Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Technology

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

Structure & Syllabus

B.Tech. (Computer Engineering)

With Effect from Academic Year 2025-26

Prepared by: - Board of Studies in Computer Engineering

Approved by: - Academic Board, Vishwakarma Institute of Technology, Pune

Chairman – BOS **Chairman – Academic Board**

Vision of the Institution

"To be globally acclaimed Institute in Technical Education and Research for holistic Socio-economic development".

Mission of the Institution

- To ensure that 100% students are employable and employed in Industry, Higher Studies, become Entrepreneurs, Civil / Defense Services / Govt. Jobs and other areas like Sports and Theatre.
- To strengthen Academic Practices in terms of Curriculum, Pedagogy, Assessment and Faculty Competence.
- Promote Research Culture among Students and Faculty through Projects and Consultancy.
- To make students Socially Responsible Citizen.

Vision of the Department

"To be a leader in the world of computing education practising creativity and innovation".

Mission of the Department

- To ensure students' employability by developing aptitude, computing, soft, and entrepreneurial skills
- To enhance academic excellence through effective curriculum blended learning and comprehensive assessment with active participation of industry
- To cultivate research culture resulting in knowledge-base, quality publications, innovative products and patents
- To develop ethical consciousness among students for social and professional maturity to become responsible citizens

Knowledge and Attitude Profile (WK)

WK	WK Statements
WK1	A systematic, theory-based understanding of the natural sciences applicable to the discipline and awareness of relevant social sciences.
WK2	Conceptually-based mathematics, numerical analysis, data analysis, statistics and formal aspects of computer and information science to support detailed analysis and modelling applicable to the discipline.
WK3	A systematic, theory-based formulation of engineering fundamentals required in the engineering discipline.
WK4	Engineering specialist knowledge that provides theoretical frameworks and bodies of knowledge for the accepted practice areas in the engineering discipline; much is at the forefront of the discipline.
WK5	Knowledge, including efficient resource use, environmental impacts, whole-life cost, reuse of resources, net zero carbon, and similar concepts, that supports engineering design and operations in a practice area.
WK6	Knowledge of engineering practice (technology) in the practice areas in the engineering discipline.
WK7	Knowledge of the role of engineering in society and identified issues in engineering practice in the discipline, such as the professional responsibility of an engineer to public safety and sustainable development.
WK8	Engagement with selected knowledge in the current research literature of the discipline, awareness of the power of critical thinking and creative approaches to evaluate emerging issues.
WK9	Ethics, inclusive behavior and conduct. Knowledge of professional ethics, responsibilities, and norms of engineering practice. Awareness of the need for diversity by reason of ethnicity, gender, age, physical ability etc. with mutual understanding and respect, and of inclusive attitudes.

Programme Outcomes [PO]

PO	Program Outcome Statements
PO1	**Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified inWK1 to WK4 respectively to develop to the solution of complex engineering problems.
PO2	**Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)
PO3	**Design/Development of Solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)
PO4	**Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
PO5	**Engineering Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6 **The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7 **Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8 **Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9 **Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10 **Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11 **Life-Long Learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

Programme Specific Outcomes [PSO]

Programme Specific Outcome Statements

PSO1 Demonstrate proficiency in programming with a sound understanding of fundamental computing principles

PSO2 Conceive well-structured design and proficient implementation to address real world challenges using software paradigms, algorithms and technologies

PSO3 Acquire and showcase expertise in emerging fields within computer science, engineering, and technology

Program Educational Objectives (PEOs)

- Demonstrate application of sound engineering foundations to be a committed technology workforce
- Apply mathematical and computing theory knowledge base to provide realistic computer engineering solutions
- Exhibit problem-solving skills and engineering practices to address problems faced by the industry with innovative methods, tools, and techniques
- Develop professional and ethical practices adopting effective guidelines to acquire desired soft skills in the societal and global context
- Aim for continuing education and entrepreneurship in emerging areas of computing

Course Name Nomenclature as per NEP (For FY and SY)

BSC: Basic Science Course	MDOE: Multi Disciplinary Open Elective
ESC: Engineering Science Course	CC: Co-curricular Course
PCC: Program Core Course	HSSM: Humanities Social Science and Management
PEC: Program Elective Course	IKS: Indian Knowledge System
ELC: Experiential Learning Course	FP: Field Project
MD: Multi Disciplinary	INT: Internship

Nomenclature for Teaching and Examination Assessment Scheme AY 2024-25

Sr No.	Category	Head of Teaching/ Assessment	Abbreviation used
1	Teaching	Theory	Th
2	Teaching	Laboratory	Lab
3	Teaching	Tutorial	Tut
4	Teaching	Open Elective	OE
5	Teaching	Multi Disciplinary	MD
6	Teaching	Computer Science	CS
7	Assessment	Laboratory Continuous Assessment	CA
8	Assessment	Mid Semester Assessment	MSA
9	Assessment	End Semester Assessment	ESE
10	Assessment	Home Assignment	HA
11	Assessment	Course Project	CP
12	Assessment	Group Discussion	GD
13	Assessment	PowerPoint Presentation	PPT
14	Assessment	Class Test –1	CT1
15	Assessment	Class Test –2	CT2
16	Assessment	Mid Semester Examination	MSE
17	Assessment	End Semester Examination	ESE
18	Assessment	Written Examination	WRT
19	Assessment	Multiple Choice Questions	MCQ
20	Assessment	Laboratory	LAB

