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 – (Applicable for 2022 batch onwards)

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

Program Structure | Computer Science and Engineering

Semester 1 (20 credits)				
Course Code	**Title**	**Credits**	**Category**	**Prerequisites**
SOE 4801	Introduction to Engineering: Ideas to Impact	2	Experiential	NA
CSD1001	Problem Solving Using Programming	4(3:0:1)	Core	NA
PHY1001	Fields, Waves & Quanta	5	Core	NA
MAT1007	Multivariate Calculus	4	Core	NA
ECE1001	The Electron's Path: Fundamentals of EEE	3	Core	NA
SNS1001	Nature's Code: Chemistry & Biology	2	Core	NA

Semester 2 (18 credits)				
Course Code	**Title**	**Credits**	**Category**	**Prerequisites**
	Connected Intelligence: Sensors and IoT	3	Core	NA
	The Matter of Materials	2	Core	NA
	Design to Reality: CAD and 3D Printing	2	Experiential	NA
	Linear Systems and Transforms	4	Core	NA
	Forces in Action	3	Core	NA
	Environment & Sustainability	4	CCC/UWE	NA

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**
CSD3001	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	CSD2001
	CCC/UWE	3	CCC/UWE	NA
	Elective 1	3	Elective	NA

Semester 5 (23 credits)				
Course Code	**Title**	**Credits**	**Category**	**Prerequisites**
CSD4001	Computer Networks	4(3:0:1)	Core	CSD3001

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**
CSD3002		Formal Languages and Automata Theory	4	Core	CSD2001
CSD4004		Advanced Algorithms	4(3:0:1)	Core	CSD2001
CSD4007		Advanced Computer Architecture	3(3:1:0)	Elective	CSD3001, CSD2003
		Major Elective 4	3	Elective	
Any one:	CSD4902	Software Project	3 (0:0:3)	Project Basket	CSD2001
	CSD4901	Software Engineering	3 (2:0:1)		
		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	Core	
	CCC/UWE	6	CCC/UWE	
CSD4903	Project-1	6	Project Basket	CSD2001

Semester 8 (9 credits)				
Course Code	**Title**	**Credits**	**Category**	**Prerequisites**
CSD4904	Project-2/Internship	3	Project Basket	CSD2001
	CCC/UWE	6	CCC/UWE	

List of Courses

List of courses in Experiential Learning category

Course code	Title	Credits	Semester	Prerequisites
SOE4801	Introduction to Engineering: Ideas to Impact	2	1	NA
	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
PHY1001	Fields, Waves & Quanta	5	1	NA
MAT1007	Multivariate Calculus	4	1	NA
ECE1001	The Electron's Path: Fundamentals of EEE	3	1	NA
SNS1001	Nature's Code: Chemistry & Biology	2	1	NA
	Connected Intelligence: Sensors and IoT	3	2	NA
	The Matter of Materials	2	2	NA
	Linear Systems and Transforms	4	2	NA
	Forces in Action	3	2	NA
CSD2001	Data Structures and Algorithms	4	3	CSD1001
CSD2002	Discrete Mathematics	5	3	CSD1001
CSD2003	Computer Organization	4	3	CSD1001
CSD2004	Object Oriented Programming	4	3	CSD1001
CSD3001	Operating Systems	4	4	CSD2001, CSD2003
CSD2006	Introduction to Probability and Statistics	4	4	NA
CSD2007	Applied Linear Algebra	4	4	NA
CSD4801	Introduction to Database Systems	4	4	CSD2001
CSD4001	Computer Networks	4	5	CSD2001
CSD4002	Artificial Intelligence and Machine Learning	4	5	CSD2001
CSD3002	Formal Languages and Automata Theory	4	6	CSD2001
CSD4004	Advanced Algorithms	4	6	CSD2001
CSD4007	Advanced Computer Architecture	4	6	CSD3001, CSD2003
		79		

List of electives:

Course code	Title	Credits	Prerequisites
CSD4201	Advanced Data Management Systems	3 (3:0:0)	CSD4801
CSD4237	AI Applications	3 (2:0:1)	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)	CS4002
CSD4211	Image Processing and Its Applications	3 (2:0:1)	CSD2002
CSD4212	Distributed Systems	3 (2:0:1)	CSD2001, CSD3001
CSD4213	Embedded systems	3 (3:0:0)	CSD2003
CSD4214	Ethical Hacking	3 (3:0:0)	CSD2001
CSD4215	Foundations of Data Science	3 (2:0:1)	CSD2001
CSD4216	Foundation of Information Security	3 (3:0:0)	CSD2001
CSD4217	Generative AI	3 (3:0:0)	CSD4002
CSD4218	High Performance Computer Architecture	3 (2:0:1)	CSD2003
CSD4219	Information Retrieval	3 (3:0:0)	CSD2001
CSD4220	Internet of Things	3 (2:0:1)	CSD2001, CSD4001
CSD4221	Optimization Techniques	3(3:0:0)	CSD2001
CSD4222	Medical Imaging	3 (2:0:1)	CSD4002
CSD4223	Mobile Computing	3 (2:0:1)	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)	CSD3001
CSD4227	Real Time Systems	3 (3:0:0)	CSD3001
CSD4228	Reinforcement Learning	3 (2:0:1)	CSD2006
CSD4229	Secure coding	3 (3:0:0)	CSD2001
CSD4230	Social and Information Networks	3 (2:0:1)	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

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 (2:0:1)	Knowledge of Probability and Statistics and Machine Learning
CSD4209	Computer Vision	3 (2:0:1)	CSD2001
CSD4210	Deep Learning	3 (2:0:1)	CSD4002
CSD4211	Image Processing and Its Applications	3 (2:0:1)	CSD2002
CSD4219	Information Retrieval	3 (3:0:0)	CSD2001
CSD4217	Generative AI	3 (3:0:0)	CSD4002
CSD4222	Medical Imaging	3 (2:0:1)	CSD4002
CSD4224	Natural Language Processing	3 (2:0:1)	CSD2001
CSD4228	Reinforcement Learning	3 (2:0:1)	CSD2006
CSD4232	Special Topics in Artificial Intelligence	3 (3:0:0)	CSD2001

List of Elective courses in specialization buckets - Cybersecurity

Course code	Title	Credits	Prerequisites
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
CSD4213	Embedded systems	3 (3:0:0)	CSD2003
CSD4214	Ethical Hacking	3 (3:0:0)	CSD2001
CSD4216	Foundation of Information Security	3 (3:0:0)	CSD2001
CSD4220	Internet of Things	3 (2:0:1)	CSD2001, CSD4001
CSD4223	Mobile Computing	3 (2:0:1)	CSD4001
CSD4229	Secure coding	3 (3:0:0)	CSD2001
CSD4233	Special Topics in Cybersecurity	3 (3:0:0)	CSD2001

List of Elective courses in specialization buckets - Data Science

Course code	Title	Credits	Prerequisites
CSD4201	Advanced Data Management Systems	3 (3:0:0)	CSD4801
CSD4203	Algorithms for Big Data	3 (3:0:0)	CSD2001
CSD4206	Cloud Computing	3 (2:1:0)	CSD1001
CSD4208	Computer Graphics	3 (2:0:1)	CSD2001
CSD4210	Deep Learning	3 (2:0:1)	CSD4001
CSD4215	Foundations of Data Science	3(2:0:1)	CSD2001
CSD4219	Information Retrieval	3 (3:0:0)	CSD2001
CSD4224	Natural Language Processing	3 (2:0:1)	CSD2001
CSD4225	Object Oriented Analysis and Design	3 (3:0:0)	CSD2001
CSD4230	Social and Information Networks	3 (2:0:1)	CSD2001
CSD4235	Special Topics in Data Science	3 (3:0:0)	CSD2001

List of Elective courses in specialization buckets – Systems and Networks

Course code	Title	Credits	Prerequisites
CSD4206	Cloud Computing	3 (2:1:0)	CSD1001
CSD4207	Compiler Design	3 (3:0:0)	CSD2001
CSD4212	Distributed Systems	3 (2:0:1)	CSD2001, CSD3001
CSD4213	Embedded systems	3 (3:0:0)	CSD2003
CSD4218	High Performance Computer Architecture	3 (2:0:1)	CSD3001, CSD2003
CSD4220	Internet of Things	3 (2:0:1)	CSD2001, CS4001
CSD4223	Mobile Computing	3 (2:0:1)	CSD4001
CSD4225	Object Oriented Analysis and Design	3 (3:0:0)	CSD2001
CSD4226	Parallel and Concurrent Programming	3 (2:0:1)	CSD3001
CSD4227	Real Time Systems	3 (3:0:0)	CSD3001
CSD4234	Special Topics in Systems and Networks	3 (3:0:0)	CSD2001

