of cloud-based IT resources, Service Models: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-aService (SaaS), Deployment models: Public Cloud, Private Cloud, Hybrid Cloud, Community Cloud, Cloud Computing Characteristics, Challenges of cloud computing, Virtualization concept, Types of virtualizations, Demo of virtualization, Virtualization Merits, Role of virtualization in cloud computing, Virtualization Demerits, VM Placement, VM Migration, VM Migration Demo, VM clustering, Design Issues in VM Clustering, Need of Dockers and Containers, Docker EcoSystem, Hypervisor vs Docker.
- Microservices, Service-Oriented Architecture, REST API, IP Addressing, Subnetting, Supernetting, Designing of Virtual Private Cloud, Demo of VPC, VPC Peering, VPC Case Study, Cloud Storage, Serverless Computing, Cloud API Gateway, Cloud Databases, Resource Provisioning, Time shared and space shared, Efficient VM Consolidation on cloud server, Task/DAG Scheduling Algorithms Task-VM Mapping, Auto Scaling, Load Balancing.
- Cloud Market analysis, Security and Compliances, Implementation of cloud security, Cloud databases, Parallel Query Execution with NoSQL Database, Handling Big Data on Cloud Platform, Map-Reduce framework for large clusters using Hadoop, Design of data applications based on Map Reduce in Apache Hadoop, Advanced Research Topics in Cloud Computing

14. Assessment Scheme

- Examination (Midterm/Final) – 50%

- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Cloud Computing (Principles and Paradigms), Edited by Rajkumar Buyya, James Broberg, Andrzej Goscinski, John Wiley & Sons, Inc. 2011
- Cloud Computing: Concepts, Technology and Architecture, Thomas Erl, Ricardo Puttini, Zaigham Mahmood, Pearson Education, 2013.
- Kai Hwang, Jack Dongarra, Geoffrey C. Fox Distributed and Cloud Computing: From Parallel Processing to the Internet of Things. Morgan Kaufman 2011.
- Jim Smith, Ravi Nair. Virtual Machines: Versatile Platforms for Systems and Processes. Morgan Kaufmann. 2005

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4207
4. Course Title	Compiler Design
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2001
8. Major Core for	NA
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

The course is designed to understand the basic principles of design and implementation a compiler and also be able to explain the underlying theories. The main emphasis is for the imperative language. The course covers Language processing and Grammars, top-down and bottom-up techniques of parsing, Semantic Analysis, Code Generation, Run-time environment, Programming Paradigms.

11. Course Objectives

- To understand the phases of a compiler and the principles of language translation.
- To study lexical, syntax, and semantic analysis techniques used in compiler construction.
- To learn parsing methods and the design of syntax analyzers for programming languages.
- To understand symbol table management, run-time environments, and intermediate code generation.
- To apply code optimization techniques for improving program efficiency.
- To develop the ability to generate target code from source programs using compiler design principles.

12. Learning Outcomes

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

- Analyze source programs and identify lexical tokens using formal language concepts.
- Design and implement lexical analyzers and parsers using appropriate compiler construction techniques.
- Apply semantic analysis methods and manage symbol tables effectively.
- Explain run-time storage organization, memory allocation, and parameter passing mechanisms.
- Generate intermediate representations and target code for programming language constructs.
- Apply machine-independent and machine-dependent optimization techniques to improve code quality and execution efficiency.

13. Course Content

- **Lexical Analysis (scanner):** Regular languages, finite automata, regular expressions, from regular expressions to finite automata, scanner generator (lex, flex).
- **Syntax Analysis (Parser):** Context-free languages and grammars, push-down automata, LL(1) grammars and top-down parsing, operator grammars, LR(O), SLR(1), LR(1),LALR(1) grammars and bottom-up parsing, ambiguity and LR parsing, LALR(1)parser generator (yacc, bison)
- **Semantic Analysis:** Attribute grammars, syntax directed definition, evaluation and flow of attribute in a syntax tree. Symbol Table: Its structure, symbol attributes and management.
- **Run-time environment:** Procedure activation, parameter passing, value return, memory allocation, and scope.
- **Intermediate Code Generation and Code Improvement:** Translation of different language features, different types of intermediate forms, code optimization.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%

- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman, *Compilers: Principles, Techniques and Tools*, 2nd Edition, Addison-Wesley, 2006.
- Michael L. Scott, *Programming Language Pragmatics*, 4th Edition, Morgan Kaufmann, 2015.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4208
4. Course Title	Computer Graphics
5. Credits (L:T:P)	3 (2:0:1)
6. Contact Hours (L:T:P)	2:0:2
7. Prerequisites	CSD2001, CSD2004
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This course provides a comprehensive introduction to computer graphics, image synthesis, and rendering techniques. It covers the mathematical foundations of graphics, geometric transformations, rasterization, ray tracing, physically based rendering, global illumination, and human visual perception. Students gain practical experience in implementing rendering algorithms and using modern graphics APIs.

11. Course Objectives

The course aims to:

- Introduce the fundamental principles of computer graphics and image generation.
- Develop understanding of geometric modeling, transformations, and rendering pipelines.
- Explore modern rendering techniques including rasterization, ray tracing, and global illumination.
- Enable students to implement graphics algorithms and analyze image realism and visual perception.

12. Learning Outcomes

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

- Understand fundamental principles of computer graphics and rendering
- Implement basic rendering algorithms using ray tracing and rasterization
- Apply material models and texturing techniques to create realistic images
- Analyse human vision effects on image perception and applying tone mapping techniques

13. Curriculum Content

- Introduction to Computer Graphics and Graphics-based Systems.
- Radiometry and Light Transport, The Rendering Equation, Ray Tracing, Transformations, Acceleration Structures.
- Material Models, Texturing, Sampling Theory, Distribution Raytracing.
- Human Vision, Color, High Dynamic Range (HDR) Imaging, and Tone Mapping
- Global Illumination, Path Tracing, Monte Carlo Estimation
- Participating media, Volumetric rendering, Subsurface scattering
- Image-based Rendering and Neural Rendering
- Rasterization, Rasterization vs. Ray-tracing, OpenGL as a Graphics API

14. Assessment Scheme

- End-Sem Exam – 30%
- Mid-Sem Exam – 25%
- Assignment – 10%
- Project – 20%
- Quiz – 15%

15. Text Books and References

Recommended textbooks:

- Steve Marschner and Peter Shirley, Fundamentals of Computer Graphics, 5th Edition, A K Peters/CRC Press, 2021.
- Matt Pharr, Wenzel Jakob, and Greg Humphreys, Physically Based Rendering: From Theory to Implementation, 4th Edition, MIT Press, 2023. Official PBRT online book
- Tomas Akenine-Möller, Eric Haines, and Naty Hoffman, Real-Time Rendering, 4th Edition, A K Peters/CRC Press, 2018.
- John F. Hughes, Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, and Kurt Akeley, Computer Graphics: Principles and Practice, 3rd Edition, Addison-Wesley Professional, 2013/2014.

Books For Programming:

- Peter Shirley, Trevor David Black, and Steve Hollasch, Ray Tracing in One Weekend series, current online release. The first volume reached version 4.0.2 in April 2025. (online)

Advanced Reading:

- Philip Dutré, Philippe Bekaert, and Kavita Bala, Advanced Global Illumination, 2nd Edition, A K Peters/CRC Press, 2006.
- Eric Haines and Tomas Akenine-Möller, editors, Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs, Apress, 2019.
- Adam Marrs, Peter Shirley, and Ingo Wald, editors, Ray Tracing Gems II: Next Generation Real-Time Rendering with DXR, Vulkan, and OptiX, Apress, 2021.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4209
4. Course Title	Computer Vision
5. Credits (L:T:P)	3 (2:0:1)
6. Contact Hours (L:T:P)	2:0:2
7. Prerequisites	CSD2001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

The goal of the Computer Vision course is to provide hands-on knowledge on applying popular Computer Vision techniques to handle images and videos. The students of this course will be given opportunities to do one research project and a set of assignments. The course curriculum is designed to equip the students with the recent advances in Computer Vision.

11. Course Objectives

- To introduce the fundamental concepts and techniques of computer vision.
- To develop an understanding of image processing, feature extraction, and image analysis methods.
- To study geometric and photometric principles used in visual perception and scene reconstruction.
- To explore machine learning and deep learning approaches for visual recognition tasks.
- To provide hands-on experience in implementing computer vision algorithms for real-world applications.
- To familiarize students with recent advances in deep learning-based computer vision.

12. Learning Outcomes

This course will enable the students to acquire a deep understanding on the recent techniques for computer vision. On successful completion of the course, students will be able to achieve the following:

- Understand the basic and traditional ideas of computer vision problems.
- Implement different popular computer vision techniques.
- Have a deep knowledge about the recent advances of Computer Vision based on deep learning techniques.
- Solve real life problems on computer vision.

13. Curriculum Content

- **Introduction to Computer Vision:** Fundamentals of computer vision, image formation, visual perception, applications and challenges.
- **Low-Level Vision:** Image processing, filtering and enhancement, edge detection, feature detection and description, image matching, feature correspondence, RANSAC, image segmentation.
- **Geometry and Photometry:** Camera models, projective geometry, epipolar geometry, stereo vision, depth estimation, structure from motion, image registration, photometric modeling.
- **Recognition and Learning:** Object recognition, image classification, pattern recognition, machine learning for computer vision, artificial neural networks, feature learning.
- **Deep Learning for Computer Vision:** Convolutional Neural Networks (CNNs), CNN architectures, object detection, semantic segmentation, instance segmentation, transfer learning, vision transformers.
- **Generative Models in Computer Vision:** Autoencoders, Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), image synthesis, image restoration and generation.
- **Advanced Topics and Applications:** Video analysis, motion tracking, activity recognition, face recognition, medical image analysis, autonomous systems, recent advances and research trends in computer vision.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Richard Szeliski, *Computer Vision: Algorithms and Applications*, 2nd Edition, Springer, 2022.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.
- Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola, *Dive into Deep Learning*, Cambridge University Press, 2023.