Title: Course Structure **FF No. 653**

Branch: Computer **Year:** S.Y. **A.Y.:** 2025-26 **Module:** III

Sub No.	Sub Code	Subject Name	Teaching Scheme (Hrs/Week			Test -1	MSA	Test -2	ESA						Total	Credits
			Th	Lab	Tut	CT1 (%)	MSE (%)	CT2 (%)	LAB CA (%)	CP (%)	PPT /GD/ HA (%)	Practical +CVV (%)	CVV (%)	ESE (%)		
S1	CS2305	PCC: Data Structures-I	2	2	0				10	30		40+20			100	3
S2	CS2302	PCC: Logic Design and Microprocessor	2	2	0				10	30			20	40(W)	100	3
S3	CS2303	PCC: Object Oriented Programming	2	2	0				10	30		40+20			100	3
S4	CS2304	PCC: Database Management System	2	2	0				10	30			20	40(W)	100	3
S5	CSM001	MDM: Discrete Structures and Graph Theory**	2	0	1	35 (O)		35 (O)			30				100	3
S6	HS2002	HSS: From Campus To Corporate – 1	2	-	-		50 (O)							50 (O)	100	2
S7	HS2001	AEC: Reasoning And Aptitude Development – 3	-	-	1									100	100	1
S8	CS2306	VSEC: Design Thinking - 1	-	-	1									100	100	1

S9	CS2307	FP: Engineering Design And Innovation - I		8			30							70	100	2
		Total	**12**	**16**	**3**	**35**	**80**	**35**	**40**	**120**	**30**	**120**	**40**	**400**	**900**	**21**

***: MDM, HSSM, RAD online**

****: MDM Course offered by Computer Department**

Title: Course Structure **FF No. 653**

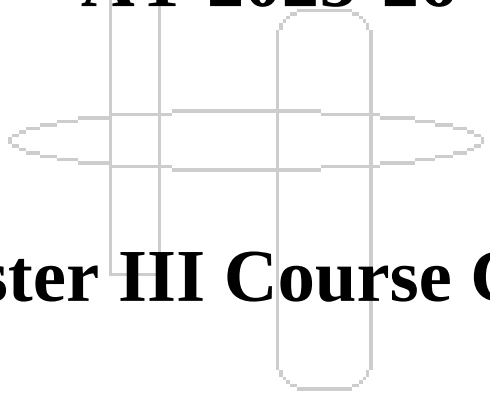
Branch: Computer **Year:** S.Y. **A.Y.:** 2025-26 **Module:** IV

Sub No.	Sub Code	Subject Name	Teaching Scheme (Hrs/Week)													Credits
			Th	Lab	Tut	Test -1	MSA	Test -2	ESA						Total	
						CT1 (%)	MSE (%)	Test (%)	LAB CA (%)	CP (%)	PPT /GD/ HA (%)	Practical + CVV (%)	CVV (%)	ESE (%)		
S1	CS2308	PCC: Data Structures-II	2	2	**0**				10	30			20	40(W)	100	3
S2	CS2309	PCC: Theory of Computation	2	0	**0**	35(W)		35(W)					30		100	2
S3	CS2310	PCC: Operating System	2	2	**0**				10	30		40+20			100	3
S4	CS2311	PCC: Software Engineering	2	2	**0**				10	30			20	40(W)	100	3
S5	CS2312	PCC: System and Application Development	0	2	0				40					60	100	1
S6	MM0702	MDM: Microcontroller	2	-	1	35 (O)		35 (O)			30				100	3
S6	MM0703	MDM: Internet of Things	2	-	1	35 (O)		35 (O)			30				100	3
S7	HS2003	HSS: From Campus To Corporate – 2	2				50(O)							50(O)	100	2

S8	HS2004	AEC: Reasoning And Aptitude Development - 4	1											100	100	1
S9	CS2313	VSEC: Design Thinking - 2			1									100	100	1
S10	CS2314	FP: Engineering Design And Innovation - 2		8			30							70	100	2
		Total	**15**	**16**	**3**	**70**	**80**	**70**	**70**	**90**	**30**	**60**	**70**	**460**	**1000**	**21**

INDEX

S. Y. B. Tech. Computer Engineering
AY 2025-26



Semester III Course Content

FF No. : 654

CS2305: Data Structures-I

Credits: 3 **Teaching Scheme: Theory: 2 Hours / Week**
Lab: 2 Hours / Week

Course Prerequisites: Basic programming Skills (C/C++)

Course Objectives:

1. To introduce the basic concepts of data structures and algorithms.
2. To learn and understand linear and non-linear data structure constructs.
3. To implement searching and sorting techniques using linear data structures.
4. To understand how to solve problems using step by step approach with the help of fundamental data structures.
5. To associate data structures in developing and implementing efficient algorithms.

Course Relevance:

This is a basic Course for Computer Engineering and allied branches. This course has a high relevance in all domains of computer engineering such as in Industries; research etc. as a basic prerequisite course. Data Structures are a crucial part of computer algorithms as they allow programmers to do data management efficiently. A wise selection of data structures can improve the performance of a computer program or algorithm in a more useful way.

Syllabus
Theory

Section 1: Linear Data Structures

Unit 1: Arrays **(4 Hours)**

Asymptotic Notations, Time and Space Complexity Introduction, Memory Representation and application of Single and Multidimensional arrays, Sparse Matrix. **Searching and sorting techniques:** Linear Search, Binary search with Analysis. **Sorting Techniques:** Bubble Sort, Insertion Sort, Merge Sort, Quick Sort with Analysis and passes.

Unit 2: Linked Lists **(4 Hours)**

Dynamic memory allocation, Singly Linked Lists, Doubly linked Lists, Circular linked lists and Generalized linked lists, Applications of Linked list, introduction to Vectors and Application.

Unit 3: Stacks and Queues **(6 Hours)**

Stack: Stack representation and Implementation using arrays and Linked lists. Applications of stack in Recursion, Expression conversions and evaluations. **Queues:** Representation and implementation

using array and Linked lists, Types of queue. Applications of Queues: Job Scheduling, Josephus problem etc.