Areas of Interdisciplinary Specialization

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 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 - 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

Course code	Title	Compulsory (Yes/No)	Offering Department
	Deep Learning	**Yes**	CSE/EED
	Image Processing and Its Applications	No	CSE
	Medical Image Processing	No	EED
	Quantum Computing	No	EED
	Robotics	No	MECH
More relevant courses may be added by other SoE departments			

List of Elective courses in specialization buckets - Cognitive Robotics

Course code	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 - Electric Mobility

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

Course code	Title	Compulsory (Yes/No)	Offering Department
	Computer Vision	No	CSE
	Connected Car Technology	**Yes**	MECH
	Digital Control of Power Converter	No	EED
	Electric Vehicle Technology	**Yes**	CSE
	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:

- Mathematics 2 (To be completed as a Core course)
- Embedded System Hardware (To be completed as UWE)

Course code	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

Course Description

Core Courses

Data Structures and Algorithms

PART A: COURSE IDENTIFIERS

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

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 gains insights into both theoretical and empirical analysis of algorithms—covering iterative and recursive techniques. Key algorithmic design strategies will be examined, including divide and conquer, greedy algorithms, and dynamic programming.

11. Course Aims

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

1. To teach the students how to select and design data structures and algorithms that are appropriate for problems that they might encounter.
2. Show the correctness of algorithms and study their computational complexities.
3. It covers a mixture of theoretical knowledge and practical experience.

12. Learning Outcomes

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

1. 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.
2. Ability to handle operations like searching, sorting, insertion, deletion, traversing mechanism, etc. on various data structures.
3. Understand the concept of hashing for associating keys with data for faster access.
4. Recognize different algorithmic design strategies which include recursion, divide-and-conquer, the greedy method, and dynamic programming, etc.
5. Understand and implement Basic and advanced concepts in Trees, Priority Queues, Graphs, and Graph Algorithms.

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
Algorithm Design Techniques: Divide and Conquer, Greedy programming, Dynamic programming
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 Tree

Textbooks and References:

1. Introduction to Algorithms: Cormen, Leiserson, Rivest and Stein
2. Data Structures and Algorithm Analysis in C by Mark Allen Weiss.
3. Data Structures Through C in Depth by S.K. Srivastava and Deepali Srivastava
4. Data Structures and Algorithms Made Easy by Narsimha Karumanchi.
5. Data Structures and Algorithms in C by Goodrich and Tamassia
6. Data Structures and Algorithms: Aho, Hopcroft and Ullmann: Addison Wesley.

Discrete Mathematics

PART A: COURSE IDENTIFIERS

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

PART B: OBJECTIVES AND PRACTICE

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.

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

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

Curriculum Content:

1. **Set Theory:** Operations on sets, Cartesian product of sets, General proofs of some fundamental identities on sets.
2. **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
3. **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
4. **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
5. **Combinatorics:** Basics of Counting, Permutation & Combination, Derangement, Pigeonhole Principle, Binomial and Multinomial Theorems, Principle of Inclusion and Exclusion
6. **Number Theory:** Modular Arithmetic, probability, application to cryptography, Algorithm complexity.
7. **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

Textbooks and References:

1. Kenneth H Rosen, "Discrete Mathematics and Its Application", 7th Edition, McGraw Hill Education.
2. C. L. Liu and Mohapatra, "Elements of Discrete Mathematics", 3rd Edition, McGraw Hill Companies.
3. N. Dev, "Graph Theory with Applications to Engineering and Computer Science",
4. Kolman, Busby, Ross, "Discrete Mathematical Structures"

5. Seymour Lipschutz and Marc Lars Lipson, “Discrete Mathematics”, Revised 3rd Edition, McGraw Hill.

Computer Organization

PART A: COURSE IDENTIFIERS

1. School	**School of Engineering**
2. Department	**Computer Science & 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	**None**

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 Aims

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:

1. Understand evolution and history of computers.
2. Understand and analyze performance of a computer using different metrics.
3. Understanding bus interconnection systems.
4. Understand and apply computer arithmetic operations.
5. Understand fundamentals of assembly languages, machine instructions and addressing modes
6. Understand CPU structure and function.
7. Understand the concept of pipelining and associated hazards.
8. Understand the computer memory system and memory hierarchy.
9. Understand and analyze cache memory, internal and external memory.
10. Understand and analyze I/O modules, programmed I/O, interrupt driven I/O and DMA.

13. Curriculum Content

M1 **Introduction**: Definition of computer organization, computer architecture, and their relationship. Basic Components: Input devices, output devices, CPU, memory, storage devices. Number Systems: Binary, octal, hexadecimal, decimal number systems, conversion between them.

M2 **Digital Logic Design:** Basic operations (AND, OR, NOT), laws and theorems (De Morgan's, distributive, associative, commutative); Logic Gates: Implementation of Boolean functions using AND, OR, NOT gates; Combinational Circuits: Half-adder, full-adder, decoder, encoder, multiplexer, demultiplexer; Karnaugh map; Minimization of Boolean Functions; Sequential Circuits: Latches (SR, D, JK), flip-flops

(SR, D, JK, T); Registers: Shift registers (serial-in, serial-out; serial-in, parallel-out; parallel-in, serial-out; parallel-in, parallel-out), counters.

M3 **Instruction Set Architecture (ISA);** Basic Concepts: Instruction format, opcode, operands, addressing modes; Instruction Types: Data transfer, arithmetic, logical, control flow; Addressing Modes: Immediate, direct, indirect, register, indexed, based, relative; RISC vs. CISC: Characteristics, advantages, and disadvantages.

M4 **The Central Processing Unit (CPU):** Control Unit: Fetch-decode-execute cycle, microprogramming; Arithmetic Logic Unit (ALU): Basic operations, implementation using combinational circuits; Registers: General-purpose registers, special-purpose registers (PC, IR, SP, etc.); Pipeline: Basic concepts, stages, hazards.

M5 **Memory:** RAM (SRAM, DRAM), ROM (PROM, EPROM, EEPROM); Memory Hierarchy; Programmable Logic Arrays (PLA). Cache Memory: Mapping techniques (direct, associative, set-associative), replacement algorithms (LRU, FIFO, random); Virtual Memory: Paging, segmentation, translation lookaside buffer (TLB);

M6 **I/O Devices:** Keyboard, mouse, monitor, printer, disk drives; I/O Interfaces: Parallel and serial interfaces (UART, USB); I/O Techniques: Programmed I/O, interrupt-driven I/O, DMA; Direct Memory Access (DMA): DMA controller, DMA transfer cycle; Interfacing; Basic Concepts: Interfacing devices to the CPU; Hardware Interfacing: Using GPIO pins, addressing devices; Software Interfacing: Device drivers, operating system interaction; Case Studies: Interfacing simple devices (e.g., LEDs, switches, sensors).

14. Bibliography

- ✔ M. Morris Mano, “Digital Logic and Computer Design”, PHI 2005
- ✔ Computer Organisation and Architecture, Smruti R. Sarangi, McGrawHill India.
- ✔ Computer Architecture: A Quantitative Approach, John L. Hennessy, David A. Patterson, 4th Edition, Morgan Kaufmann Publishers.
- ✔ Nicholas P Carter Adapted by Raj Kamal, “Computer Architecture and Organization”, 2nd edition, Schaum’s Outline, Tata McGraw Hill,2010.
- ✔ Computer Organization and Architecture: Designing for Performance, William Stallings, 10th Edition, Pearson India.

Object Oriented Programming

PART A: COURSE IDENTIFIERS

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

PART B: OBJECTIVES AND PRACTICE

0. **Course Summary**

This course includes the introducto and advanced concepts and implementation of the Object Oriented Paradigm using the Java programming language. Topics would include Introduction to Objects and Classes, Inheritance and Polymorphism, String handling, GUI Basics and Components, Graphics, Exception Handling, Abstract Classes and Interfaces, Event-Driven Programming, Multi-Threading, JDBC.