PART A: COURSE IDENTIFIERS

1. School	SOE
2. Department	Computer Science and Engineering
3. Course Code	CSD4210
4. Course Title	Deep Learning
5. Credits (L:T:P)	3 (2:0:1)
6. Contact Hours (L:T:P)	2:0:2
7. Prerequisites	CSD4002
8. Major Core for	NA
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

Deep learning, a subset of machine learning, focuses on algorithms inspired by the structure and function of the brain called artificial neural networks. This course introduces students to the core concepts, techniques, and applications of deep learning, enabling them to build and train neural networks for a wide range of tasks.

11. Course Objectives

The aim of the Deep Learning course is to equip students with a comprehensive understanding of the theoretical principles and practical techniques of deep learning. Students will develop the ability to design, implement, and optimize neural networks for solving complex problems across various domains, such as computer vision, natural language processing, and autonomous systems.

12. Learning Outcomes

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

- Develop an understanding of the theoretical underpinnings of neural networks and deep learning algorithms.
- Implement and experiment with various deep learning models using libraries like TensorFlow or PyTorch.
- Apply deep learning techniques to solve practical problems in domains like computer vision, NLP, and robotics.
- Critically evaluate the performance of deep learning models and their societal implications.

13. Curriculum Content

- **Introduction to Deep Learning:**
Understanding neural networks and their biological inspiration.
Differences between deep learning, machine learning, and traditional programming.
Backpropagation and Loss functions.
Universal Approximation Theorem
- **Linear Models:**
Linear Regression: loss function, feature encoding, parameters
Linear Classification: loss function, softmax function.
- **Multi-Layer Perceptron:**
Basic architecture of neural networks.
Activation functions, loss functions, and gradient descent optimization.
Why Deep is better than Shallow?
Vanishing and Exploding gradients issues and solutions.
Requirement of Regularization, Dropouts.
- **Convolutional Neural Networks (CNNs):**
Understanding convolutional layers and their application in image processing.
Striding and Deconvolutions (Transpose convolution)
Pooling layers and fully connected layers for classification tasks.
Importance of Residual connections.

- **Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM):**
Sequential data modeling, including time series and natural language processing.
Overcoming limitations of vanilla RNNs with LSTMs and GRUs.
- **Regularization Techniques (In detail):**
Preventing overfitting with methods like dropout, batch normalization, and data augmentation.
- **Transformers:**
Formulation of transformer and attention Mechanism
Application in sequential data and vision data.
- **Generative Models:**
Autoencoders, generative adversarial networks (GANs), and their applications in generative tasks.
- **Transfer Learning:**
Using pre-trained models for specific tasks to reduce training time and computational costs.
- **Optimization Techniques(in detail):**
Adam, RMSprop, and advanced techniques to optimize deep networks.
- **Applications of Deep Learning:**
Real-world use cases such as image classification, natural language processing, autonomous systems, and healthcare.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola, *Dive into Deep Learning*, Cambridge University Press, 2023.
- Simon J. D. Prince, *Understanding Deep Learning*, MIT Press, 2023.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.
- Christopher M. Bishop and Hugh Bishop, *Deep Learning: Foundations and Concepts*, Springer, 2024.

PART A: COURSE IDENTIFIERS

1. School	SOE	
2. Department	Computer Science and Engineering	
3. Course Code	CSD4211	
4. Course Title	Image Processing and Applications	
5. Credits (L:T:P)	3 (2:0:1)	
6. Contact Hours (L:T:P)	2:0:2	
7. Prerequisites	CSD2002	
8. Major Core for	NA	
9. Major Elective for	CSE	

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

Image is one of the fundamental data types in current era. Extracting essential information and correcting acquired image data are important functions for any image related application. Students going to learn both functions in image processing course. Course will cover the fundamental mathematical models of images and many traditional algorithms. Also course will introduce deep learning based methods and it's relation to traditional algorithms.

11. Course Objectives

The course aims to:

- Introduce the mathematical foundations of digital image processing.
- Develop understanding of image enhancement, restoration, segmentation, and analysis techniques.
- Enable students to implement image processing algorithms using modern computational tools.
- Introduce learning-based approaches and their relationship to traditional image processing methods.

12. Learning Outcomes

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

- Explain image formation, sampling, quantization, and digital image representation.
- Apply spatial-domain and frequency-domain techniques for image enhancement and restoration.
- Implement image processing algorithms for filtering, segmentation, and feature extraction.
- Analyze image data using mathematical and statistical techniques.
- Develop image processing applications using Python libraries such as NumPy and OpenCV.
- Compare traditional image processing methods with modern learning-based approaches.

13. Curriculum Content**Introduction:**

1. Difference between Imaging and Image Processing.
2. Sampling and Quantization.
3. Image Representation.
4. Image Resolution, Interpolation.
5. Neighbors of Pixel, Adjacency, Region and Boundaries.
6. Matrix operations

Intensity Transformation:

1. Basic intensity transformation
2. Histogram processing

Spatial Filtering:

1. Mechanics of spatial filtering and application
2. Smoothing and Sharpening

Frequency domain Filtering:

1. Complex numbers, Fourier series
2. Fourier Transform and Convolutions
3. Discrete Fourier Transform

4. Smoothing and Sharpening in Frequency domain

Image Restoration and Reconstruction:

1. Noise Model
2. Spatial Noise Filtering
3. Frequency domain Noise filtering

Morphological Image Processing:

1. Erosion and Dilation
2. Opening and Closing
3. Morphological Operations

Image Segmentation:

1. Point, Line and Edge detection
2. Thresholding
3. Region based segmentation
4. Pattern Based Segmentation.
5. Learning Patterns.

Some Special Topics in Advanced Image Processing**14. Assessment Scheme**

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing*, 4th Edition, Pearson, 2018.
- Anil K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.
- Wilhelm Burger and Mark J. Burge, *Digital Image Processing: An Algorithmic Introduction Using Java*, 3rd Edition, Springer, 2022.
- Simon J. D. Prince, *Computer Vision: Models, Learning, and Inference*, Cambridge University Press, 2012.
- Richard Szeliski, *Computer Vision: Algorithms and Applications*, 2nd Edition, Springer, 2022.
- Kevin P. Murphy, *Probabilistic Machine Learning: An Introduction*, MIT Press, 2022.
- Simon J. D. Prince, *Understanding Deep Learning*, MIT Press, 2023.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4212
4. Course Title	Distributed Systems
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2001, CSD2005
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

Distributed systems are ubiquitous and inherently complex, making them challenging to understand and develop. This course comprehensively explores the foundational principles, techniques, and tools essential for analyzing, designing, and building distributed applications and systems.

Students will learn the core concepts of distributed computing, including communication and coordination through message passing, and delve into classical distributed algorithms such as logical and vector clocks, leader election, fault tolerance, data consistency, and consensus. Additionally, the course will examine the architecture and design of large-scale distributed systems. Students will gain hands-on experience by implementing many of these concepts through homework assignments and projects, equipping them with practical skills to develop robust and scalable distributed applications.

11. Course Objectives

- Students get a solid understanding of the principles and practices of distributed computing.
- They learn how to design, develop, and manage systems where components located on networked computers communicate and coordinate to achieve common goals.
- To enable students with the fundamental concepts such as communication protocols, synchronization, fault tolerance, consistency, and scalability.
- By the end of the course, students will be equipped to address challenges in building reliable, efficient, and scalable distributed applications in real-world environments.

12. Learning Outcomes

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

- Gain the ability to design and develop functional distributed systems while applying key principles and best practices.
- Learn to create and evaluate algorithms and protocols for various distributed system applications.
- Transition from thinking in terms of serial programming to designing loosely coupled, asynchronous distributed systems.
- Acquire the skills to independently design and implement moderately complex distributed systems.
- Analyze the correctness of distributed algorithms and develop custom solutions for unique scenarios.
- Gain insights into the design and engineering of contemporary distributed systems, understanding their architecture and operational principles.

13. Curriculum Content

- Introduction to Distributed Systems, Message Passing, Remote Procedure Call (RPC), System Model, The Two Generals' Problem, Byzantine Generals' Problem
- Leader Election, Distributed Models, Physical and Logical Time, Clock Synchronization Techniques, Causality, Happens-Before Relationship, Two-Phase Commit Protocol, Linearizability and Eventual Consistency Models
- Broadcast Ordering and Algorithms, Global State & Snapshot and Distributed Mutual Exclusion-Non-Token and Quorum based approaches, Distributed Mutual Exclusion-Token based approaches
- Data Replication and State Machine Replication, Token-Based Distributed Mutual Exclusion, Consensus Protocols: Raft & Agreement Techniques, Checkpointing & Rollback Recovery

- Deadlock Detection, Distributed Shared Memory (DSM), Distributed Minimum Spanning Tree (MST) Algorithms, Termination Detection, Message Ordering & Group Communication, Fault Tolerance and Self-Stabilization
- Distributed Randomized Algorithms, Distributed Hash Tables (DHT) and Peer-to-Peer (P2P) Computing
- Case Study: Google File System (GFS), Hadoop Distributed File System (HDFS), MapReduce Programming Model and Apache Spark, Sensor Networks, Authentication & Security in Distributed Systems

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Sukumar Ghosh, *Distributed Systems: An Algorithmic Approach*, 2nd Edition, Chapman and Hall/CRC Press, 2014.
- Ajay D. Kshemkalyani and Mukesh Singhal, *Distributed Computing: Principles, Algorithms, and Systems*, Cambridge University Press, 2011. ISBN: 978-0521189842.
- Gerard Tel, *Introduction to Distributed Algorithms*, 2nd Edition, Cambridge University Press, 2000.
- Hagit Attiya and Jennifer L. Welch, *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*, 2nd Edition, Wiley-Interscience, 2004.
- Vijay K. Garg, *Elements of Distributed Computing*, Wiley-Interscience, 2002.
- Andrew S. Tanenbaum and Maarten van Steen, *Distributed Systems: Principles and Paradigms*, 2nd Edition, Pearson Education, 2007.