Section 2: Non-Linear Data Structures

Unit 4: Trees **(7 Hours)**

Basic terminology, representation using array and linked lists. Tree Traversals: Recursive and Non recursive, Operations on binary tree. Binary Search trees (BST).

Unit 5: Graphs **(7 Hours)**

Terminology and representation using Adjacency Matrix and Adjacency Lists, Graph Traversals and Application: BFS and DFS, Connected graph, Bipartite Graph, Detecting Cycle in graph. Minimum Spanning tree: Prims and Kruskal's Algorithm, Shortest Path Algorithms, Union Find.

Unit 6: Hashing **(2 Hours)**

Hashing techniques, Hash table, Hash functions. Collision handling and Collision resolution techniques.

Syllabus
Laboratory

List of Experiments

1) To implement the different sorting algorithms.

2) To implement the linked list.

3) To implement any application of Stack data structure.

4) Implement various expression conversions using Stack.

5) To implement any application of Queue data structure.

6) To implement an algorithm to perform Binary Search Tree (BST) operations (Create, Insert, Delete and Traversals).

7) To implement an algorithm to perform various operations on Binary Tree (Mirror image, Height, Leaf node display, Level wise display etc.)

8) To implement an algorithm to perform various Tree traversals using Stack.

9) To implement Graph traversal: algorithms: Depth First Search and Breadth First Search.

10) To implement Prim's and Kruskals Algorithms to find a Minimum Spanning Tree (MST).

11) To implement Dijkstra's algorithm to solve a Single Source Shortest Path Problem.

12) To implement Hashing algorithms.

Course Project

List of Course Projects

1) Finding Nearest Neighbors.

2) Calendar Application using File handling.

3) Path finder in Maze.

4) Word Completion Using Trie.

5) Bloom Filters.

6) Different Management Systems.

7) Scheduling Applications and Simulation.

8) Shortest Path Applications. (Kirchhoff's Circuit, TSP with Scenarios).

9) Efficient Storage and Data Retrieval Systems.

10) Different Gaming Application.

Course Outcomes

The student will be able to –

1) Make use of single and multi-dimensional array for searching and sorting based applications.

2) Construct computer science applications with the help of dynamic storage representation.

3) Build computer science applications using stacks and queues.

4) Demonstrate the use of tree data structure to represent and manipulate hierarchically organized data in various applications.

5) Utilize graph data structure to design social media, network based and circuit applications.

6) Design and develop the single and multithreads applications by applying hash table and hash map techniques.

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PSO1**	**PSO2**	**PSO3**
CO1	2	2	3								2	3		2
CO2	2	3	2								2	3		2
CO3	3	3	3			2	2				2	3	2	2
CO4	3	3	3	3		2	2				2	3	2	2
CO5	3	3	2								2	3		2
CO6	2	3	3								2	3		2
Average	**2.5**	**3.0**	**2.66**	**3.0**		**2.0**	**2.0**				**2.0**	**3.0**	**2.0**	**2.0**

CO Attainment levels:

Weights for attainment levels: L1 - Easy-0.75 L2 - Comfortatble-0.7 L3 – Medium – 0.65
L4 – Somewhat difficult – 0.6 L5 – Difficult – 0.55
CO1 – L3, CO2– L3, CO3 – L2, CO4 – L4, CO5 – L4 and CO6 – L5

Future Courses Mapping:
Advanced Data Structures, Design and Analysis of Algorithms, Compiler Design, Systems Programming, Data Science and similar courses.

Job Mapping:
Data Structures is must necessary part of any core programming job. Without Data structures it is not possible to be good in Competitive coding. All Industries always look for a strong knowledge in Advanced Data structures. Without learning this course, one can't imagine a job in computer/IT related industries and research.

Text Books:
1. E. Horwitz , S. Sahani, Anderson-Freed, " Fundamentals of Data Structures in C", Second Edition, Universities Press.
2. Y. Langsam, M.J. Augenstein, A.M.Tenenbaum, "Data structures using C and C++", Pearson Education, Second Edition.
3. Narasimha Karumanchi, "Data Structures and Algorithm Made Easy", Fifth Edition, CareerMonk publication.

Reference Books:
1.J. Tremblay, P. Soresan, "An Introduction to data Structures with applications", TMHPublication, 2nd Edition.

For MOOCs and other learning Resources

1. www.nptelvideos.in,
2. www.geeksforgeeks.org
3. https://www.youtube.com/watch?v=244YpoG1pqA&list=PLrikLQMZHuSonRoDheibeb9ffd9phWIyu&index=5
4. https://classroom.volp.in/

FF No. : 654

CS2302: Logic Design and Microprocessor

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Lab: 2 Hours/Week

Course Prerequisites:

1. Basic Electronics
2. Computer Architecture and Organization

Course Objectives:

1. To understand the concept of logic gates, codes, simplification of logical equations using Boolean algebra and Karnaugh map.
2. To design combinational logic circuits.
3. To design sequential logic circuits.
4. To study the fundamental concepts of Computer System and Microprocessor
5. To gain knowledge of Microprocessor Operating Modes.
6. To study data organization of Pentium and analyse the processor's actual functioning.

Course Relevance:

The course addresses the concepts, principles and techniques of designing digital systems. The course teaches the fundamentals of digital systems applying the logic design and development techniques. This course forms the basis for the study of advanced subjects like Microcontroller through Interfacing, VLSI Designing, Operating System and Parallel Computing.

Syllabus
Theory

Section 1: Topics/Contents

Unit 1: Fundamentals of Digital Logic (4 Hours)

Introduction: Analog vs digital circuits, Computer Codes (BCD, Excess-3, Gray code,ASCII Code), Logic Gates: Basic gates (AND, OR NOT, EX-OR, EX-NOR),Universal gates (NOR, NAND), Logic functions: Boolean laws, Conversion of logic functions into a truth table, and vice versa. SOP and POS forms of representation. Minterms and Maxterms. Karnaugh's Map: Simplification of logic functions by theorems and K-Map. Don't care about the conditions.