0. Course Aims

This course will enable the students to understand different programming paradigms and learn the concepts of object oriented programming using java, and apply its structures in detail.

0. Learning Outcomes

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

1. Ability to understand, write and execute java programs using various input and output mechanisms.
2. Understand object oriented programming concepts.
3. Understand and use abstract classes and interfaces.
4. Understand use Inheritance and Polymorphism.
5. Understand and apply string-handling mechanisms
6. Understand and create Graphical user interface.
7. Understand and perform event driven programming.
8. Understand and apply exception handling.
9. Understand and apply database programming.
10. Understand and apply the components of Java Collections Framework.

0. Curriculum Content

- **Object oriented thinking**: Need for oop paradigm, A way of viewing world – methods, classes and instances, class hierarchies (Inheritance), etc.
- **OOP Basics**: Concepts of classes, encapsulation, 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.
- **Multithreaded Programming:** Basics of threads, Java threaded model, defining threads using Runnable interface and Thread superclass, Multiple threads, Thread Synchronization using synchronized methods and using synchronized blocks

- **Database Programming using JDBC**: Introduction to Java Database Connectivity (JDBC), Connecting Java to Databases, Executing SQL Queries from Java.
- **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

Laboratory:

- Hands-on practice by implementing OOP concepts taught in the class.
- Students will use NetBeans IDE with latest version of JDK to write the programs.

0. Bibliography

Textbook:

- Introduction to Java Programming, by Y. Daniel Liang, Comprehensive version, 10th edition.

Reference Books:

- Java: The Complete Reference, by Herbert Schildt, 11th edition.
- Java, How to Program, 11th edition, Paul Deitel, Harvey Deitel, PHI Learning Pvt. Ltd. ISBN: 10: 0133813436 ISBN-13: 9780133813432.

Operating System

PART A: COURSE IDENTIFIERS

1. School	**School of Engineering**
0. Department	**Computer Science and Engineering**
0. Course Code	**CSD3001**
0. Course Title	**Operating System**
0. Credits (L:T:P)	**4 (3:0:1)**
0. Contact Hours (L:T:P)	**3:0:2**
0. Prerequisites	**CSD2001, CSD2003**
0. Major Core for	**CSE**
0. 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 includes 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 Aims

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

1. To develop the student's basic understanding of Linux commands and its implementation.
2. To provide the students with in-depth knowledge about Operating System abstraction and mechanisms.
3. To introduce the students with the concept of process synchronization, deadlock resolution, and efficient CPU scheduling using various algorithms.
4. Practically create multiple threads using pthreads, concurrency, and multi-threaded programming to exploits the multi-core hardware architecture.
5. To introduce the students with memory management, filesystems, and I/O systems.
6. 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:

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

13. Curriculum Content

Unit 1: Overview of Operating System: Introduction, System Structure
Unit 2: Process Management: Process State and lifecycle, Process Scheduling Algorithms, Thread Creation
Unit 3: Process Synchronization: Synchronization tools, Synchronization Example, pthread: Multi-threaded Programming, Deadlocks
Unit 4: Memory Management: Memory Management Strategies, Page Replacement Algorithms, Virtual Memory Management

Unit 5: Storage Management & File System: File System, Implementing File Systems, Secondary Storage Structure, Disk scheduling algorithms, I/O Systems
Unit 6: Protection & Security: System Protection, System Security, UNIX/LINUX and Windows as an example.
Unit 7: Special Purpose System: Real Time Systems, Distributed Systems

Textbook & Reference Books:

1. "Operating System Concepts" by Abraham Silberschatz, Peter B. Galvin, and Greg Gagne: https://os.ecci.ucr.ac.cr/slides/Abraham-Silberschatz-Operating-System-Concepts-10th-2018.pdf
2. "Modern Operating Systems" by Andrew S. Tanenbaum
3. Operating Systems: Three Easy Pieces by Remzi H. Arpaci-Dusseau (Author), Andrea C. Arpaci-dusseau (Author): https://pages.cs.wisc.edu/~remzi/OSTEP/

Introduction to Probability and Statistics

PART A: COURSE IDENTIFIERS

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

PART B: OBJECTIVES AND PRACTICE

0. 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 of 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 enquires statistical tools for analysis of data and inference.

0. Course Aims

The course aims to cover fundamental concepts in Probability and Statistics.

0. Learning Outcomes

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

. have a better understanding of the principles to develop probability models and their analysis, learn to generate random numbers and carry out simulations.

. carry out exploratory data analysis and learn methods of statistical inference- testing of hypothesis, estimation of parameters, linear and graphical models, and regression analysis.

0. 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.

Text Books

- Pishro-Nik, Hossein. *Introduction to probability, statistics, and random processes*. Blue Bell, PA, USA: Kappa Research, LLC, 2014.

- Harchol-Balter, Mor. *Introduction to probability for computing*. Cambridge University Press, 2023
- Introduction to Probability and Statistics for Engineers and Scientists, Fifth edition, Sheldon Ross, Elsevier 2014
- Bertsekas, Dimitri, and John N. Tsitsiklis. *Introduction to probability*. Vol. 1. Athena Scientific, 2008.
- Ross, Sheldon M. *Introduction to probability models*. 12th edition Academic press, 2019.

Applied Linear Algebra

PART A: COURSE IDENTIFIERS

1. School	**School of Engineering**
0. Department	**Computer Science and Engineering**
0. Course Code	**CSD2007**
0. Course Title	**Applied Linear Algebra**
0. Credits (L:T:P)	**4 (3:1:0)**
0. Contact Hours (L:T:P)	**3:1:0**
0. Prerequisites	**NA**
0. Major Core for	**None**
0. Major Elective for	**4^{th} / 5^{th} Semester**

PART B: OBJECTIVES AND PRACTICE

0. 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.

0. Course Aims

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.

0. Learning Outcomes

i. 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.
ii. Further, this course provides a foundation for further studies in fields such as data science, machine learning, computer science, engineering, economics, and statistics.

0. Curriculum Content

Vectors – Definition and operations, Linear functions, Norm and distance, Clustering, Linear independence

Matrices – Definition and operations, Types of Matrices, Linear equations, Matrix multiplication, Matrix inverses, Matrix decomposition

Least squares, Least squares data fitting, Least squares classification, Multi-objective least squares, Constrained least squares, Nonlinear least squares, Constrained nonlinear least squares

0. Textbooks and References

Introduction to Applied Linear Algebra, by *Stephen Boyd* and *Lieven Vandenberghe*

Introduction to Databases

PART A: COURSE IDENTIFIERS

1. School	**School of Engineering**
0. Department	**Computer Science and Engineering**
0. Course Code	**CSD4801**
0. Course Title	**Introduction to Databases**
0. Credits (L:T:P)	**4 (3:0:1)**
0. Contact Hours (L:T:P)	**3:0:2**
0. Prerequisites	**CSD2001**
0. Major Core for**	**CSE**
0. Major Elective for	**None**

PART B: OBJECTIVES AND PRACTICES

0. **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.
0. **Course Aims:**

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.

0. **Learning Outcomes:** On successful completion of the course, students will be able to achieve the following:

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

0. **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.

0. Bibliography

Textbook and References:

1. Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems,* Pearson.
2. Avi Silberschatz, Henry F. Korth, S. Sudarshan, *Database System Concepts,* McGgraw-Hill.
3. Raghu Ramakrishnan, *Database Management Systems*, McGraw-Hill.
4. J. D. Ullman, *Principles of Database Systems,* Galgotia.

Computer Networks

PART A: COURSE IDENTIFIERS

1. School	**School of Engineering**	
0. Department	**Computer Science and Engineering**	
0. Course Code	**CSD4001**	
0. Course Title	**Computer Networks**	
0. Credits (L:T:P)	**4 (3:0:1)**	
0. Contact Hours (L:T:P)	**3:0:2**	
0. Prerequisites	**CSD3001**	
0. Major Core for	**CSE**	
0. Major Elective for	**NA**	

PART B: OBJECTIVES AND PRACTICE

1. 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.

0. Course aims

The aims of this course are:

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

0. Learning outcomes

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

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

0. Curriculum content

ISO/OSI stack, medium access control, LAN technologies (Ethernet, Token ring), Understanding the WIFI medium access control mechanism, Concepts of hubs, switches, gateways and routers, Flow and error control techniques, IP(v4/v6), Routing algorithms, Internetworking, Congestion control, TCP/UDP, Concept of sockets, Application layer protocols (DNS, SMTP, POP, FTP, HTTP), Basics of network security.