PART A: COURSE IDENTIFIERS

1. School	SOE	
2. Department	Computer Science and Engineering	
3. Course Code	CSD4213	
4. Course Title	Embedded Systems	
5. Credits (L:T:P)	3 (3:0:0)	
6. Contact Hours (L:T:P)	3:0:0	
7. Prerequisites	CSD2003	
8. Major Core for	NA	
9. Major Elective for	CSE	

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This course is intended to familiarize the learner with the main technical concepts concerning embedded systems, and features of hardware and software components and subsystems for typical embedded systems as also with practices related to development of such systems for candidate target applications.

11. Course Objectives

The course aims to:

- Introduce the architecture and components of embedded systems.
- Develop understanding of embedded hardware and software design methodologies.
- Enable students to design and implement embedded applications using microcontrollers and single-board computers.
- Familiarize students with real-time operating systems, communication protocols, and interfacing techniques.
- Develop skills for evaluating embedded systems based on performance, reliability, and cost.

12. Learning Outcomes

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

- Explain the architecture, characteristics, and applications of embedded systems.
- Analyze embedded hardware components including processors, memory, and I/O subsystems.
- Evaluate embedded systems with respect to performance, reliability, power consumption, and cost.

13. Curriculum Content

- Introduction to Embedded Systems:
- Definition and examples of an embedded system, Difference between real and non-real time systems, Major components of an Embedded system
- Embedded Systems Components and Architecture:
- Architecture and major components of Embedded systems : Single board computers, Overview of Processors, Memory, Input/Output Devices, Special purpose processors; input-output design and I/O communication protocols
- Embedded Processors and Memory:
- Digital Signal Processors, Microcontrollers, System on Chip (SoC), Different kinds of Memory : Processor Memory, Primary Memory, Secondary Memory, Memory Interfacing, Hierarchy, Management
- Embedded Systems I/O:
- Interfacing Bus, Addressing and Protocols, Interrupts : Hardware and Software, Digital Interfacing : USB and DMA, Analog Interfacing : A/D and D/A Converters
- Embedded System Software:
- Real Time operating system issues with respect to embedded system applications; time constraints, Programming issues
- Software Engineering Issues:

- Software Engineering for Embedded Systems, Requirements Analysis and Specification, Modelling Timing Constraints, Testing of Embedded Systems

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Frank Vahid and Tony Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*, John Wiley & Sons.
- Wayne Wolf, *Computers as Components: Principles of Embedded Computing System Design*, 3rd Edition, Morgan Kaufmann, 2012.
- James K. Peckol, *Embedded Systems: A Contemporary Design Tool*, Wiley, 2008.
- Richard Zurawski (Ed.), *Embedded Systems Handbook*, 2nd Edition, CRC Press, 2018.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4214
4. Course Title	Ethical Hacking
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

From this course student will learn a comprehensive overview of ethical hacking techniques and methodologies with introduction to cryptographic primitives and important tools of network security. Students will learn how to identify vulnerabilities in systems, perform penetration testing, and implement security measures to protect information systems.

11. Course Objectives

- Enable students to understand the importance of ethical hacking and how to apply laws and regulations relevant to computer systems to practice ethical hacking.
- Fundamentals steps of ethical hacking and tools for ethical hacking.
- Identify common vulnerabilities and security challenges in computing systems.
- Analyze real-world security incidents and get exposure to develop an overall organizational security.

12. Learning Outcomes

- Understand the role of ethical hackers in cybersecurity.
- Identify common vulnerabilities, attacks and ways to secure them.
- Understand how to identify software vulnerabilities, attacks against weak passwords, web application attacks etc.
- Understand OSINT approaches and identify various sources of data that contain indicators of compromise.

13. Curriculum Content

- Introduction to ethical hacking, Hacking laws, Fundamentals of computer networking, Introduction to cryptography, private-key and public-key encryption, Cryptographic hash functions, digital signature and certificate.
- Foot-Printing, reconnaissance, Google hacking, vulnerable sites, Google as a proxy server, directory listings, locating directory listings, finding specific directories, finding specific files, Server versioning, scanning, system hacking cycle, enumeration, cracking password, type of password attacks.
- Trojan and Backdoors, types of Trojans, Viruses, Worms, Sniffers, Types of Sniffing, Phishing attack, Process of Phishing, Denial of Service, Classification of DoS attacks, System and network vulnerability, Session Hijacking, Spoofing vs Hijacking, Session Hijacking Levels, Network Level Hijacking, IP Spoofing, RST Hijacking, TCP/IP Hijacking, Hacking web servers, Web Server Defacement, Proxy and Packet filtering, SQL Injection.
- Hacking Wireless Networks, Bluetooth hacking, Mobile Phone Hacking, Tools for ethical hacking.
- Case studies: various attacks scenarios and their remedies.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References:

- William Stallings and Lawrie Brown, *Computer Security: Principles and Practice*, 3rd Edition, Pearson Education, 2015.
- James F. Kurose and Keith W. Ross, *Computer Networking: A Top-Down Approach*, 8th Edition, Pearson, 2021.
- Kevin Beaver, *Hacking For Dummies*, 7th Edition, Wiley Publishing, Inc., 2015.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4215
4. Course Title	Foundations of Data Science
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This course introduces the mathematical and computational foundations of Data Science. Students learn techniques for data preprocessing, exploratory data analysis, linear algebra, probability, statistics, optimization, and machine learning. The course covers both supervised and unsupervised learning methods and provides the theoretical foundations required for advanced studies in machine learning, artificial intelligence, and data analytics.

11. Course Objectives

The course aims to:

- Introduce the mathematical foundations underlying modern data science techniques.
- Develop understanding of statistical inference, probability, and optimization methods.
- Familiarize students with supervised and unsupervised learning algorithms.
- Enable students to analyze real-world datasets and extract meaningful insights.
- Provide the theoretical basis for advanced machine learning and artificial intelligence courses.

12. Learning Outcomes

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

- Apply data preprocessing and feature engineering techniques to prepare datasets for analysis.
- Use concepts from linear algebra, probability, and statistics to model data-driven problems.
- Implement and evaluate supervised learning algorithms for prediction and classification tasks.
- Apply unsupervised learning techniques for clustering, dimensionality reduction, and pattern discovery.
- Formulate optimization problems and apply gradient-based methods for model training.
- Analyze and interpret data science models using appropriate performance metrics.

13. Curriculum Content

- Introduction - Data preprocessing: aggregation, sampling, feature creation, selection and extraction
- Linear Algebra: Introduction to Basis, Diagonalization and Subspaces of a matrix, Symmetric and PSD Matrices
- Probability and Statistics: Random Variable, Discrete and Continuous , Probability Distribution, Multivariate Normal Distribution, MLE and MAP, Hypothesis and Inference
- Supervised Learning - Basic principle, Regression, Naïve Bayes, SVM, KNN, Decision Trees, Neural Networks
- Unsupervised Learning - Basic principle, Dimensionality Reduction: SVD, PCA, LDA, Clustering: k-means, spectral clustering, Graph learning: basics, Directed Graphs, Page Rank
- Optimization - Gradient Descent, Linear Optimization Basics

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Blum, Avrim, John Hopcroft, and Ravindran Kannan. *Foundations of data science*. Cambridge University Press, 2020.
- Grus, Joel. *Data science from scratch: first principles with python*. O'Reilly Media, 2019.
- Pang –Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar. *Introduction to Data Mining*, Second Edition, 2019
- Murphy, Kevin P. *Probabilistic machine learning: an introduction*. MIT press, 2022.
- Bishop, Christopher M., and Hugh Bishop. *Deep learning: Foundations and concepts*. Springer Nature, 2023.
- Calafiore, Giuseppe C., and Laurent El Ghaoui. *Optimization models*. Cambridge university press, 2014.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4216
4. Course Title	Foundation of Information Security
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

This course introduces the fundamental concepts of information security, including security principles, cryptography, access control mechanisms, common system and network attacks, and defense strategies. It provides an overview of practical security solutions and techniques used to protect information systems from evolving cyber threats.

11. Course Objectives

- To develop an understanding of the fundamental principles and concepts of information security.
- To enable students to analyze vulnerabilities in computing systems and design appropriate security solutions.
- To equip students with the ability to identify security threats and vulnerabilities in computer networks and communication systems.
- To enable students to evaluate and compare security mechanisms and technologies used to protect real-world information systems.

12. Learning Outcomes

Upon successful completion of this course, students will be able to:

- Explain the fundamental principles, concepts, and objectives of information security.
- Identify and analyze security threats, vulnerabilities, and risks in computing and communication systems.
- Apply appropriate security mechanisms, including cryptographic and access control techniques, to address real-world security challenges.
- Evaluate the effectiveness of security solutions and defense mechanisms against common cyber attacks.
- Recommend suitable security practices and countermeasures for protecting information systems and networks.