Unit 2: Combinational Logic Circuits (5 Hours)

Half adder, Full adder, Half Subtractor, Full Subtractor, Multiplexers:74153, De-multiplexers, Encoder, Decoders:74138,Code converter: BCD to Excess-3 and vice versa ,Gray to Binary and vice versa, 2 bit Comparator.

Unit 3: Sequential Logic Circuit (5 Hours)

Introduction of flip-flop (F.F), 1 bit memory cell, clocked S-R, J-K-7476, T, D Flip-flop: Truth table, Excitation table, Characteristics table, Shift Register, Up-down Asynchronous counter, Up-down Synchronous counter.

Section 2: Topics/Contents

Unit 4: Introduction to 8086 Microprocessor (5 Hours)

History of Microprocessors, Evolution, Block diagram of microprocessor based systems .Introduction to 8086, Architecture of 8086: BIU & EU, Register organization, Segmentation, Memory Banking, Addressing modes of 8086.

Unit 5: Operating Modes of Pentium (5 Hours)

Introduction to 80386, Block diagram of 80386, Segmentation and Paging in 80386 ,Introduction to Pentium Processor, Real Mode, Protected Mode,Virtual 86 mode, Software Model of Pentium, Functional Description of Pentium, Register Set of Pentium .Addressing Modes of Pentium.

Unit 6: Data Organization and Hardware details of Pentium (4 Hours)

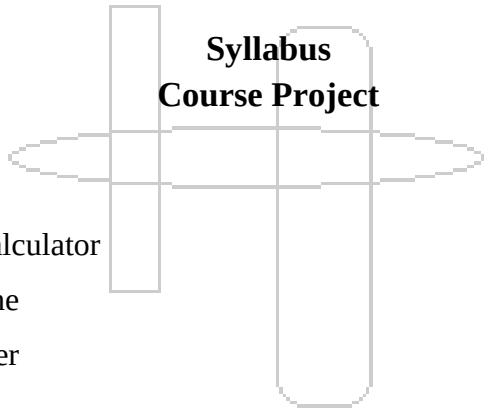
Pentium Data Organization ,Interrupts .Pentium Pin Description ,Bus Operations, The Pentium's Superscalar Architecture, Pipelining, Branch Prediction, The Instruction & Data Caches, The Floating-Point Unit.

Syllabus
Laboratory

List of Experiments

1. Verification of Logical Gates and Simplification of logic equation using Boolean Algebra
2. K-map- e.g.Code converters Excess-3 to BCD and vice versa using logical gates
3. Multiplexer-e.g. Full adder 8:1 Mux using 4:1 Mux IC-74153
4. Decoder -e.g. 2 bit Comparator using 3:8 decoder IC-74138

5. Asynchronous Up /down counter using JK flip-flop IC-7476
6. Synchronous Up /down counter using JK flip-flop IC-7476
7. Shift register- e.g. 4-bit Shift Register using D FF IC-7474
8. (Write an ALP to print"Hello World" and also implement Macro using Netwide assembler)
9. Write an ALP to print 2 digit number in hexadecimal format on the screen
10. Write an ALP to find positive number and negative numbers from the array of signed number stored in memory and display magnitude of negative and positive numbers
11. Write an ALP to perform non-Overlapping block transfer operation without using string operations, Data bytes in a block stored in one array transfer to another array.
12. Write an ALP to convert 4-digit HEX number into equivalent 5-digit BCD number
13. Write an ALP to convert 5-digit BCD number into equivalent 4-digit HEX number
14. Write an ALP for Multiplication of two 8-bit Binary numbers using Successive and Add method



List of Course Projects

1. Multiplexer-Based Calculator
2. Digital Voting Machine
3. Traffic Light Controller
4. Digital Stopwatch
5. Automatic Door Control System
6. Electronic Safe Lock System
7. Elevator Control System
8. Ebike Speed Controller System
9. Wireless Biomedical Parameter Monitoring System Using ARM9
10. Solar Sea Weather and Pollution Transmitter Buoy
11. Detection System of Collision using ARM Cortex M3
12. ARM and RFID Based Security System (Home, Office, Industrial)
13. A combination of Microcontroller and Machine Learning, Deep Learning, AI, Blockchain etc Based

14. Microcontroller based any Agriculture, Health Care, Education, Govt., Transportation, Banking, Insurance Based but not limited

Course Outcomes

1. Describe logic gates, codes, simplification of logic equations using Boolean algebra and K-map.(1)
2. Design and optimize combinational circuits using logic gates .(2)
3. Design and analyze sequential circuits using logic gates and flip flops .(3)
4. Describe the various microprocessor operations (4)
5. Discuss the operating modes of Pentium processor (5)
6. Relate concept of data organization and working model of Pentium processor (5)

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PSO1**	**PSO2**	**PSO3**
CO1	3	1	2				2		2			3		
CO2	2	3	2	1		2		3	2			3		
CO3	2		3	2	2	2	3		2		2		3	
CO4	2		2		3	3				2			3	
CO5	2	1	2		3	3				2			3	
CO6	3													
Average	2.33	0.83	1.83	0.5	1.5	1.66	1 .16	0 .5	1	1	1.66	1	1.83	0 .5

Future Courses Mapping:

Microcontroller & Interfacing, Advanced Computer Architecture, Operating System, Parallel Computing, Hardware Security, Fault-Tolerant Design, Secure Microcontrollers

Job Mapping:

VLSI Engineer, Hardware Design Engineer, Embedded Systems Engineer, Robotics Engineer , System Architect, AI Hardware Engineer

RTOS Developer, IoT Device Developer, Avionics/Defense System Engineer, Industrial Automation Engineer