0. Bibliography

- Kurose and Ross “Computer Networking: A Top-Down Approach”, 4th Edition, 2007.
- 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.

Artificial Intelligence and Machine Learning

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)	**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

An introduction to the basic principles, techniques, and applications of Artificial Intelligence. Coverage includes x problem solving, search algorithms, game theory, perception, machine learning, planning, and agent design. The course also covers the topic expert systems, neural networks, fuzzy logic, robotics, natural language processing, and computer vision in introductory level data.

11. Course Aims

a) Understand different types of data.
b) Learn how to construct models that can predict from data (supervised learning) and organize data into coherent categories (unsurpervised learning).
c) Understand where and how machine learning can go wrong.

12. Learning Outcomes

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

a. Given a search problem, analyze the problem, select the search method, and developed the algorithm for it
b. Implement A* and iterative deepening search. Derive heuristic functions for A* search for a given problem.
c. For constraint satisfaction problems, implement backtracking search with conflict directed back jumping, arc consistency, and the Minimum Remaining Values and Least Constraining Value heuristics.
d. Implement local search with the min conflicts heuristic and Implement and execute by hand alpha-beta search. Design good evaluation functions and strategies for game playing.Build models for prediction and data organization from data.
e. Understand the basic theories and concepts that underly machine learning.
f. Understand machine learning models and learn to use basic Machine Learning libraries.

13. Curriculum Content

Introduction to Artificial Intelligence and Machine Learning, Solving Problems by Searching, heuristic search techniques, constraint satisfaction problems, stochastic search methods,
Game Playing : mini max, alpha-beta pruning
The learning problem, Types of learning.
Training, validation, testing, generalization, overfitting.
Features and feature engineering, dimensionality reduction.
Bayesian decision theory.
Parametric methods.
Tree models.
Linear models.
SVMs and kernel based models.
Nearest neighbour models.

Markov models.
Neural network models.
Ensemble methods - boosting, bagging, voting schemes.
Distance metrics and cluster based models.

Textbooks:

- Artifiicial Intelligence: A Modern Approach, Russell and Norvig, Pearson Education (Low Priced Edition), 2004.
- E Charnaik and D McDarmott. Introduction to Artificial Intelligence. Addison Wesley Publishing Co., New York.
- Rich. Artificial Intelligence. McGraw-Hill International.
- N J Nilson. Principles of Artificial Intelligence. Galgotia Publications (P) Ltd., New Delhi.

Formal Languages and Automata theory

PART A: COURSE IDENTIFIERS

1.	**School**	**School of Engineering**	
2.	**Department**	**Computer Science and Engineering**	
3.	**Course Code**	**CSD3002**	
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

Develop a formal notation for strings, languages and machines.
Design finite automata to accept a set of strings of a language.
Prove that a given language is regular and apply the closure properties of languages.
Learn about DFA and NFA and conversion from NFA to DFA
Proof using Pumping Lemma
Design context free grammars to generate strings from a context free language and convert them into normal forms.
Prove equivalence of languages accepted by Push Down Automata and languages generated by context free grammars.
Convert from PDA to CFG and Vice Versa,
Proof using Pumping lemma for CFG
Identify the hierarchy of formal languages, grammars and machines
Distinguish between computability and non-computability and Decidability and undecidability.

11. Course Aims

Formal Languages and Automata Theory deals with the concepts of automata, formal languages, grammar, algorithms, computability, decidability, and complexity.
Formal Languages and Grammars have an important role to play in understanding on compiler design, and Complexity on cryptography. The purpose of this course is to acquaint the student with an overview of the theoretical foundations of computer science from the perspective of formal languages.

12. Learning Outcomes

- *Provide* 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.

Bibliography:

- 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.

Advanced Algorithms

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 Objectives

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, elements of computational geometry, NP completeness.

11. **Course Aims:**

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

0. 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.
- 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.
- Describe approximation algorithm and Compute the approximation factor of an approximation algorithm (PTAS and FPTAS).

13. Course 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, etc. Advanced data structures: Fibonacci heap, union find splay trees. Amortized complexity analysis. Randomized algorithms:

Randomized algorithms to be introduced a bit early, i.e. before NP completeness to highlight randomization as an algorithmic technique. Application areas(i)Geometric algorithms: convex hulls, nearest neighbor, Voronoi diagram, etc.(ii)Algebraic and number-theoretic algorithms: FFT, primality testing, etc.(iii)Graph algorithms: network flows, matching, etc.(iv)Optimization techniques: linear programming

Reducibility between problems and NP completeness: discussion of different NP-complete problems like satisfiability, clique, vertex cover, independent set, Hamiltonian cycle, TSP, knapsack, set cover, bin packing, etc. Backtracking, branch and bound
Approximation algorithms: Constant ratio approximation algorithms.
Miscellaneous: Introduction to external memory algorithms, parallel algorithms.

14. Bibliography:

- Rajeev Motwani and Prabhakar Raghavan, Randomized Algorithms, Cambridge University Press.
- Allan Borodin, Ran El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press.
- Nancy Lynch, Distributed Algorithms, Morgan Kaufmann.
- Robert EndreTarjan, Data Structures and Network Algorithms, SIAM.
- L. Grotchel, L. Lovasz, and A. Schrijver, Geometric algorithms and Combinatorial Optimization, Springer.
- M. Kearns and U. Vazirani, An Introduction to Computational Learning Theory. MIT Press.
- N. Alon and J. H. Spencer, The Probabilistic Method, John Wiley.
- Vijay Vazirani, Approximation Algorithms, Springer.
- Fan Chung, Spectral Graph Theory, American Mathematical Society

Software Engineering

PART A: COURSE IDENTIFIERS

1. School	**School of Engineering**
0. Department	**Computer Science and Engineering**
0. Course Code	**CSD4901**
0. Course Title	**Software Engineering**
0. Credits (L:T:P)	**3 (2:0:1)**
0. Contact Hours (L:T:P)	**2:0:2**
0. Prerequisites	**CSD2001**
0. Major Core for	**CSE**
0. Major Elective for	**None**

PART B: OBJECTIVES AND PRACTICE

0. 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.

0. Course Aims

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

1. Learning Outcomes

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

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

0. Curriculum Content

Module-1 (10 hours)

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.

Module-2 (11 hours)

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

Module-3 (11 hours)
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.

Module-4 (10 hours)
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

0. Bibliography

1. Fundamentals of Software Engineering, 5th Edition, Rajib Mall, PHI.
2. Software Engineering, 9th Edition, Ian Sommerville, Pearson Publications
3. Software Engineering: A Precise Approach, Pankaj Jalote.
4. Berzins and Luqi, Software Engineering with Abstraction, 1st Edition, Addison-Wesley, 1991.
5. Martin L. Shooman, Software Engineering – Design, Reliability and Management, McGraw-Hill Education, 1984
6. Software Engineering, 3rd Edition, K K Aggarwal & Yogesh Singh, New Age International Publishers.

Advanced Computer Architecture

PART A: COURSE IDENTIFIERS

2. School	**School of Engineering**
1. Department	**Computer Science and Engineering**
1. Course Code	**CSD4007**
1. Course Title	**Advanced Computer Architecture**
1. Credits (L:T:P)	**4 (3:1:0)**
1. Contact Hours (L:T:P)	**3:0:2**
1. Prerequisites	**CSD2003, CSD3001**
1. Major Core for	**CSE**
1. Major Elective for	**None**

PART B: OBJECTIVES AND PRACTICE

1. **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. The modern applications operate on advanced handheld electronic devices, powered by high-performance microprocessors. To meet the growing demands of modern applications, CPU designers continually address challenges through cost-effective architectural innovations. This course offers an in-depth exploration of the design and development of high-end microprocessors, equipping students with the knowledge to support future technological advancements and applications.

2. **Course Aims**

   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.

3. **Learning Outcomes**

   i. Draw a functional block diagram of the organisation of computer-based systems and discuss how applications influence the range of design choices.

   ii. For a given memory configuration details compute response time, throughput, miss-rate, miss penalty

   iii. For a given multi-core processor architecture, develop multi-threaded programs and determine the speed-ups.

   iv. Analyse the performance of a given cache memory organisation and its optimization techniques.

   v. Basic understanding of the Network-on-chip architecture and its future research perspectives.