13. Curriculum Content

- Security Overview, CIA model, Threats, Policy and Mechanisms, Security Policies, Risk Analysis.
- Cryptography Basics, Classical Cryptosystems, Symmetric Key Cryptography, Public Key Cryptography, Digital Signature, Cryptographic Hash Functions, Digital Certificate.
- Authentication Basics, Password-based Authentication, Challenge Response Protocol, Access Control mechanisms, Confidential Policy model.
- Malicious Logic, Trojan Horses, Viruses, Worms, Logic Bombs, Defences, Sandboxing, Intrusion Detection System
- DoS and DDoS attack. Firewalls and Proxies.
- Introduction to Ethical Hacking, Types of Hackers, Ethical Hacking Phases, Types of Penetration Test.
- Overview of Security Protocols SSL/TLS and IPSec

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Matt Bishop, S.S. Venkatramanayya, “Introduction to Computer Security, 3/e”, Pearson Education
- W Stallings, “Cryptography and Network Security: Principles and Practice, 6/e”, Pearson Education
- William Stallings and Lawrie Brown, “Computer Security Principles and Practice”, 3/e, Pearson Education

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4217
4. Course Title	Generative AI
5. Credits (L:T:P)	3 (2:0:1)
6. Contact Hours (L:T:P)	2:0:2
7. Prerequisites	CSD4002
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

This course explores the principles, techniques, and applications of generative artificial intelligence, focusing on models that create content across various modalities, including text, images, and audio. Students will gain practical experience in implementing and evaluating generative models.

11. Course Objectives

- Understand key concepts of generative models.
- Gain practical experience in text generation using models like RNNs, LSTMs, and GPT.
- Learn and apply image generation techniques using VAEs, GANs, etc.
- Explore audio and video generation.
- Understand data augmentation techniques.
- Evaluate generative models with a focus on quality, bias, and ethics.

12. Learning Outcomes

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

- Understand generative AI techniques and their applications.
- Design and implement generative AI models for various tasks like text, image and video generation.
- Gain practical experience through real-world projects and case studies.
- Apply data augmentation techniques in machine learning projects.
- Evaluate generative models, focusing on performance, bias, and ethics.

13. Curriculum Content

- **Introduction to Generative AI:** Overview of generative models and their historical context. Generative vs. Discriminative Models.
- **Practical:** Assignment on Exploring Gen. AI Use Cases
- **Text Generation:** Introduction to Text Generation, Sequence-to-Sequence Models, RNN and LSTMs Models, Transformer architecture and introduction to BERT and GPT.
- **Practical:** Project to build a basic chatbot using generative text models
- **Image Generation:** Variational AutoEncoders (VAEs), Generative Adversarial Networks (GANs), Diffusion Models, Training process of VAEs and GANs and their applications in Image Generation and Image Synthesis.
- **Practical:** Project to develop a tool to generate images using GANs and VAEs
- **Audio and Video Generation and IVR System:** Introduction to Audio and Video Generation, Diffusion Models, Building an IVR System, Integration and Practical Applications.
- **Data Augmentation:** Data Augmentation Techniques, Applications in Machine Learning.
- **Evaluation of Generative Models:** Metrics for quality evaluation, including discussions on bias and ethics.

14. Assessment Scheme:

- Examination (Midterm/Final) – 50%
- Lab/Assignment/Project – 30%

- Quiz/Class participation – 20%

15. Textbooks and References

- Joseph Babcock and Raghav Bali, Generative AI with Python and TensorFlow 2: Create Images, Text, and Music with VAEs, GANs, LSTMs, Transformer Models, Packt Publishing, 2021.
- Simon J. D. Prince, Understanding Deep Learning, MIT Press, 2023.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville, Deep Learning, MIT Press, 2016.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science & Engineering
3. Course Code	CSD4219
4. Course Title	Information Retrieval
5. Credits (L:T:P)	3 (2:0:1)
6. Contact Hours (L:T:P)	2:0:2
7. Prerequisites	CSD2001
8. Major Core for	NA
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

This is an undergraduate-level introductory course for information retrieval. This course provides an in-depth introduction to the fundamental concepts, models, and techniques of Information Retrieval (IR), emphasizing the impact of web technologies on the field. It covers various IR models such as Boolean, vector-space, and probabilistic models like BM25, as well as text indexing, storage, and compression techniques. Students will learn how to evaluate IR systems using performance metrics like precision, recall, and nDCG, and explore methods for text classification and clustering. The course also introduces web search engines, including spidering, indexing, and ranking, along with emerging topics such as semantic search and knowledge graphs.

11. Course Objectives

To enable students:

- To learn the role of information retrieval in various real-time applications
- To learn and apply information retrieval models
- To design Search Engine
- To be exposed to Link Analysis
- Learn document text mining techniques

12. Learning Outcomes

On successful completion of the course, students will have good knowledge of:

- Understand the basics and evolution of Information Retrieval (IR).
- Apply various IR models, including Boolean, vector-space, and probabilistic approaches.
- Implement text indexing, storage, and compression techniques.
- Evaluate IR systems using metrics like precision, recall, and nDCG.
- Perform text classification and clustering using methods such as Naïve Bayes and k-means.
- Understand the architecture of web search engines and web page ranking.
- Address ethical issues in IR, such as bias, privacy, and handling fake news.

13. Curriculum Content

- **Introduction:** Goals and history of IR. The impact of the web on IR.
- **Basic IR Models:** Boolean and vector-space retrieval models, ranked retrieval, text similarity metrics, TF-IDF weights, cosine similarity, Introduction to Probabilistic Models, BM25, Introduction to language-model, Markov random fields.
- **Text Indexing, Storage and Compression:** Text encoding: tokenization, stemming, stop words, phrases, index optimization. Index compression: lexicon compression and postings, lists compression. Gap encoding, gamma codes, Zipf's Law. Index construction. Postings size estimation, merge sort, dynamic indexing, positional indexes, n-gram indexes, real-world issues.
- **Experimental Evaluation of IR:** Performance metrics: recall, precision, and F-measure, Evaluations on benchmark text collections, MAP, Precision Recall Curve, nDCG, and BPref.
- **Query Operations and Languages:** Relevance feedback, Query expansion, Query languages.

- **Text Classification and Clustering:** Introduction to text classification, Naïve Bayes models, Spam filtering. Vector space classification using hyperplanes, centroids, k-Nearest Neighbour classification. Clustering versus classification, Partitioning methods, k-means clustering, Hierarchical agglomerative clustering and Clustering terms using documents.
- **Web Search:** Search engines, spidering, indexing, link analysis, web page ranking.
- **Introduction to Semantic Search:** Semantic Web and Ontologies, Knowledge Graphs in IR, Basics of Entity Recognition and Disambiguation.
- **Responsible IR:** handling bias and fake-news, privacy, etc.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Lab/Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.
- Ricardo Baeza-Yates and Berthier Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology behind Search*, 2nd Edition, Addison-Wesley/ACM Press, 2011.
- Bruce Croft, Donald Metzler, and Trevor Strohman, *Search Engines: Information Retrieval in Practice*, Pearson Education, 2010.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4220
4. Course Title	Internet of Things
5. Credits (L:T:P)	3 (2:0:1)
6. Contact Hours (L:T:P)	2:0:2
7. Prerequisites	CSD2001, CSD4001
8. Major Core for	NA
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

In recent years, IoT has evolved as a promising solution to extend coverage where the cellular base stations can not provide services. IoT is expected to be integrated with the traditional cellular networks such as long-term evolution (LTE) and new radio (NR). The IoT technology is well known for its wide range of applications such as healthcare, agriculture, industrial automation etc. In this course, a comprehensive overview of different IoT technologies and their applications have been provided; Moreover, analyses and implementation perspectives of IoT systems have also been discussed.

11. Course Objectives

The aims of this course are:

- To understand the IoT Protocol stack and its difference from TCP/IP protocol suite.
- To understand IoT Sensing and actuation, processing topologies, network technologies (e.g., IEEE 802.15.4, Zigbee, LoRa, NB-IoT etc.) and communication protocols (Infrastructure protocols, discovery protocols, data protocols and identification protocols) used in IoT, Fog computing.
- To explore the hardware platforms (e.g., Arduino), simulation using tinker cad simulator.
- To provide a comprehensive overview on the security issues in IoT.
- To introduce Machine Learning techniques to analyse IoT systems.
- To explore some use cases of IoT.

12. Learning Outcomes

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

- **Understand** the emerging network technologies used in IoT and their application domains.
- **Demonstrate** IoT applications based on Arduino.
- **Understand** different sensing mechanisms in IoT.
- **Understand** the security issues in IoT.
- **Explore** different application scenarios of IoT (e.g., healthcare).

13. Curriculum Content

- *Module-1 (IoT Technologies):*
Internet of Things (IoT) architecture, the emerging network technologies used in IoT, common application domains, IoT Sensing and actuation, IoT technologies (e.g., IEEE 802.15.4, Zigbee, LoRa, NB-IoT etc.), IoT communication protocols (Infrastructure protocols, discovery protocols, data protocols and identification protocols), Health care IoT, Fog computing: formulation of service placement problem as an optimization problem, an overview on solution methodologies.
- *Module-2 (Arduino programming):*
Introduction to Arduino programming and tinker cad simulator (Applications: Blink LED, printing message, reading from the buffer, integrating analog sensors with Arduino and display the output on serial monitor, communication between Arduino board etc.).
- *Module-3 (IoT analytics):*
Introduction to IoT Analytics: review of probability theory, simulation techniques, Introduction to ML, selected algorithms in ML: k-nearest neighbour (KNN), random forest, Agglomerative clustering.
- *Module-4 (Physical layer security):*

Definition, notion of secrecy throughput, different types of security threats in IoT (e.g. Eavesdropping, Spoofing, Jamming).