Books and E-Resources

For Reference Print Book -

1. R.P. Jain; 'Modern Digital Electronics';3rd Edition; Tata McGraw-Hill;2003
2. James Antonakos;"The Pentium Microprocessor"; 2004. Pearson Education ISBN 81- 7808-545-3
3. Bahadure Nilesh;"Microprocessors: The 8086/8088, 80186/80286, 80386/80486 and the Pentium Family";2010

For Reference Electronic Book –

1. M. Mano; 'Digital Design';3rd Edition; Pearson Education; 2002
2. A. Malvino, D. Leach; 'Digital Principles and Applications'; 5th Edition; Tata McGraw Hill; 2003
3. Lyla B Das; 'The x86 Microprocessors: 8086 to Pentium, Multicores, Atom and the 8051 Microcontroller, 2/e: Programming and Interfacing'; 2nd Edition, Kindle Edition

For MOOCs and other learning Resources

1. Prof. Santanu Chattopadhyay; 'Digital Circuits, IIT Kharagpur'; NPTEL
https://nptel.ac.in/courses/108105113

2. Prof. N.J. Rao; 'Digital Systems, IISc Bangalore'; NPTEL;

https://nptel.ac.in/courses/106108099

4. Prof. Santanu Chattopadhyay; 'Microprocessor and Microcontroller", IIT Kharagpur'; NPTEL

https://onlinecourses.swayam2.ac.in/cec21 cs16/preview

FF No. : 654

CS2303: Object Oriented Programming

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Lab: 2 Hours/Week

Course Prerequisites:

1. C programming
2. Problem solving skills using programming language

Course Objectives:

1. Differentiate various programming paradigms.
2. Understand the concept of data abstraction and encapsulation, how to design C++ classes for code reuse.
3. Learn to implement class member functions, constructors and operators overloading in C++.
4. Learn Inheritance, virtual functions and dynamic binding with polymorphism.
5. Utilize file handling techniques in C++ for sequential and random access, while incorporating robust, reliable and efficient data management.
6. Learn how to use exception handling in C++ programs and how to design and implement generic classes with C++ templates. Learn to implement scalable and maintainable code using SOLID principles.

Course Relevance:

The Object-Oriented Programming (OOP) course using C++ is highly relevant because it introduces core programming concepts such as encapsulation, inheritance, and polymorphism, which are essential for modern software development. It helps students understand how to structure and manage complex systems effectively. The course also provides practical skills in writing maintainable and scalable code using one of the most widely used programming languages. Additionally, mastering OOP in C++ lays a strong foundation for learning other OOP languages like Java and Python.

Syllabus

Theory

Section 1: Topics/Contents

Unit 1: Introduction to Object Oriented Programming **(3 Hours)**

Programming Paradigms: Imperative and Declarative, Procedural programming paradigm and its limitations, Need of Object-Oriented Programming, Fundamentals of Object-Oriented Programming: Objects, Classes, Data Members, Methods, Data Encapsulation, Data Abstraction and Information Hiding, Inheritance, Polymorphism, Static and Dynamic Binding, Message Passing.

Unit 2: Classes and Objects **(6 Hours)**

Creating a Class, Visibility/Access Modifiers, Encapsulation, Methods: Adding a Method to Class, returning a Value, adding a Method That Takes Parameters, the 'this' Keyword, Method Overloading, Object Creation, Using Object as a Parameters, Returning Object, Array of Objects, Memory Allocation: 'new', Memory Recovery: 'delete', Static Data Members, Static Methods, Forward Declaration, Classes as Objects.

Unit 3: Constructors and Destructors **(4 Hours)**

Introduction, Use of Constructor, Characteristics of Constructors, Types of Constructors, Constructor Overloading, Dynamic Initialization of an Object, Constructor with Default Arguments, Destructors.

Section 2: Topics/Contents

Unit 4: Inheritance and Polymorphism **(6 Hours)**

Introduction, Need of Inheritance, Types of Inheritance, Benefits of Inheritance, Cost of Inheritance, Constructors in derived Classes, Method Overriding, Abstract Classes and Interfaces. Polymorphism and Software Reuse: Introduction, Types of Polymorphism (Compile Time and Run Time Polymorphism), Mechanisms for Software Reuse, Efficiency and Polymorphism, Friend function

Unit 5: File Handling **(4 Hours)**

Introduction to File Handling, Classes for File Stream Operations, Opening and Closing Files, File Modes and Combinations, File Pointers and Their Manipulators, Sequential Input and Output Operations, Error Handling during File Operations.

Unit 6: Exception Handling, Generic Programming and SOLID **(5 Hours)**

Introduction, basics of Exception Handling, Exception and its Types, Exception-Handling Mechanism, Uncaught Exception, Using try and Catch, Multiple Catch Clauses, Nested Try Statements. Introduction to Generics, Introduction of SOLID principles, Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Dependency Inversion.

Syllabus
Laboratory

List of Experiments

1. Write a program to implement a class for Book details as follows Private: bookid, book_name, author, price Public: get_details(), print_details() Get details of some books from user and print in tabular form. Also find the total prize of all the books.
2. Write a C++ program to implement the concept of objects, classes, constructors, destructors.
3. Design and implement a C++ program to model a banking system where the class BankAccount keeps track of the total number of accounts created and the total balance across all accounts. Use static data members to maintain these values, and static member functions to retrieve and

display them. Add a member function in BankAccount class that takes another BankAccount object as an argument and display the balances in ascending order.