4. **Curriculum Content**

   a. 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.

   b. 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.

   c. **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

d. **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.
e. **Memory system:** DRAM memory, Secondary Storage Systems, Hard Disk and Flash memory.
f. **Peripherals:** Devices Bus structures and standards; Synchronous and asynchronous buses; Types and uses of storage devices; Interfacing I/O to the rest of the system; Reliability and availability; I/O system design; Platform architecture
g. **Network-on-chips(NOCs):** Tiled Chip Multicore Processors(TCMP), Routing Techniques in Network on Chip(NoC), NoC Router Microarchitecture, TCMP and NoC: Design and Analysis, Future Trends in Computer Architecture Research.

5. Bibliography

- John Paul Shen and Mikko H. Lipasti, Modern Processor Design: Fundamentals of Superscalar Processors, Tata McGraw-Hill.
- M. J. Flynn, Computer Architecture: Pipelined and Parallel Processor Design, Narosa Publishing House.
- Kai Hwang, Advanced Computer Architecture: Parallelism, Scalability, Programmability, McGraw-Hill
- Computer Architecture - A Quantitative Approach,5th edition, John L. Hennessy, David A. Patterson.
- Computer Systems Design and Architecture, 2nd Edition, Vincent P. Heuring
- Computer Organization and Architecture, 6th Edition, William Stallings
- Advanced Computer Architectures-A Design Space Approach, Dezsosima, Terence Fountain, Peter Kacsuk.

Course Description

Elective Courses

Advanced Database Management System

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 Aims

To enable students:

i. To learn the query optimization techniques.
ii. Explore indexing types and apply them in case studies To design Search Engine
iii. Understand distributed databases and NoSQL systems through hands-on labs.
iv. Learn ETL processes and OLAP operations in data warehousing.
v. 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:

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

13. Curriculum Content:

1. **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.

2. **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.

3. **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.

4. **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.

5. **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.

6. **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.

7. **Special purpose databases**: Temporal, spatial and multimedia databases.

14. Bibliography

Textbook and References:

1. Avi Silberschatz, Henry F. Korth, S. Sudarshan, *Database System Concepts,* McGgraw-Hill.
2. M. Tamer Özsu and Patrick Valduriez, *Principles of Distributed Database Systems,* Springer.
3. Pramod J. Sadalage and Martin Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence,* Addison-Wesley.
4. Jonathan Seidman, Ted Malaska, *Foundations for Architecting Data Solutions: Managing Successful Data Projects,* Paperback, O'Reily.
5. Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems,* Pearson.

AI Applications

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 (2:0:1)**	
6. Contact Hours (L:T:P)	**2:0:2**	
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

AI basics, Deductive and Inductive approaches overview, ML basics, Applications of AI and ML in different fields

11. Course Aims

- 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

- Expertise in Deductive and Inductive approaches to solve real-life problems
- Applying AI and ML concepts for public good

13. Curriculum Content

- AI Search
- Decision Modelling approaches
- AI Methods, focusing on applications
- ML Methods: Supervised and unsupervised
- Perceptrons
- Deep learning
- Semi-supervised, self-supervised and Weakly Supervised approaches
- Continual Learning
- Active LEarning
- Applications in Various fields

14. Bibliography

1. Charu C Aggarwal, Artificial Intelligence: A Textbook, Springer, 2021.
2. Stuart Russell, Peter Norvig, Artificial Intelligence: A Modern Approach, 4th Ed., Pearson Education, 2021.
3. Dive into Deep Learning, Alexander Smola, Cambridge University Press, 2023
4. Understanding Deep Learning, Simon J. D. Prince, MIT Press, 2023
5. Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press, 2016
6. Deep Learning, Cris Bishop, Springer, 2024

Algorithms for Big Data

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	**CSD1001, CSD2002**
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 Outcomes

On successful completion of the course, students will be able to design and analyze sketching and streaming algorithms for big data.

12. 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 Techique: Biclique and Bipartiteness Testing
Random Walks and Testing Bipartiteness & Expansion
Regularity Lemma and Testing Triangle Freeness
Boolean Functions, BLR test for Linearity

13. Text Books:

- Probability and Computing: Randomized Algorithms and Probabilistic Analysis, by Mitzenmacher and Upfal.
- Algorithmic and Analysis Techniques in Property Testing, by Dana Ron.
- Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches, by Graham Cormode, Minos Garofalakis, Peter J. Haas and Chris Jermaine.
-

Applied Cryptography

PART A: COURSE IDENTIFIERS

14. School	**School of Engineering**
15. Department	**Computer Science and Engineering**
16. Course Code	**CSD4204**
17. Course Title	**Applied Cryptography**
18. Credits (L:T:P)	**3 (3:0:0)**
19. Contact Hours (L:T:P)	**3:0:0**
20. Prerequisites	**CSD1001, CSD2002**
21. Major Core for	**None**
22. Major Elective for	**CSE**

PART B: OBJECTIVES AND PRACTICE

23. Course Summary

This course will introduce students to basic building blocks of cryptography and applications of cryptographic protocols in real world. The focus will be on how cryptography and its application can maintain privacy and security in electronic communications and computer networks.

24. Course Aims

- To teach basic cryptographic primitives
- To teach how to apply and use cryptographic concepts to real world problems
- To emphasis on learning the security solutions with assignments/programming assignments related to known attacks and important crypto primitives.

25. Learning Outcomes

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

- Learn fundamentals of Cryptography and standard algorithms used to provide confidentiality, integrity, authenticity and nonrepudiation.
- Learn the use case scenarios of modern cryptographic methods (symmetric encryption, public key encryption, hash functions, key management, digital signatures, certificates).
- Learn basics of applied cryptography to understand existing security solutions.
- Learn the approaches to use cryptographic concepts to real world problems.

26. 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 management, 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.

Text Books:

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

Blockchain Technology

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 Contacts, 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 Aims

- 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

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

13. Curriculum Content

Unit-I: Introduction to cryptography, blockchain & cryptocurrencies, Bitcoin, Merkle tree, P2P networks, transactions, blocks, Proof of work, wallets.
Unit-II: Foundation of consensus, network models, corruption tolerance, different decentralized consensus mechanisms such as proof of work, proof of stake.
Unit-III: Ethereum basics, Decentralized Apps, EVM, and the Ethereum blockchain, Introduction of programming in solidity**.**
Unit-IV: Introduction to private blockchain: Hyperledger fabric architecture, use case scenarios.
Unit-V: Security and privacy in public blockchain, anonymity, privacy enhancing cryptography, ring signature, overview of Monero crypto currency.
Unit-VI: 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.

Readings and Text Books:

1. Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, Steven Goldfeder, Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction, Princeton University Press, 2016.
2. Bitcoin: A Peer-to-Peer Electronic Cash System, by Satoshi Nakamoto
3. Ethereum yellow paper, https://ethereum.github.io/yellowpaper/paper.pdf
4. Solidity documentation.
5. Tutorials — Hyperledger Fabric Docs main documentation, https://hyperledger-fabric.readthedocs.io/en/latest/tutorials.html
6. Zero to Monero: First Edition a technical guide to a private digital currency; for beginners, amateurs, and experts Published June 26, 2018 (v1.0.0) Kurt M. Alonso, https://www.getmonero.org/library/Zero-to-Monero-1-0-0.pdf
7. Draft version of "S. Shukla, M. Dhawan, S. Sharma, S. Venkatesan, 'Blockchain Technology: Cryptocurrency and Applications', Oxford University Press, 2019.
8. Josh Thompson, 'Blockchain: The Blockchain for Beginnings, Guild to Blockchain Technology and Blockchain Programming', Create Space Independent Publishing Platform, 2017.

Cloud Computing

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 Aims

The aim of the course is that the students are able 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

Module 1 (Contact hours: 10)

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.

Module 2 (Contact hours: 10)

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.

Module 3 (Contact hours: 8)

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. Bibliography

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

Compiler Design

PART A: COURSE IDENTIFIERS

1. School	**SOE**	
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**	

10. Course Summary:

The aim the course is 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. Learning Outcomes:

- Given high level language program statement determine the sequence of tokens of it
- Outline the functions of Lexical Analyzer and Syntax Analyzer and developed the parsing technique to parse the given input string.
- Optimize the complexity and the size of the given source code using machine dependent and independent code improving transformations.
- Generate the functionally equivalent target code by utilizing registers in an effective way.