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- S. Mishra, A. Mukherjee and A. Roy, “Introduction to IoT”, Cambridge University press, 2021.
- H. G. Perros, “An introduction to IoT analytics”, CRC press, Taylor and Francis group, 2021.
- D. Hanes, G. Salgueiro, P. Grossetete, R. Barton, J. Henry, “IoT Fundamentals: Networks technologies, protocols and use cases”, Pearson.
- Research papers.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering	
2. Department	Computer Science and Engineering	
3. Course Code	CSD4223	
4. Course Title	Mobile Computing	
5. Credits (L:T:P)	3 (3:0:0)	
6. Contact Hours (L:T:P)	3:0:0	
7. Prerequisites	CSD4001	
8. Major Core for	NA	
9. Major Elective for	CSE	

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

In this course, a comprehensive overview of different wireless communication systems (e.g., 4G, 5G, WIFI etc.) is provided, and important issues such as mobility management, channel assignment and interference management for capacity enhancement of cellular systems is discussed. Moreover, various use cases (e.g., vehicular network, device-to-device communication) will be explored through state-of-the-art simulator such as ns-3/OMNET++.

11. Course Objectives

The aims of this course are:

- Understanding different wireless standards (4G, 5G, 6G and WiFi).
- Understanding Multiple access techniques for different wireless communication technologies (e.g., WCDMA, OFDMA, NOMA, CSMA/CA etc.).
- Understanding the Device-to-Device communication technology (LTE D2D).
- Understanding security threats in wireless communication systems.
- Exploring some use cases using state-of-the-art simulators such as OMNET++.

12. Learning Outcomes

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

- **Understanding** various aspects of wireless network technologies.
- **Understanding** different cellular standards and WIFI.
- **Understanding Device**-to-device technology and its application areas.
- **Exploring** different use cases using OMNET++ simulator.
- **Understanding** security threats in wireless networks and their solutions.

13. Curriculum Content

- *Module-1: Introduction to wireless system*
Evolution of mobile radiocommunication, Concept of Frequency reuse, Channel assignment strategies, Handoff strategies: prioritizing handoffs, practical handoff considerations, Interference and system capacity, Improving coverage and capacity in cellular systems: Cell splitting, sectoring, large scale path loss, small scale fading and multipath.
- *Module-2: Multiple access techniques for wireless communication*
Frequency division multiple access (FDMA), Time division multiple access (TDMA), Code division multiple access (CDMA), Space division multiple access (SDMA), ALOHA, Pure ALOHA, Slotted ALOHA, CSMA/CA, Spread spectrum access techniques, Capacity analysis of cellular systems.
- *Module-3: Cellular standards & Wi-Fi*
Network architecture of 2G, 3G, 4G, 5G and Wi-Fi; packet scheduling algorithms, Mobility management.
- *Module-4: Device to Device communication*
Definition, Application scenarios, Type of D2D communication, Interference in D2D communication, Pricing algorithm, Interference management in LTE-A enabled D2D communication.

14. Assessment Scheme:

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- **Theodore S. Rappaport**, *Wireless Communications: Principles and Practice*, 2nd Edition, Pearson Education, 2002.
- **James F. Kurose and Keith W. Ross**, *Computer Networking: A Top-Down Approach*, 8th Edition, Pearson, 2021.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4224
4. Course Title	Natural Language Processing
5. Credits (L:T:P)	3(2:0:1)
6. Contact Hours (L:T:P)	2:0:2
7. Prerequisites	CSD2001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

This undergraduate course offers an introduction to Natural Language Processing (NLP), covering the essential concepts, techniques, and algorithms used for processing and analyzing natural language data. The course will explore various linguistic structures, machine learning approaches, and popular NLP applications such as text classification, sentiment analysis, and machine translation.

11. Course Objectives

- Understand the fundamentals of NLP and the linguistic basis of language processing.
- Explore machine learning algorithms for NLP, including supervised and unsupervised approaches.
- Apply NLP techniques for tasks such as text pre-processing, classification, named entity recognition, and translation.
- Implement and evaluate models using NLP libraries and frameworks.

12. Learning Outcomes

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

- Understand the core concepts and techniques used in NLP.
- Develop practical solutions for NLP tasks such as text classification, machine translation, and question answering.
- Implement machine learning models and apply them to language data.
- Evaluate the performance of NLP models and understand their limitations.
- Gain experience with Python-based NLP libraries such as NLTK, spaCy, and Hugging Face Transformers.

13. Curriculum Content

- **Introduction to Natural Language Processing:** Overview of NLP and its applications, Brief introduction to linguistics: syntax, semantics, and pragmatics.
- **Practical:** Setting up Python, NLTK/spaCy, and exploring simple NLP tasks
- **Text Pre-processing Techniques:** Tokenization, stemming, lemmatization, Stop-word removal, text normalization.
- **Language Models and N-Grams:** Introduction to language models, N-gram models: unigrams, bigrams, trigrams, Smoothing techniques for language modelling
- **Practical:** Implementing N-gram models and text generation
- **Word Embeddings:** Vector space models, TF-IDF, Word2Vec, GloVe, fastText
- **Practical:** Generating and visualizing word embeddings using pre-trained models
- **Text Classification and Sentiment Analysis:** Text classification techniques: Naive Bayes, SVM, logistic regression, Sentiment analysis and use cases
- **Practical:** Implementing sentiment analysis using machine learning models
- **Sequence Models in NLP:** Introduction to sequence models: Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs), Applications: part-of-speech tagging, named entity recognition (NER)

- **Introduction to Neural Networks for NLP:** Basics of neural networks and deep learning for NLP, RNNs, LSTMs, and GRUs
- **Practical:** Implementing RNNs for text generation tasks
- **Attention Mechanisms and Transformers:** Introduction to attention and self-attention, Overview of the Transformer architecture
- **Machine Translation:** Phrase-based and neural machine translation, Encoder-decoder models and attention-based models
- **Practical:** Building a simple machine translation model using Transformer
- **NLP for Question Answering and Summarization:** Techniques for question answering systems, Abstractive vs extractive text summarization
- **Practical:** Implementing text summarization using pre-trained Transformer models
- **Topic Modelling and Document Clustering:** Latent Dirichlet Allocation (LDA) and other topic modelling methods, Document clustering with k-means and hierarchical clustering
- **Ethics and Challenges in NLP:** Ethical considerations in NLP: bias, privacy, fairness, Limitations and challenges of current NLP models

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Lab/Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- **Daniel Jurafsky and James H. Martin**, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd Edition, 2025.
- **Hobson Lane, Cole Howard, and Hannes Max Hapke**, *Natural Language Processing in Action*, 2nd Edition, Manning Publications, 2023.
- **Steven Bird, Ewan Klein, and Edward Loper**, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*, O'Reilly Media, 2009.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	CSE
3. Course Code	CSD4225
4. Course Title	Object-Oriented Analysis and Design
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

This course on object oriented design describes the fundamental principles of object-oriented approaches to model software requirements and design. The initial focuses will emphasis on how object-oriented analysis and design is different from the traditional structured design approaches. Understanding the concepts of an object model, its elements and the use of Unified Modeling Language (UML) facilitates. This course also examines the pragmatics of object-oriented development in the context of software development life cycle, particularly during project management on how the object model reduces the risks inherent in developing complex systems.

11. Course Objectives

To familiarize the students with object-oriented analysis and design techniques and enable them to solve problems using these techniques. To familiarize the students with important design patterns and teach them how to apply in solving various problems.

12. Learning Outcomes

- Compare the object-oriented design from traditional structured design approaches with respect to building flexible, reusable & robustness.
- Create an object model based on the user's requirements using visual modeling tools (UML) to establish explicit coupling between conceptual and executable artifacts
- Use design patterns wherever appropriate in a design

13. Curriculum Content

- **MODULE-1 INTRODUCTION:**
Basic concepts, abstraction, encapsulation, information hiding, inheritance, dynamic binding, polymorphism, overview of OOAD.
- **MODULE-2 UNIFIED MODELLING LANGUAGE (UML):**
UML views and diagrams, Use case modelling, actors and use cases, factoring use cases, Class diagrams, class relations, association, inheritance, aggregation/composition, inheritance, dependency, Packages, Interaction diagrams, sequence diagrams, fragments, Communication diagram, State diagram, events, guards, composite states, concurrent states, history state, activity diagram, swim lanes, events, messages, object flow.
- **MODULE-3 OBJECT-ORIENTED DESIGN PROCESS:**
Overview of the design process, Domain modelling, identifying objects, boundary objects, control objects, entity objects, CRC cards, CASE support.
- **MODULE-4 BASIC PRINCIPLES:**
SOLID principles, Single Responsibility Principle (SRP), Open-Close Principle (OCP), Liskov Substitution principle (LSP), Interface segregation Principle (ISP), Dependency Inversion Principle (DIP), Martin's Package metrics, CK metrics.
- **MODULE-5 DESIGN PATTERNS:**
Overview of patterns, Architectural, design, and code patterns, GRASP and GoF patterns, Expert, Creator, Law of Demeter, Controller, Singleton, Model View Separation patterns, Observer, MVC,

Publish-Subscribe, Singleton, State, Composite, Façade, Decorator, Proxy, Bridge, Strategy, Mediator, Visitor, Iterator, Flyweight, Template, Memento.