4. Write a C++ program to create a student class with details like Roll no, Name, Marks of five subjects, percentage, class (First, Second, etc) have following functions: parameterized constructor, destructor, Display, Calculate percentage and grade.
5. Design a class 'Complex' with data members for real and imaginary part. Provide default and Parameterized constructors. Write a program to perform arithmetic operations of two complex numbers.
6. Identify commonalities and differences between Publication, Book and Magazine classes. Title, Price, Copies are common instance variables and saleCopy is a common method. The differences are, Bookclass has author and orderCopies(). Magazine Class has methods orderQty, Current issue, receiveissue().Write a program to find how many copies of the given books are ordered and display total sale of publication.
7. Design and develop inheritance for a given case study, identify objects and relationships and implement inheritance wherever applicable. Employee class hasEmp_name, Emp_id, Address, Mail_id, and Mobile_no as members. Inherit the classes: Programmer, Team Lead, Assistant Project Manager and Project Manager from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 52% of BP as DA, 27 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and net salary.
8. Design a base class shape with two double type values and member functions to input the data and compute_area() for calculating area of shape. Derive two classes: triangle and rectangle. Make compute_area() as an abstract function and redefine this function in the derived class to suit their requirements. Write a program that accepts dimensions of triangle/rectangle and displays calculated area. Implement dynamic binding for given case study.
9. Design and develop a context for given case study and implement an interface for Vehicles Consider the example of vehicles like bicycle, car and bike. All Vehicles have common functionalities such as Gear Change, Speed up and apply brakes. Make an interface and put all these common functionalities. Bicycle, Bike, Car classes should be implemented for all these functionalities in their own class in their own way.

10. Implement a program for maintaining a database of student records using Files. Students have Student_id,name, Roll_no, Class, marks and address. Display the data for a few students. 1. Create Database 2. Display Database 3. Delete Records 4. Update Record 5. Search Record
11. Implement a program to handle Arithmetic exceptions, Array Index Out of Bounds.
12. Write a program to extend Book class with dynamic memory allocation. (Use of "new" keyword and object pointers)
13. Implement a generic program using any collection class to count the number of elements in a collection that have a specific property such as even numbers, odd number, prime number and palindromes.
14. Write a generic function template that sorts an array of any type using Bubble Sort or Selection Sort.

Syllabus
Course Project

List of Course Projects

1. Electricity billing system
2. e-Healthcare management system
3. Library management system
4. Online bank management system
5. Online medical management system
6. Online quiz management system
7. Online Survey System
8. Stock management system
9. Supply chain management system
10. Hospital Management System
11. Alumni Database System
12. PayRoll System
13. Simple Inventory System
14. Employee Management System
15. Car Rental System
16. Student Grading System
17. Hotel Reservation System
18. Movie Ticket Booking System

19. Vehicle Parking Management
20. Library Fine Calculation System

Course Outcomes

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

1. Understand object-oriented programming features.
2. Identify classes, objects, methods, and handle object creation, initialization, and destruction to model real-world problems.
3. Understand and implement various types of constructors and destructors.
4. Identify relationships among objects using inheritance and polymorphism principles.
5. Use files for persistent data storage to model real world applications.
6. Handle different types of exceptions, apply generic programming concepts and SOLID principles to write scalable and maintainable code.

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PSO1**	**PSO2**	**PSO3**
CO1	3	2	2									3		
CO2	3	3	3	1	1	2	2	2			2	3		3
CO3	3	2	3	1	1		2					3		3
CO4	3	3	3	1							2	3	3	3
CO5	3	3	3	1	1						2	3		3
CO6	3	2	3									3		3
Average	3	2.5	2.83	1	1	2	2	2			2	3	3	3

Future Courses Mapping:

Advanced Data structure, Operating System, Advanced C++ / Java , Object-Oriented Design (OOD), Design Patterns, Embedded Systems (with C++), CUDA Cyber Security.

Job Mapping:

Software Developer / Engineer, Game Developer, System Programmer, Software Architect

Books and E-Resources

For Reference Print / E-Book -

1. The C++ Programming Language, Bjarne Stroustrup, 4th Edition, Addison-Wesley Pearson Education .
2. Herbert Schildt , “C++: The Complete Reference”, 4th Edition.

3. E. Balaguruswamy, “Object Oriented Programming Using C++ ”, Tata McGraw Hill.
4. An Introduction to Object Oriented Programming (3rd Ed), by Timothy A. Budd, published by Addison-Wesley,2002.

For MOOCs and other learning Resources

1. https://onlinecourses.nptel.ac.in/noc25_cs34/preview
2. https://onlinecourses.nptel.ac.in/noc20_cs07/preview
3. https://www.coursera.org/learn/packt-fundamentals-of-object-oriented-programming-c-b5fxn#modules
4. SOLID principles: implementation and examples in C++ | by Oleksandra Shershen | Medium

FF No. : 654

CS2304: Database Management Systems

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Lab: 2 Hours/Week

Course Prerequisites: Data structures, Discrete Mathematics

Course Objectives:

1. To introduce the core principles and architectures of modern database systems.
2. To apply data modelling techniques using Entity-Relationship and relational models.
3. To design and normalize relational schemas for efficient data storage.
4. To write effective SQL and PL/SQL programs for data manipulation and transaction control.
5. To understand the concepts of query processing, optimization, and indexing.
6. To explore the role of NoSQL databases, Big Data systems, and distributed databases in modern applications.

Course Relevance:

This course builds foundational knowledge and practical skills in database systems, crucial for software development, data analytics, and backend engineering roles. Students will gain exposure to both relational and non-relational databases, optimization techniques, and distributed data management — all of which are essential in today's data-driven industry and high-demand job profiles such as Database Developer, Data Engineer, Backend Developer, and Big Data Analyst.