12. 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) gram-mars 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: Translation of different language features, different types of intermediate forms.

Code Improvement (optimization): programming languages.

13. Bibliography:

- Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers: Principles, Techniques and Tools, Addison-Wesley.
- Michael L. Scott, Programming Language Pragmatics, Elsevier.

Computer Graphics

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**
8. Major Core for	**None**
9. Major Elective for	**CSE**

PART B: OBJECTIVES AND PRACTICE

10. Course Summary

A comprehensive introduction to computer graphics and rendering, covering fundamental principles, techniques, and technologies. It covers the fundamental principles of computer graphics and rendering algorithms using both ray tracing, path tracing, and rasterization.

11. Course Aims

The course aims to cover fundamental concepts in Computer Graphics.

12. Learning Outcomes

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

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

13. Curriculum Content

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

14. Text Books and References

a. Matt Pharr and Greg Humphreys, Physically Based Rendering, 3rd Edition, Morgan Kaufmann, 2016 (available online)
b. Peter Shirley, Fundamentals of Computer Graphics, 4th Edition, AK Peters, 2015 (available online)
c. Andrew S. Glassner, An Introduction to Ray Tracing, 1st Edition, Morgan Kaufmann, 1989 (available online)

d. Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL, 6th Edition, by Edward Angel and Dave Shreiner (available online).

Computer Vision

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

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.

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:

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

Curriculum Content:

1.Low-level vision
–image processing, edge detection, feature detection, image matching, RANSAC, etc.

2.Geometry and Photometry
–projective geometry, stereo, structure from motion, etc.

3.Recognition and Learning
–Recognition, Machine Learning, ANN, CNN, etc.
–CNN and Computer Vision
–Generative Models in Computer Vision

Textbooks and References:

1. Rick Szeliski, *Computer Vision: Algorithms and Applications*
   online at: http://szeliski.org/Book/
2. Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016
   online at: http://www.deeplearningbook.org/
3. Dive Into Deep Learning https://d2l.ai/

Deep Learning

PART A: COURSE IDENTIFIERS

14. School	**SOE**	
15. Department	**Computer Science and Engineering**	
16. Course Code	**CSD4210**	
17. Course Title	**Deep Learning**	
18. Credits (L:T:P)	**3 (2:0:1)**	
19. Contact Hours (L:T:P)	**2:0:2**	
20. Prerequisites	**CSD4001**	
21. Major Core for	**NA**	
22. Major Elective for	**CSE**	

PART B: OBJECTIVES AND PRACTICE

23. 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.

24. Course Aims

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.

25. Learning Outcomes

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

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

26. Curriculum Content

a. Introduction to Deep Learning:
   i. Understanding neural networks and their biological inspiration.
   ii. Differences between deep learning, machine learning, and traditional programming.
   iii. Backpropagation and Loss functions.
   iv. Universal Approximation Theorem

b. Linear Models
   i. Linear Regression: loss function, feature encoding, parameters
   ii. Linear Classification: loss function, softmax function.

c. Multi-Layer Perceptron:
   i. Basic architecture of neural ne.tworks.
   ii. Activation functions, loss functions, and gradient descent optimization.
   iii. Why Deep is better than Shallow?
   iv. Vanishing and Exploding gradients issues and solutions.
   v. Requirement of Regularization, Dropouts.

d. Convolutional Neural Networks (CNNs):
   i. Understanding convolutional layers and their application in image processing.
   ii. Striding and Deconvolutions (Transpose convolution)
   iii. Pooling layers and fully connected layers for classification tasks**.**

iv. Importance of Residual connections.

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

27. Bibliography

a. Dive into Deep Learning, Alexander Smola, Cambridge University Press, 2023
b. Understanding Deep Learning, Simon J. D. Prince, MIT Press, 2023
c. Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press, 2016
d. Deep Learning, Cris Bishop, Springer, 2024

Image Processing and Applications

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 Aims

Course aims to cover mathematical models and algorithms in Image processing. Course also aims to make student proficient in matrix processing using NumPy/Tensorflow/Pytorch.

12. Learning Outcomes

On successfully completion of course, students will be able to:
i) have better understanding of the images and its background acquisition.
ii) carry out data analysis on images to extract intended information.
iii) perform noise correction on ill-acquired image data.

13. Curriculum Content

Introduction: 2 lectures
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: 2 lectures
1. Basic intensity transformation
2. Histogram processing

Spatial Filtering: 2-3 lectures
1. Mechanics of spatial filtering and application
2. Smoothing and Sharpening

Frequency domain Filtering: 2-3 lectures
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: 3 lectures
1. Noise Model
2. Spatial Noise Filtering
3. Frequency domain Noise filtering

Morphological Image Processing: 3 lectures

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

Image Segmentation: 4 lectures

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. Bibliography

1. Gonzalez and Woods, “Digital Image Processing”, 4th Edition, Pearson.
2. Kevin P. Murphy, “Probabilistic Machine Learning: An Introduction”, MIT Press

Distributed Systems

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 (2:0:1)**
6. Contact Hours (L:T:P)	**2:0:2**
7. Prerequisites	**CSD2001, CSD3001**
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 Aims

- 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:

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

11. Curriculum Content

- **Unit-I:** Introduction to Distributed Systems, Message Passing, Remote Procedure Call (RPC), System Model, The Two Generals' Problem, Byzantine Generals' Problem

- **Unit-II:** Leader Election, Distributed Models, Physical and Logical Time, Clock Synchronization Techniques, Causality, Happens-Before Relationship, Two-Phase Commit Protocol, Linearizability and Eventual Consistency Models
- **Unit-III:** Broadcast Ordering and Algorithms, Global State & Snapshot and Distributed Mutual Exclusion-Non-Token and Quorum based approaches, Distributed Mutual Exclusion-Token based approaches
- **Unit-IV:** Data Replication and State Machine Replication, Token-Based Distributed Mutual Exclusion, Consensus Protocols: Raft & Agreement Techniques, Checkpointing & Rollback Recovery
- **Unit-V:** Deadlock Detection, Distributed Shared Memory (DSM), Distributed Minimum Spanning Tree (MST) Algorithms, Termination Detection, Message Ordering & Group Communication, Fault Tolerance and Self-Stabilization
- **Unit-VI:** Distributed Randomized Algorithms, Distributed Hash Tables (DHT) and Peer-to-Peer (P2P) Computing
- **Unit-VII:** Case Study: Google File System (GFS), Hadoop Distributed File System (HDFS), MapReduce Programming Model and Apache Spark, Sensor Networks, Authentication & Security in Distributed Systems

Textbooks and References:

- Sukumar Ghosh, Distributed Systems: An Algorithmic Approach, Second Edition
- A.D. Kshemkalyani, M. Singhal, Distributed Computing: Principles, Algorithms, and Systems, ISBN: 9780521189842, Cambridge University Press, March 2011.
- Gerard Tel. Introduction to Distributed Algorithms
- Hagit Attiya, Jennifer Welch. Distributed Computing: Fundamentals, Simulations, and Advanced Topics
- Vijay K. Garg. Elements of Distributed Computing
- Distributed Systems, Principles and Paradigms, Andrew S. Tanenbaum, Maarten Van Steen, 2nd Edition, PHI.

Embedded Systems

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.

12. Learning Outcomes:

- Write the goals, technologies, constraints and applications of embedded computing and describe the typical embedded hardware and software products and standards
- Design, implement and test a simple embedded systems for a given application using RASPBERRY PI/ARDUINO
- Assess embedded systems products and systems in terms of their performance, reliability, quality cost.
- Design I/O interface (A/D and D/A Converters, USB and DMA) with embedded systems using MATLAB Software.

11. 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

13. Bibliography:

- Frank Vahid, Tony Givargis, “Embedded System Design, A Unified Hardware/Software Introduction”, John Wiley and Sons Inc, 2002

- Wayne Wolf, " Computers as Components", Morgan Kaufmann, Harcourt India, 2001
- James, K. Peckol, Embedded Systems – A Contemporary Design Tool, Wiley India, 2008
- Richard Zurawski Ed., "Embedded Systems Handbook", CRC Press, Taylor and Francis, 2006

Ethical Hacking

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 Aims

- 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

Unit-I: 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.