14. Assessment Scheme:

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, and Kelli A. Houston, *Object-Oriented Analysis and Design with Applications*, 3rd Edition, Addison-Wesley Professional, 2007.
- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, 1994.
- Robert C. Martin, *UML for Java Programmers*, Prentice Hall PTR, 2003.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	CSE
3. Course Code	CSD4227
4. Course Title	Real Time Systems
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2005
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

In several software applications, especially in embedded application, the operating system is required to support the application to meet the timing constraints. The operating system achieves this by deploying suitable scheduling algorithms. A major problem arises, when the real-time tasks share resources. Priority inversions can take place in this case, unless suitable techniques are deployed. Starting with a brief introduction to real-time operating systems, we first discuss the important real-time task/thread scheduling algorithms and resource sharing protocols. An effort towards standardization of real-time operating systems has come to be known as POSIX-RT. We review POSIX-RT requirements. Besides, we review several commercial and open source real-time operating systems.

11. Course Objectives

To teach the fundamental issues that arise in developing real-time applications and the features of a real-time operating system.

12. Learning Outcomes

- Compare different scheduling algorithms and the schedulability criteria.
- Determine schedulability of a set of periodic tasks given a scheduling algorithm and develop algorithms to decide the admission criterion of sporadic jobs and the schedule of aperiodic jobs.
- Integrate resource access mechanisms with the scheduling techniques and develop integrated schedulability criteria.
- Design real time embedded systems using the concepts of RTOS

13. Curriculum Content

- **MODULE-1 INTRODUCTION:**
Types of real time applications: Hard and soft real time systems, Types of real-time tasks: periodic, aperiodic, and sporadic, timing constraints, A reference model of Real-time systems, modelling time constraints, precedence constraints & dependencies
- **MODULE-2 TASK SCHEDULING:**
Cyclic and priority driven approaches, Clock Driven Scheduling: Static timer driven scheduler, Cyclic Executives, Improving Average Response times of Aperiodic Jobs, Scheduling Sporadic jobs, Practical Considerations, Pros and Cons of Clock Driven Scheduling. Priority-driven scheduling of periodic tasks: Fixed Priority vs Dynamic Priority schemes, rate monotonic (RM) and earliest deadline first (EDF) schedulers, Maximum schedulable Utilization, Optimality of the RM and DM algorithms, Schedulability Test for Fixed Priority Tasks, Practical Factors. Scheduling Aperiodic and Sporadic Jobs in Priority-driven scheduling: Deferrable Servers, Sporadic Servers, and Constant Utilization. Total Bandwidth, and Weighted Fair-Queueing Servers, Scheduling of Sporadic Jobs
- **MODULE-3 RESOURCES AND RESOURCE ACCESS CONTROL:**
Non-preemptive critical sections, priority inversion, basic priority-inheritance protocol, highest locker protocol, priority ceiling protocol, multiprocessor scheduling, predictability and validation of dynamic multiprocessor systems flexible applications, tasks with temporal constraints.
- **MODULE-4 REAL TIME OPERATING SYSTEMS:**

Time Services and Scheduling Mechanisms, Basic Operating System Functions, Open System Architecture, POSIX and POSIX-RT, Capabilities of Commercial RTOS, benchmarking real-time systems.

- **MODULE-5 REAL-TIME COMMUNICATION AND REAL-TIME DATABASES:**

Real-time traffic, real-time communication in a LAN and over packet-switched networks, quality of service. Introduction to real-time database, temporal data, concurrency control in real-time databases.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Rajib Mall, Real-Time Systems, Pearson Education, 2010
- Krishna, C.M. and Shin, K.G., Real Time Systems, 3rdEd., Tata McGraw Hill, 2010.
- Liu, J.W.S., Real-Time Systems, Pearson Education, 2013.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4226
4. Course Title	Parallel and Concurrent Programming
5. Credits (L:T:P)	3 (2:0:1)
6. Contact Hours (L:T:P)	2:0:2
7. Prerequisites	CSD2005
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

Nowadays, all computing machines (cell phones, laptops, computation servers) have multi-core chips in them. These multi-core processors can be seen as parallel processors. Now, to harness the power of the multi-core hardware, the software applications being developed also have to be parallelized.

This course will teach students how to harness the power of widely available multi-core processors using parallel programming techniques. The course will focus on developing various Concurrent Data Structures such as stacks, queues, matrices, trees, lists, hash tables, and graphs while proving their correctness through serializability and linearizability.

11. Course Objectives

- Equip students with the knowledge and skills to design, implement, and analyze programs that can execute multiple tasks simultaneously.
- Students will learn to effectively utilize multi-core processors, and other parallel computing environments to enhance the performance and efficiency of their applications.
- The course will cover key concepts such as parallel algorithms, synchronization, thread management, and concurrent data structures, preparing students to tackle real-world challenges in high-performance and scalable software development.

12. Learning Outcomes

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

- Describe and implement fundamental concepts of concurrency and parallelism in programming.
- Develop effective solutions for parallel and concurrent programming challenges using suitable algorithms.
- Apply performance optimization techniques to enhance the parallel efficiency and scalability of programs.
- Discuss current and emerging trends in parallel architectures and programming models.
- Demonstrate a blend of theoretical understanding and practical experience in parallel and concurrent programming.

13. Curriculum Content

- **Unit-I:** Overview of Parallel and Concurrent Programming, Differences between parallelism and concurrency, Parallel computing models (Shared memory, Distributed memory), Speedup, Efficiency, and Scalability, Amdahl's Law, Introduction to multicore architectures
- **Unit-II:** Concurrency Concepts: Processes and Threads, Synchronization primitives (Locks, Semaphores, Monitors), Deadlock, Livelock, Starvation, Thread safety and race conditions, Mutual Exclusion
- **Unit-III:** Task and Data Parallelism, Defining and Working with Concurrent Objects, Sequential consistency, and Linearizability, Shared Memory Programming: PThreads, OpenMP, TBB
- **Unit-IV:** Concurrent Data Structures (CDS): Linked Lists, Queues, Stacks, Counters, Trees, and Hash Tables.
- **Unit-V:** Design Considerations for CDS: Coarse-Grained vs. Fine-Grained Concurrency Control, Lock-free CDS, Wait-free CDS, and Obstruction-Free

- **Unit-VI: GPU Programming:** GPU architectures, CUDA programming model, Parallel Graphs and Matrix Algorithms

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Maurice Herlihy and Nir Shavit, *The Art of Multiprocessor Programming*, Revised Reprint Edition, Morgan Kaufmann, 2012.
- Gregory R. Andrews, *Multithreaded, Parallel, and Distributed Programming*, Addison-Wesley Professional, 2000.
- Calvin Lin and Lawrence Snyder, *Principles of Parallel Programming*, Addison-Wesley, 2008.
- Ananth Grama, George Karypis, Vipin Kumar, and Anshul Gupta, *Introduction to Parallel Computing*, 2nd Edition, Addison-Wesley, 2003.
- Peter Pacheco, *An Introduction to Parallel Programming*, Morgan Kaufmann, 2011.
- Wen-mei W. Hwu, David B. Kirk, and Izzat El Hajj, *Programming Massively Parallel Processors: A Hands-on Approach*, 4th Edition, Morgan Kaufmann, 2022.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4228
4. Course Title	Reinforcement Learning
5. Credits (L:T:P)	3 (2:0:1)
6. Contact Hours (L:T:P)	2:0:2
7. Prerequisites	CSD2006
8. Major Core for	None
9. Major Elective for	CSE, EE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

Traditionally, the optimal control of engineering systems is achieved by the prevalent methodology of offline optimization. Various control problems are first formulated as different kinds of optimization problems and then solved by the carefully handcrafted algorithms, using mathematical tools from optimization theories, such as convex optimization, game theory, dynamical programming, etc. Moreover, these algorithms are periodically executed to find the optimal configuration according to the current situation of the system. Nevertheless, such offline algorithms are insufficient to handle increasingly complex control problems. The main deficiencies lie in high complexity, low robustness, and high overhead.

Recently, reinforcement learning (RL) has emerged as a promising solution to deal with the complex optimization problems in real engineering scenarios. Apart from that RL is desirable for learning network dynamics from raw observations and learning in distributive fashion based on local observations.

In this course, the objective is to introduce different aspects of RL problems and to introduce different model free RL techniques; Moreover, some case studies will also be done.

11. Course Objectives

The aims of this course are:

- To **understand** the aspects of a Reinforcement learning problem: Agent, Environment, Finite Markov Decision Process, Dynamic Programming.
- To apply Tabular solution methods to different engineering problems: Multi-arm Bandits, Monte Carlo Methods, Temporal difference learning, eligibility traces.
- To apply approximate solution methods to different engineering problems: On-policy approximation of action values, off-policy approximation of action values.
- To understand cutting-edge deep RL algorithms such as Rainbow, PPO and SAC.
- Case studies on Dynamic channel allocation and Job-Shop scheduling.

12. Learning Outcomes

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

- **Understand** the aspects of Reinforcement learning (RL) problems.
- **Apply** tabular solution methods to different engineering problems.
- **Apply** approximate solution methods to different engineering problems.
- **Apply** deep RL algorithms to different engineering problems.

13. Curriculum Content

- Introduction: Elements of a Reinforcement learning (RL) problem (states, actions, reward, agent, environment), examples, Formulation of RL problems as Markov decision process (MDP), Solving MDP using value iteration and policy iteration, importance of model-free learning, discussion on Tabular solution methods and Approximate solution methods.
- Multi-arm bandit: n -armed bandit problem, action-value methods, incremental implementations, Tracking a nonstationary problem, Gradient bandits.
- Monte Carlo Methods: Monte Carlo (MC) prediction, MC estimation of action values, MC control, MC control without exploring starts, Off-policy prediction.