Section 1: Topics/Contents

Unit 1: Introduction **(4 Hours)**

Introduction: Need of Database Management Systems, Evolution, Database System Concepts and Architecture, Database Design Process Data Modeling: Entity Relationship (ER) Model, Extended ER Model, Relational Model, Codd's Rules;

Unit 2: Database Design **(5 Hours)**

Need of Normalization, Functional Dependencies, Inference Rules, Functional Dependency Closure, Minimal Cover, Decomposition Properties, Normal Forms: 1NF, 2NF, 3NF and BCNF, Multi-valued Dependency, 4NF, Relational Synthesis Algorithms

Unit 3: Query Languages **(5 Hours)**

Query Languages: Relational Algebra, SQL: DDL, DML, Select Queries, Set, String, Date and Numerical Functions, Aggregate Functions ,Group by and Having Clause, Join Queries, Nested queries, DCL, TCL, PL/SQL: Procedure, Function, Trigger, Mapping of Relational Algebra to SQL, Assertions, roles and privileges , Embedded SQL, Dynamic SQL.

Section 2: Topics/Contents

Unit 4: Storage and Querying (4 Hours)

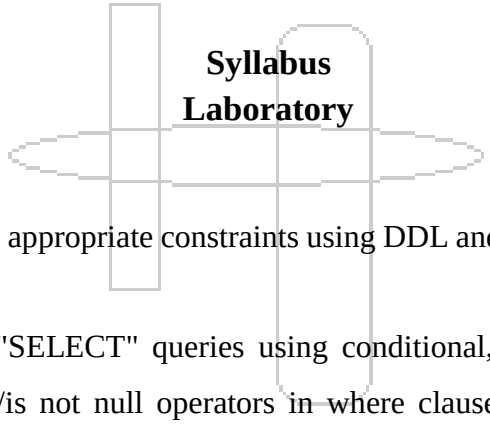
Storage and Querying: Storage and File structures, Indexed Files, Single Level and Multi Level Indexes; Query Processing, Query Optimization, Query Cost Estimation Parquet file format. Transaction Management: Basic concept of a Transaction, ACID Properties, State diagram, Concept of Schedule, Serializability – Conflict and View, Concurrency Control Protocols, Recovery techniques

Unit 5: NOSQL Databases and Big Data Storage Systems (5 Hours)

NOSQL Databases and Big Data Storage Systems: Introduction to NOSQL Databases, Types of NOSQL Databases, BASE properties, CAP theorem, CRUD operation in MongoDB, Big Data, HADOOP: HDFS, MapReduce. Data Warehousing: Architecture and Components of Data Warehouse, Warehouse Schemas, OLAP

Unit 6: Parallel and Distributed Databases (5 Hours)

Parallel and Distributed Databases: Architecture, I/O Parallelism, Interquery, Intraquery, Intraoperation and Interoperation Parallelism, Types of Distributed Database Systems, Distributed Data Storage, Distributed Query Processing, Introduction to Elastic Search index.



List of Experiments

1. Create a database with appropriate constraints using DDL and populate/modify it with the help of DML.
2. Design and Execute "SELECT" queries using conditional, logical, like/not like, in/not in, between...and, is null/is not null operators in where clause, order by, group by, aggregate functions, having clause, and set operators. Use SQL single row functions for date, time, string etc.
3. Write equijoin, non equijoin, self join and outer join queries. Write queries containing single row / multiple row / correlated sub queries using operators like =, in, any, all, exists etc. Write DML queries containing sub queries. Study a set of query processing strategies.
4. Write PL/SQL blocks to implement all types of cursor.
5. Write useful stored procedures and functions in PL/SQL to perform complex computation.
6. Write and execute all types of database triggers in PL/SQL.

7. Execute DDL statements which demonstrate the use of views. Try to update the base table using its corresponding view. Also consider restrictions on updatable views and perform view creation from multiple tables.
8. Create a database with suitable example using MongoDB and implement Inserting and saving document, Removing document, Updating document
9. Execute at least 10 queries on any suitable MongoDB database that demonstrates following querying techniques: find and findOne, Query criteria, Type-specific queries
10. Implement Map Reduce operation with suitable example using MongoDB

Syllabus
Course Project

List of Course Projects

1. Library Management System using SQL and PL/SQL.
2. Online Shopping Cart with MySQL plus MongoDB
3. Student Performance Analytics using OLAP
4. Mini Social Network Data Storage
5. Query Optimizer Simulator
6. Distributed Employee Database System
7. Crime Record Management using Elasticsearch
8. COVID-19 Data Warehouse with Hadoop

Course Outcomes

1. Design and construct conceptual database models using ER and EER diagrams for real-life applications.
2. Transform high-level data models into normalized relational schemas using functional dependencies and synthesis techniques.
3. Apply the concepts of normalization to develop the quality relational data model
4. Formulate and execute queries using relational algebra, SQL, and develop procedural constructs using PL/SQL.
5. Explore and implement modern database technologies such as NoSQL and Big Data frameworks like MongoDB and Hadoop.
6. Demonstrate understanding of physical database structures, indexing mechanisms, and query optimization techniques

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
CO1	2	3	3		2				2	2	2	2		
CO2	2	2	3		2				2	2	2	2	3	2
CO3	2	3	3						2	2	2	2	3	2
CO4	2	3	3	2					2		2	2		2
CO5	2	-	3						2		2	2	3	2
CO6	2	-									2	2		2
Average	2	2.75	3	2	2				2	2	2	2	3	2

CO attainment levels

CO1:1 CO2:4 CO3:2 CO4:4 CO5:3 Co6:5

Future Courses Mapping:

Advanced databases
Big Data Management
Cloud Databases
Database Administrator

Job Mapping:

Database Engineer
SQL developer
PL/SQL developer
Data Analyst
Data Engineer

Text Books:

1. Abraham Silberschatz, Henry F. Korth, S. Sudarshan; ―Database System Concepts‖; 6th Edition, McGraw-Hill Education
2. Ramez Elmasri, Shamkant B. Navathe; ―Fundamentals of Database Systems‖; 7th Edition, Pearson

Reference Books:

1. Thomas M. Connolly, Carolyn E. Begg, "Database Systems: A Practical Approach to Design", Implementation, and Management, 6th Edition ;Pearson
2. Raghu Ramakrishnan, Johannes Gehrke; Database Management Systems‖, 3rd Edition; McGraw Hill Education
3. Kristina Chodorow, "MongoDB The definitive guide", O'Reilly Publications, ISBN: 978-93-5110-269-4, 2nd Edition.
4. Dr. P. S. Deshpande, "SQL and PL/SQL for Oracle 10g Black Book", DreamTech.
5. Ivan Bayross, SQL, PL/SQL: The Programming Language of Oracle, BPB Publication.
6. Reese G., Yarger R., King T., "Williums H, Managing and Using MySQL", Shroff Publishers and Distributors Pvt. Ltd., ISBN: 81 - 7366 - 465 – X, 2nd Edition.