Unit-II: 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.

Unit-III: 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.

Unit-IV: Hacking Wireless Networks, Bluetooth hacking, Mobile Phone Hacking, Tools for ethical hacking.

Unit-V: Case studies: various attacks scenarios and their remedies.

Readings and Text Books:

1. William Stallings and Lawrie Brown, “Computer Security Principles and Practice”, 3/e, Pearson Education.
2. James Kurose, Keith Ross, Computer Networking: A Top-Down Approach, Pearson Education.
3. Kevin Beaver, Hacking For Dummies, Wiley Publishing, Inc.

Foundation of Data Science

PART A: COURSE IDENTIFIERS

1. School	**School of Engineering**
2. Department	**Computer Science and Engineering**
3. Course Code	**CSD4215**
4. Course Title	**Foundation of Data Science**
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 course will introduce students to fundamentals of information security, cryptography, access control mechanisms, functional security solutions, system attacks and defences against them.

11. Course Aims

- Enable students to analyze the vulnerabilities in any computing system and hence be able to design a security solution
- Enable students to identify the basic security issues in the computer network communications
- Enable students to evaluate various security mechanisms used in real world

12. Learning Outcomes

- Learn information security basics
- Learn to use and apply various security mechanisms to real world problems

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

Text Books:

- 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
- Introduction to Data Mining (umn.edu)
- 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.

Foundation of Information Security

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 will introduce students to fundamentals of information security, cryptography, access control mechanisms, functional security solutions, system attacks and defences against them.

11. Course Aims

- Enable students to analyze the vulnerabilities in any computing system and hence be able to design a security solution
- Enable students to identify the basic security issues in the computer network communications
- Enable students to evaluate various security mechanisms used in real world

12. Learning Outcomes

- Learn information security basics
- Learn to use and apply various security mechanisms to real world problems

13. Curriculum Content

Unit-I: Security Overview, CIA model, Threats, Policy and Mechanisms, Security Policies, Risk Analysis.
Unit-II: Cryptography Basics, Classical Cryptosystems, Stream Ciphers, Block Ciphers, Public Key Cryptography, Digital Signature, Cryptographic Hash Functions, Digital Certificate.
Unit-III: Authentication Basics, Password-based Authentication, Challenge Response Protocol, Access Control mechanisms, Confidential Policy model.
Unit-IV: Malicious Logic, Trojan Horses, Viruses, Worms, Logic Bombs, Defences, Sandboxing, Intrusion Detection: Principles and Basics, Anomaly modelling, Host and network-based Information Gathering, Organization of Intrusion Detection Systems, Intrusion Response.
Unit-V: DoS and DDoS attack. Firewalls and Proxies, DMZ server,
Unit-VI: Introduction to Ethical Hacking, Typers of Hackers, Ethical Hacking Phases, Types of Penetration Test.
Unit-VII: Overview of Security Protocols SSL/TLS and IPSec

Text Books:

1. Matt Bishop, S.S. Venkatramanayya, "Introduction to Computer Security, 3/e", Pearson Education
2. 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

Generative AI

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 (3:0:0)**
6. Contact Hours (L:T:P)	**3:0:0**
7. Prerequisites	**CSD4001**
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 Aims

- 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

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

13. Curriculum Content

1. **Introduction to Generative AI:** Overview of generative models and their historical context. Generative vs. Discriminative Models.
   Practical: Assignment on Exploring Gen. AI Use Cases

2. **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

3. **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

4. **Audio and Video Generation and IVR System:** Introduction to Audio and Video Generation, Diffusion Models, Building an IVR System, Integration and Practical Applications.

5. **Data Augmentation:** Data Augmentation Techniques, Applications in Machine Learning.

6. **Evaluation of Generative Models:** Metrics for quality evaluation, including discussions on bias and ethics.

14. Bibliography

Text and Reference Books:

- *"Generative AI with Python and TensorFlow2*: *Create images*, *text, and music with VAEs*, *GANs*, *LSTMs*, *Transformer models*" by Joseph Babcock
- "Understanding Deep Learning" by *Simon J.D. Prince, MIT Press.*
- "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
- Research papers and online resources provided throughout the course.

High-Performance Computer Architecture

PART A: COURSE IDENTIFIERS

1. School	**School of Engineering**	
2. Department	**CSE**	
3. Course Code	**CSD4218**	
4. Course Title	**High-Performance Computer Architecture**	
5. Credits (L:T:P)	**3 (2:0:1)**	
6. Contact Hours (L:T:P)	**2:0:2**	
7. Prerequisites	**CSD2003**	
8. Major Core for	**NA**	
9. Major Elective for	**CSE**	

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 Aims

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

a. Draw a functional block diagram of the organisation of computer-based systems and discuss how applications influence the range of design choices.

b. For a given memory configuration details compute response time, throughput, miss-rate, miss penalty

c. For a given multi-core processor architecture, develop multi-threaded programs and determine the speed-ups.

d. Analyse the performance of a given cache memory organisation and its optimization techniques.

e. Given a RAID configuration, determine the data reliability and data throughput

13. Curriculum Content

a. 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.

b. 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.

c. **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

d. **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.
e. **Memory system:** DRAM memory, Secondary Storage Systems, Hard Disk and Flash memory.

14. Bibliography

- John Paul Shen and Mikko H. Lipasti, Modern Processor Design: Fundamentals of Superscalar Processors, Tata McGraw-Hill.
- M. J. Flynn, Computer Architecture: Pipelined and Parallel Processor Design, Narosa Publishing House.
- Kai Hwang, Advanced Computer Architecture: Parallelism, Scalability, Programmability, McGraw-Hill
- Computer Architecture - A Quantitative Approach,5th edition, John L. Hennessy, David A. Patterson.
- Computer Systems Design and Architecture, 2nd Edition, Vincent P. Heuring
- Computer Organization and Architecture, 6th Edition, William Stallings
- Advanced Computer Architectures-A Design Space Approach, Dezsosima, Terence Fountain, Peter Kacsuk.

Information Retrieval

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 (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

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 Aims

To enable students:

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

12. Learning Outcomes

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

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

13. Curriculum Content

1. **Introduction**: Goals and history of IR. The impact of the web on IR.
2. **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.
3. **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.

4. **Experimental Evaluation of IR**: Performance metrics: recall, precision, and F-measure, Evaluations on benchmark text collections, MAP, Precision Recall Curve, nDCG, and BPref.
5. **Query Operations and Languages**: Relevance feedback, Query expansion, Query languages.
6. **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.
7. **Web Search**: Search engines, spidering, indexing, link analysis, web page ranking.
8. **Introduction to Semantic Search:** Semantic Web and Ontologies, Knowledge Graphs in IR, Basics of Entity Recognition and Disambiguation.
9. **Responsible IR:** handling bias and fake-news, privacy, etc.

14. Bibliography

Textbook and Reference Books:

1. Introduction to Information Retrieval, C. Manning, P. Raghavan, and H. Schütze , Cambridge University Press, 2008.
2. Modern Information Retrieval: The Concepts and Technology behind Search, Ricardo Baeza -Yates and Berthier Ribeiro – Neto, ACM Press Books.
3. Search Engines: Information Retrieval in Practice, Bruce Croft, Donald Metzler and Trevor Strohman, Pearson

Internet of Things

PART A: COURSE IDENTIFIERS

15. School	**School of Engineering**	
16. Department	**Computer Science and Engineering**	
17. Course Code	**CSD4220**	
18. Course Title	**Internet of Things**	
19. Credits (L:T:P)	**3 (2:0:1)**	
20. Contact Hours (L:T:P)	**2:0:2**	
21. Prerequisites	**CSD2001, CSD4001**	
22. Major Core for	**NA**	
23. Major Elective for	**CSE**	

PART B: OBJECTIVES AND PRACTICE

24. 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.

25. Course Aims

The aims of this course are:

1. To understand the IoT Protocol stack and its difference from TCP/IP protocol suite.
2. 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.
3. To explore the hardware platforms (e.g., Arduino), simulation using tinker cad simulator.
4. To provide a comprehensive overview on the security issues in IoT.
5. To introduce Machine Learning techniques to analyse IoT systems.
6. To explore some use cases of IoT.