- Temporal-difference (TD) learning: TD prediction, advantages, Optimality of TD (0), SARSA, Q-learning, discussion on eligibility traces.
- Approximate solution methods: Value prediction with function approximation, Gradient-descent method, Linear methods, Control with function approximation, Bootstrap.
- Deep reinforcement learning (DRL): the DRL problem, Algorithms for DRL such as Rainbow, PPO and SAC.
- Case studies: Dynamic channel allocation and Job-Shop scheduling.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%
- Assignment/Project/Lab – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning: An Introduction”, The MIT press, 2015.
- Phil Winder, Reinforcement Learning, O’Reilly, 2020 [ISBN: 9781098114831].

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4229
4. Course Title	Secure Coding
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE**10. Course Summary**

From this course student will learn a comprehensive overview of secure coding practices specifically for C and C++. It will help to identify common vulnerabilities, implement secure coding techniques, and apply tools for testing and analysis. It will enable students to understand how software coding defects lead to software vulnerabilities and to develop secure software.

11. Course Objectives

- To provide a detailed explanation of common programming errors in C and C++ and describe how these errors can lead to code that is vulnerable to exploitation.
- To teach secure software development tools and processes demonstrating security issues intrinsic to the C and C++ programming languages and associated libraries.

12. Learning Outcomes

- Understand key vulnerabilities associated with C and C++ programming.
- Implement secure coding practices to mitigate risks.
- Improve the security of any C or C++ application.
- Identify, exploit, and eliminate insecure input manipulation logic.

13. Curriculum Content

- **Unit-I:** Introduction to Secure Coding, Secure coding principles, Importance of security in C and C++, Common security threats.
- **Unit-II:** Memory Management Vulnerabilities: Buffer overflows, Memory leaks, Use of dynamic memory and associated risks.
- **Unit-III:** Input Validation: Techniques for validating input, Sanitization and filtering, Strategies for safe string handling
- **Unit-IV:** Output Encoding: Proper handling of output to prevent injection attacks, Encoding techniques for different contexts (HTML, URLs, etc.).
- **Unit-V:** Error Handling and Logging: Effective error handling strategies, Logging practices that protect sensitive information
- **Unit-VI:** Security and Cryptography: Basic principles of cryptography in secure coding, Implementing secure encryption and hashing methods
- **Unit-VII:** Secure Coding Standards and Guidelines: Overview of relevant standards (e.g., CERT, MISRA), Best practices for coding and design.
- **Unit-VIII:** Testing and Tools: Techniques for testing code for vulnerabilities, Static and dynamic analysis tools.
- **Unit-IX:** Case Studies and Real-World Examples: Analysis of security incidents related to C and C++ applications, Lessons learned from historical vulnerabilities

14. Assessment Scheme:

- Examination (Midterm/Final) – 50%
- Assignment/Project – 30%

- Quiz/Class participation – 20%

15. Textbooks and References

- Secure Coding in C and C++, Second Edition by Robert C. Seacord, Addison-Wesley Pearson Education.

PART A: COURSE IDENTIFIERS

1. School	School of Engineering
2. Department	Computer Science and Engineering
3. Course Code	CSD4230
4. Course Title	Social and Information Networks
5. Credits (L:T:P)	3 (3:0:0)
6. Contact Hours (L:T:P)	3:0:0
7. Prerequisites	CSD2001
8. Major Core for	None
9. Major Elective for	CSE

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

In the Social and information networks course, we will learn efficient techniques designed for analysis of large network data, models that represent these real-world phenomena and combinatorial/probabilistic techniques.

11. Course Objectives

The aim of this course is to enable students to:

- Understand how the social and information networks can be represented through graphs and find important properties of a graph.
- Understand the structure of the web.
- Appreciate the concept of information cascades and their versatile applications.
- Learn to analyse different information networks – eg. biological networks, congestion networks, etc.
- Learn heterogeneous graph representation and analysis.

12. Learning Outcomes

After successful completion of this course, students will be able to:

- Understand the theoretical concepts covering graph theory, abstraction of real-world networks into graphs and their analysis.
- Understand the combinatorial or probabilistic approaches used for analyzing these networks and drawing inferences.
- Design a framework for a real-world problems that can be represented using graphs.

13. Curriculum Content

- Introduction Relevant concepts in graphs – basic definitions, paths, traversal, connectivity
Application areas
- Network Datasets
- Tie strength
- Network structure in large-scale data
- Homophily, Affiliation, Spatial model of segregation
- Positive and Negative Relationships
- The structure of the Web,
- Power laws, Preferential attachment
- Link Analysis and Web search Information cascades,
- Epidemics
- Finding communities
- Modularity Building and analysing information networks
- Congestion networks – Game theory approach
- Select advanced topics.

14. Assessment Scheme

- Examination (Midterm/Final) – 50%

- Assignment/Project – 30%
- Quiz/Class participation – 20%

15. Textbooks and References

- David Easley and Jon Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, Cambridge University Press, 2010.
- Reza Zafarani, Mohammad Ali Abbasi, and Huan Liu, *Social Media Mining: An Introduction*, Cambridge University Press, 2014.
- Mark Newman, *Networks*, Oxford University Press, 2018

PART A: COURSE IDENTIFIERS

1. School	School of Engineering	
2. Department	Computer Science and Engineering	
3. Course Code	CSD4236	
4. Course Title	Connected Car Technology	
5. Credits (L:T:P)	(2:0:1):3	
6. Contact Hours (L:T:P)	(2:0:1):3	
7. Prerequisites	CSD4001	
8. Major Core for	NA	
9. Major Elective for	Computer Science and Engineering	

PART B: OBJECTIVES AND PRACTICE

10. Course summary

This course introduces students to Wireless networking technologies for autonomous vehicle, Onboard networking for connected car technology and advanced driver assisted system; and to make them industry ready for MG motors.

11. Course Objectives

The main aims of this course are:

- **Introducing** students to Wireless networking technology for autonomous vehicles.
- **Introducing** students Onboard networking technologies for connected car technology.
- **Introducing** students to advanced driver assisted system.
- **Enabling** students to design and implement new AI/ML algorithms relevant to autonomous vehicle.

12. Learning outcomes

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

- **Understand** the computer networking fundamentals for EV.
- **Understand** the wireless technologies for EV.
- **Understand** the advanced driver assisted system.
- **Design AI/ML** algorithms to develop new EV technologies.
- **Explore** different features of EV through state-of-the art simulators.

13. Curriculum content

Module-1: *Computer networking fundamental:*

Basics of Computer Networking and Internet of Things, Idea of TCP/IP protocol suite and IoT stack, Wireless Networking Fundamentals: pathloss, fading, shadowing effects, Overview to 2G to 5G Networks, Services in 5G: enhanced mobile broadband (eMBB), ultra reliable low latency (uRLLc) and massive machine type communication (mmTC), Mobility management, challenges for uRLLc (Automotive Application) in 5G.

Module-2: *Wireless networking for Autonomous vehicle:*

Transmission & Receiver Systems, Radio Transmission, Concepts for Automotive Application, Wireless Networking and Applications to Vehicle Autonomy: C-V2X operation modes, IEEE 802.11p, Semipersistent algorithm for side link selection (both Mode 1 and Mode 2) in 5G enabled autonomous vehicle Differences inside link selection mechanisms between LTE and NR systems, Standards for Autonomous Vehicle Applications, Use case on Vehicular IoT: understanding connectivity fundamentals (Car to Networks and within Car), Navigation and Other Applications, Applications of Machine Learning in V2X.

Module-3: *Onboard networking for connected car technology:*

Integration of Wireless Networking and On-Board Vehicle Networks, Concept of Network mobility, Introduction to Network mobility basic support protocol (NEMO BSP), Exploration of different V2X applications (Vehicle to vehicle, Vehicle to Roadside and Vehicle to Infrastructure applications), Application layer security aspects in LTE V2X.

Module-4: *Introduction to advanced driver assisted system:*

In Car Assistance, Multimedia and Infotainment, Android Auto/ Apple Car play, Car as a Platform, Fastag, GPS, Introduction to Automotive Cybersecurity, building a connected Vehicle Platform connecting vehicles, storing and analysing data and building consumer application as a Case Study, AI/ML algorithms for advanced driver assisted system.

LAB EXPERIMENTS:

- Demonstration of V2I, V2X and V2V communication using state-of-the-art simulator such as OMNET++/ns 3.
- Demonstration/performance analysis of Semipersistent algorithm using state-of-the-art simulator such as OMNET++/ns 3.
- Demonstration of ADAS using Digital Twin.

14. Assessment Scheme

Midterm: 20%.

Quiz Test (internal): 20%

Lab: 20%

End term: 40%.

15. Textbooks and References

- Computer Networking: A Top-down approach, 8th Edition, by Kurose and Rose.
- IoT Fundamentals: Networking Technologies, Protocols and Use cases for Internet-of-things by D. Hanes, Pearson Indian education services.
- Materials given by JSW MG Motors.