26. Learning Outcomes

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

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

27. 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 (Device-to-device communication):
Definition, application scenario, types of D2D communication, Interference in D2D communication, Pricing algorithm, Interference management in LTE-A enabled D2D communication.
Module-5 (Physical layer security):
Definition, notion of secrecy throughput, different types of security threats in IoT (e.g. Eavesdropping, Spoofing, Jamming).

Bibliography

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

Medical Imaging

PART A: COURSE IDENTIFIERS

1. School	SOE	
2. Department	Computer Science and Engineering	
3. Course Code	CSD4222	
4. Course Title	Medical Imaging	
5. Credits (L:T:P)	3 (2:0:1)	
6. Contact Hours (L:T:P)	2:0:2	
7. Prerequisites	CSD4001	
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. The course focuses on the mathematical theory, and the associated algorithms, within advanced topics in data science, image analysis and processing, and machine learning, for a wide range of applications in medical image computing (also known as medical vision; related to computer vision).

11. Course Aims

Course aims to cover mathematical models and algorithms related to medical images. The course assignments involve several computing experiments to explore the behaviour of the theories and algorithms on real-world medical image data, such as MRI scans, CT scans, Ultrasound data.

12. Learning Outcomes

On successfully completion of course, students will be able to:
i) have better understanding of the medical images and its background acquisition.
ii) carry out data analysis on medical images to extract intended information.
iii) perform noise correction on ill-acquired medical image data.
iv) able to carry out experiments on medical image datasets.

13. Curriculum Content

1) Introduction to imaging modalities.
- Mathematical imaging models for physical signals, sampling, noise and artefact models. Signal modelling and model fitting.
- X ray, computed tomography (CT), positron emission tomography (PET), magnetic resonance imaging (MRI) (including diffusion MRI, functional MRI), microscopy, ultrasound, 2D and 3D imaging.

2) Visualization.
- Methods: sectioning, multimodal images, overlays, rendering surfaces and volumes, using glyphs.

3) Image reconstruction.
- Methods: mathematical models of image regularity, random fields, practical data sampling and acquisition schemes, problem formulations, optimization algorithms.
- Application domains: MRI, CT, PET, others.

4) Image restoration.

- Methods: degradation models for corrupted and missing data, Bayesian graphical modeling and inference, regression methods, learning based methods.
- Application domains: MRI, CT, PET, microscopy, ultrasound, others.

5) Image segmentation, object delineation, classification.
- Methods: clustering, graph partitioning, classification, hidden Markov random fields, multivariate Gaussian, kernel methods, computer aided diagnosis.
- Applications across biological structures and imaging modalities.

6) Statistical shape analysis.
- Methods: descriptors, shape spaces, learning shape models, learning shape mean and modes of variation.
- Application domains: Analysis of change in shape of organs and substructures

7) Image retrieval.
- Methods: image descriptors, image similarity, searching, databases, hierarchical methods.

14. Text Book

1. Mathematics of Medical Imaging. Charles L. Epstein. Prentice Hall 2003.
2. Gonzalez and Woods, "Digital Image Processing", 3rd Edition, Pearson.

18. Reference Book

1. Kevin P. Murphy, "Probabilistic Machine Learning: An Introduction", MIT Press
2. Machine Learning and Medical Imaging. G. Wu, Dinggang Shen, M. Sabuncu. Elsevier 2016.
3. Markov Random Field Modeling in Image Analysis. Stan Z Li. Springer 2009.
4. Introduction to Mathematical Statistics. R. V. Hogg, J. W. McKean, Allen Craig. Pearson 2012.
5. Introduction To Linear Algebra. Gilbert Strang. Wellesley Cambridge Press 2009.
6. Applied Numerical Linear Algebra. James W. Demmel. SIAM 1997.

Mobile Computing

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 (2:0:1)**	
6. Contact Hours (L:T:P)	**2:0:2**	
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 Aims

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.

Module-5: Wireless security protocols

Security attacks in wireless networks, Security protocols in 5G and Wi-Fi, Physical layer security solutions.

14. Bibliography

1. Wireless Communications: Principles and Practice by Theodore S. Rappaport, Pearson.
2. Computer Networking: A Top-Down approach by James F. Kurose and Keith W. Ross, Pearson.
3. J. Liu, N. Kato, J. Ma and N. Kadowaki, "Device-to-Device Communication in LTE-Advanced Networks: A Survey," in IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 1923-1940, Fourth quarter 2015.
4. M. N. Tehrani, M. Uysal and H. Yanikomeroglu, "Device-to-device communication in 5G cellular networks: challenges, solutions, and future directions," in IEEE Communications Magazine, vol. 52, no. 5, pp. 86-92, May 2014.
5. OMNET++: https://omnetpp.org/

Natural Language Processing

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 Aims

- 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. Bibliography

Textbook:

- "Speech and Language Processing" by Daniel Jurafsky and James H. Martin.

Reference Books:

- "Natural Language Processing in Action", Second Edition by Hobson Lane and Maria Dyshel
- "Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit" by Authors: Steven Bird, Ewan Klein, and Edward Loper

Object-Oriented Analysis and Design

PART A: COURSE IDENTIFIERS

Fill both columns if the course is a joint offering and/or is cross-listed with different codes.

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 Aims

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

12. Learning Outcomes

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

1. Grady Booch, Object-Oriented Analysis and Design with Applications (3rd Edition).
2. Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software.
3. Robert Martin, UML for Java Programmers, Prentice Hall,2003

Real Time Systems

PART A: COURSE IDENTIFIERS

15. School	**School of Engineering**	
16. Department	**CSE**	
17. Course Code	**CSD4227**	
18. Course Title	**Real Time Systems**	
19. Credits (L:T:P)	**3 (3:0:0)**	
20. Contact Hours (L:T:P)	**3:0:0**	
21. Prerequisites	**CSD3001**	
22. Major Core for	**None**	
23. Major Elective for	**CSE**	

PART B: OBJECTIVES AND PRACTICE

24. 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.

25. Course Aims

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

26. Learning Outcomes

i. Compare different scheduling algorithms and the schedulability criteria.
ii. 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.
iii. Integrate resource access mechanisms with the scheduling techniques and develop integrated schedulability criteria.
iv. Design real time embedded systems using the concepts of RTOS

27. 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, Schedulabililty 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-Queuing 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.

28. Bibliography

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

Parallel and Concurrent Programming

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	**CSD3001**
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 Aims

- 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:

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

11. 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

Textbooks and References:

- The Art of Multiprocessor Programming. Maurice Herlihy, Nir Shavit
- Multithreaded, Parallel, and Distributed Programming. Gregory R. Andrews
- Principles of Parallel Programming, by Calvin Lin, Larry Snyder, Addison-Wesley.
- Introduction to Parallel Computing (2nd Edition). Ananth Grama , George Karypis, Vipin Kumar, Anshul Gupta
- An Introduction to Parallel Programming. Peter Pacheco
- Wen-Mei W Hwu, David B Kirk, Programming Massively Parallel Processors A Hands-on Approach, Morgann Kaufmann, 3e.'

Reinforcement Learning

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	**CSD224**	
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 Aims

The aims of this course are:

1. To **understand** the aspects of a Reinforcement learning problem: Agent, Environment, Finite Markov Decision Process, Dynamic Programming.
2. To **apply** Tabular solution methods to different engineering problems: Multi-arm Bandits, Monte Carlo Methods, Temporal difference learning, eligibility traces.
3. To **apply** approximate solution methods to different engineering problems: On-policy approximation of action values, off-policy approximation of action values.
4. To **understand** cutting-edge deep RL algorithms such as Rainbow, PPO and SAC.
5. **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:

1. **Understand** the aspects of Reinforcement learning (RL) problems.
2. **Apply** tabular solution methods to different engineering problems.
3. **Apply** approximate solution methods to different engineering problems.
4. **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. Bibliography

1. Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning: An Introduction", The MIT press, 2015.
2. Phil Winder, Reinforcement Learning, O'Reilly, 2020 [ISBN: 9781098114831].

Secure Coding

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 Aims

- 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

Readings and Text Book:

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

Social and Information Networks

PART A: COURSE IDENTIFIERS

1. School	**School of Engineering**
2. Department	**Computer Science and Engineering**
3. Course Code	**CSD4330**
4. Course Title	**Social and Information Networks**
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:

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 Aims:

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 biological networks Congestion networks – Game theory approach Select advanced topics.

14. Bibliography

https://www.cs.cornell.edu/home/kleinber/networks-book/networks-book.pdf