Appendix – I : List of courses (Core, Experiential and Electives)**List of courses in Experiential Learning category**

Course code	Title	Credits	Semester	Prerequisites
SOE 4801	Introduction to Engineering: Ideas to Impact	2	1	NA
SOE 4802	Design to Reality: CAD and 3D Printing	2	2	NA
CSD4801	Introduction to Database Systems	4	4	CSD2001
		8		

List of courses in Core category

Course code	Title	Credits	Semester	Prerequisites
CSD1001	Problem Solving Using Programming	4	1	NA
PHY1011	Fields, Waves & Quanta	5	1	NA
MAT1003	Multivariate Calculus	4	1	NA
ECE1001	The Electron's Path: Fundamentals of EEE	3	1	NA
CHD1001	Nature's Code: Chemistry & Biology	2	1	NA
ECE1002	Connected Intelligence: Sensors and IoT	3	2	NA
MED1001	The Matter of Materials	2	2	NA
MAT1004	Linear Systems and Transforms	4	2	NA
CED1001	Forces in Action	3	2	NA
CSD2001	Data Structures and Algorithms	4	3	CSD1001
CSD2002	Discrete Mathematics	4	3	CSD1001
CSD2003	Computer Organization	5	3	CSD1001
CSD2004	Object Oriented Programming	4	3	CSD1001
CSD2005	Operating Systems	4	4	CSD2001, CSD2003
CSD2006	Introduction to Probability and Statistics	4	4	NA
CSD2007	Applied Linear Algebra	4	4	NA
CSD4001	Computer Networks	4	5	CSD2005
CSD4002	Artificial Intelligence and Machine Learning	4	5	CSD2007
CSD4003	Formal Languages and Automata Theory	4	6	CSD2001
CSD4004	Advanced Algorithms	4	6	CSD2001
CSD4005	Advanced Computer Architecture	4	6	CSD2005, CSD2003
		79		

List of electives:

Course code	Title	Credits	Prerequisites
CSD4201	Advanced Database Management System	3 (3:0:0)	CSD4801
CSD4237	AI Applications	3 (3:0:0)	Knowledge of Probability and Statistics and Machine Learning
CSD4203	Algorithms for Big Data	3 (3:0:0)	CSD2001
CSD4204	Applied Cryptography	3 (3:0:0)	CSD1001, CSD2002
CSD4205	Blockchain Technology	3 (3:0:0)	CSD2001
CSD4206	Cloud Computing	3 (2:1:0)	CSD1001
CSD4207	Compiler Design	3 (3:0:0)	CSD2001

CSD4208	Computer Graphics	3 (2:0:1)	CSD2001
CSD4209	Computer Vision	3 (2:0:1)	CSD2001
CSD4210	Deep Learning	3 (2:0:1)	CSD4002
CSD4211	Image Processing and Applications	3 (2:0:1)	CSD2002
CSD4212	Distributed Systems	3 (3:0:0)	CSD2001, CSD2005
CSD4213	Embedded systems	3 (3:0:0)	CSD2003
CSD4214	Ethical Hacking	3 (3:0:0)	CSD2001
CSD4215	Foundations of Data Science	3 (3:0:0)	CSD2001
CSD4216	Foundation of Information Security	3 (3:0:0)	CSD2001
CSD4217	Generative AI	3 (2:0:1)	CSD4002
CSD4219	Information Retrieval	3 (2:0:1)	CSD2001
CSD4220	Internet of Things	3 (2:0:1)	CSD2001, CSD4001
CSD4221	Optimization Techniques	3(3:0:0)	CSD2001
CSD4223	Mobile Computing	3 (3:0:0)	CSD4001
CSD4224	Natural Language Processing	3 (2:0:1)	CSD2001
CSD4225	Object Oriented Analysis and Design	3 (3:0:0)	CSD2001
CSD4226	Parallel and Concurrent Programming	3 (2:0:1)	CSD2005
CSD4227	Real Time Systems	3 (3:0:0)	CSD2005
CSD4228	Reinforcement Learning	3 (2:0:1)	CSD2006
CSD4229	Secure coding	3 (3:0:0)	CSD2001
CSD4230	Social and Information Networks	3 (3:0:0)	CSD2001
CSD4231	Special Topics in Applications	3 (3:0:0)	CSD2001
CSD4232	Special Topics in Artificial Intelligence	3 (3:0:0)	CSD2001
CSD4233	Special Topics in Cybersecurity	3 (3:0:0)	CSD2001
CSD4234	Special Topics in Systems and Networks	3 (3:0:0)	CSD2001
CSD4235	Special Topics in Data Science	3 (3:0:0)	CSD2001
CSD4236	Connected Car Technologies	3 (2:0:1)	CSD4001

Appendix-II: List of specializations

Areas of Specialization

The students enrolled in B. Tech. Computer Science and Engineering (4 year) would have an option to specialize in one the following emerging areas-

1. Artificial Intelligence
2. Cybersecurity
3. Data Science
4. Systems and Networks

Minimum Requirement for Specialization:

The student wishing to do specialization must have overall CGPA ≥ 6 and CGPA in Specialization component ≥ 7 .

The student must complete minimum of 12 credits from the list of elective courses from the chosen specialization bucket.

At the time of graduation (end of 8th semester before convocation), students who have completed the specialization requirement may apply for a specialization in CSE to UG advisor for further processing. A student can apply for a specialization in only one of the mentioned areas.

List of Elective courses in specialization buckets - Artificial Intelligence

Course code	Title	Credits	Prerequisites
CSD4237	AI Applications	3	Knowledge of Probability and Statistics and Machine Learning
CSD4209	Computer Vision	3	CSD2001
CSD4210	Deep Learning	3	CSD4002
CSD4211	Image Processing and Applications	3	CSD2002
CSD4219	Information Retrieval	3	CSD2001
CSD4217	Generative AI	3	CSD4002
CSD4224	Natural Language Processing	3	CSD2001
CSD4228	Reinforcement Learning	3	CSD2006
CSD4232	Special Topics in Artificial Intelligence	3	CSD2001

List of Elective courses in specialization buckets - Cybersecurity

Course code	Title	Credits	Prerequisites
CSD4204	Applied Cryptography	3	CSD1001, CSD2002
CSD4205	Blockchain Technology	3	CSD2001
CSD4206	Cloud Computing	3	CSD1001
CSD4213	Embedded systems	3	CSD2003
CSD4214	Ethical Hacking	3	CSD2001
CSD4216	Foundation of Information Security	3	CSD2001
CSD4220	Internet of Things	3	CSD2001, CSD4001
CSD4223	Mobile Computing	3	CSD4001
CSD4229	Secure coding	3	CSD2001
CSD4233	Special Topics in Cybersecurity	3	CSD2001

List of Elective courses in specialization buckets - Data Science

Course code	Title	Credits	Prerequisites
CSD4201	Advanced Database Management System	3	CSD4801
CSD4203	Algorithms for Big Data	3	CSD2001
CSD4206	Cloud Computing	3	CSD1001
CSD4210	Deep Learning	3	CSD4002
CSD4215	Foundations of Data Science	3	CSD2001
CSD4219	Information Retrieval	3	CSD2001

CSD4224	Natural Language Processing	3	CSD2001
CSD4225	Object Oriented Analysis and Design	3	CSD2001
CSD4230	Social and Information Networks	3	CSD2001
CSD4235	Special Topics in Data Science	3	CSD2001

List of Elective courses in specialization buckets – Systems and Networks

Course code	Title	Credits	Prerequisites
CSD4206	Cloud Computing	3	CSD1001
CSD4207	Compiler Design	3	CSD2001
CSD4212	Distributed Systems	3	CSD2001, CSD2005
CSD4213	Embedded systems	3	CSD2003
CSD4220	Internet of Things	3	CSD2001, CSD4001
CSD4223	Mobile Computing	3	CSD4001
CSD4225	Object Oriented Analysis and Design	3	CSD2001
CSD4226	Parallel and Concurrent Programming	3	CSD2005
CSD4227	Real Time Systems	3	CSD2005
CSD4234	Special Topics in Systems and Networks	3	CSD2001

Appendix-III: List of interdisciplinary specializations

The students enrolled in B. Tech. Computer Science and Engineering (4 year) would have an option to specialize in one the following interdisciplinary areas-

1. Applied Machine Intelligence
2. Cognitive Robotics
3. Electric Mobility
4. Real-time Embedded Systems

Minimum Requirement for Specialization:

The student wishing to do specialization must have overall CGPA ≥ 6 . The student must complete minimum of 12 credits from the list of elective courses from the chosen specialization bucket.

At the time of graduation (end of 8th semester before convocation), students who have completed the specialization requirement may apply for a specialization in CSE to UG advisor for further processing. A student can apply for a specialization in only one of the mentioned areas.

List of Elective courses in specialization buckets - Applied Machine Intelligence

Prerequisites – Fundamental knowledge of the following two topics:

- Probability and Statistics (Axioms of probability, conditional probability and independence, Bayes theorem, Random variables, basics of information theory) and
- Machine Learning

Title	Compulsory (Yes/No)	Offering Department
Deep Learning	Yes	CSE/EED
Image Processing and Applications	No	CSE
Quantum Computing	No	EED
Robotics	No	MECH

List of Elective courses in specialization buckets - Cognitive Robotics

Title	Compulsory (Yes/No)	Offering Department
Computer Vision	No	CSE
Industrial Automation	No	MECH
Internet of Things	No	CSE/EED
Modern Control	No	EED
Soft Robotics	No	MECH
Robotics	Yes	MECH

List of Elective courses in specialization buckets - eMobility

Prerequisites – CSE students are required to complete Power electronics & machine drives as a UWE.

Title	Compulsory (Yes/No)	Offering Department
Computer Vision	No	CSE
Connected Car Technology	Yes	CSE
Digital Control of Power Converter	No	EED
Electric Vehicle Technology	Yes	MED
Energy Storage and Conversion Devices for Electric Vehicles	No	MECH
EV Converters and Power Train	Yes	EED

List of Elective courses in specialization buckets - Real-time Embedded Systems

Prerequisites – CSE students are required to complete the following prerequisite courses:

- Linear Systems and Transforms (To be completed as a Core course)
- Embedded System Hardware (To be completed as UWE)

Title	Compulsory (Yes/No)	Offering Department
Digital Design with FPGA	No	EED

Cloud Computing	No	CSE
Distributed Systems	No	CSE
Graph Signal Processing	No	EED
Real Time Systems	No	CSE
Reconfigurable Computing	No	EED