7. Dalton Patrik, “SQL Server – Black Book”, DreamTech Press.
8. Eric Redmond, Jim Wilson, “Seven databases in seven weeks”, SPD, ISBN: 978-93-5023-918-6.
9. Jay Kreibich, Using SQLite, SPD, ISBN: 978-93-5110-934-1, 1st edition.

For MOOCs and other learning Resources

1. https://nptel.ac.in/courses/106/105/106105175/
2. https://onlinecourses.nptel.ac.in/noc21_cs04/preview
3. https://www.datacamp.com/courses/introduction-to-sql
4. Oracle MOOC: PL/SQL Fundamentals - Oracle APEX

FF No. : 654

Multidisciplinary Minor**

CSM001: Discrete Structure and Graph Theory

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Tutorial: 1 Hour/Week

Course Prerequisites: Basic understanding of school mathematics

Course Objectives:

1. To study basic discrete structures (such as functions, relations, sets, graphs, and trees)
2. To express mathematical properties via the formal language of propositional and predicate logic
3. To develop recurrence relations for a wide variety of combinatorial problems
4. To study advanced combinatorial techniques
5. To study elementary topics in number theory
6. To study elementary concepts in graph theory

Course Relevance: This is a foundational course for Computer Science and Engineering. The discrete structures play an essential role while modeling problems in computer science and engineering. The reasoning with different discrete structures is useful in understanding the underlying computer science problem more concretely. The course also builds problem solving ability.

Syllabus

Theory

Section 1: Topics/Contents

Unit 1: Logic and Proofs **(4 Hours)**

Propositional logic, propositional equivalences, truth tables, predicates and quantifiers, rules of inference, introduction to proofs: direct, contraposition, contradiction, counterexamples, principle of mathematical induction, strong induction. Proving correctness of programs.

Unit 2: Elementary Discrete Structures and Basic Counting **(5 Hours)**

Elementary set theory (sets, set builder notation, cardinality, subsets, some finite and infinite sets, operations on sets), relations (relations and their properties, representing relations, closure of relations, equivalence relations), functions, partial orders, basic counting principles, permutations, combinations, generalized permutations and combinations (with/without repetitions, distinguishable/indistinguishable objects), Binomial coefficients and identities.

Unit 3: Advanced Combinatorial Techniques (5 Hours)

Double counting, combinatorial proof technique, Pigeon-Hole Principle, generalized pigeon-hole principle, some applications from: Ramsey theorem, Mantel's theorem, Turan's theorem, Erdos-Szekeres theorem.

Inclusion Exclusion Principle: Counting with Venn Diagrams, counting Derangements, number of primes up to n, number of onto functions, Euler's phi function.

Section 2: Topics/Contents

Unit 4: Recurrence relations and Generating Functions (5 Hours)

Recurrence relations, modelling using recurrence relations, some examples from: Fibonacci numbers, Catlan numbers, Derangements, Tower of Hanoi, partitions, solution of linear recurrence relations with constant coefficients (homogenous and non-homogenous), generating functions and their application in counting.

Unit 5: Modular Arithmetic (4 Hours)

Divisibility and modular arithmetic, Division Algorithm, primes, greatest common divisor, Euclid's Algorithm, extended Euclid's algorithm, modular inversion, Fundamental Theorem of Arithmetic, Congruence's, Fermat's little theorem, Euler's phi function, Chinese remainder theorem.

Unit 6: Graph Theory (5 Hours)

Graphs, different representations, properties of incidence and adjacency matrices, directed/undirected graphs, connected components, degree of a vertex, paths, cycles in graph, Euler and Hamiltonian tours/graphs, Trees, bipartite graphs (graph with only odd cycles, 2-colorable graphs), Planar graphs, Theorem on bound on number of edges, Graph colorings, matching in bipartite graphs

Syllabus
Tutorials

List of Tutorials

1. Problem solving based on propositional logic
2. Problem solving based on basic set theory
3. Problem solving based on relations and functions
4. Problem solving based on basic counting principles
5. Problem solving based on properties of binomial coefficients
6. Problem solving based on permutations, combinations
7. Problem solving based on combinatorial proof technique
8. Problem solving based on double counting
9. Problem solving based on pigeon-hole principle

10. Problem solving based on inclusion exclusion principle
11. Problem solving based on modular arithmetic
12. Problem solving based on recurrence relations
13. Problem solving based on generating functions
14. Problem solving based on graphs and their properties

Course Outcomes

1. Reason mathematically about elementary discrete structures (such as functions, relations, sets, graphs, and trees) used in computer algorithms and systems
2. Express mathematical properties via the formal language of propositional and predicate logic
3. Develop recurrence relations for a wide variety of combinatorial problems
4. Demonstrate use of advanced combinatorial techniques
5. Describe elementary concepts in modular arithmetic and their applications
6. Exhibit understanding of basic graph theory and its applications

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PSO1**	**PSO2**	**PSO3**
CO1	3	1		1							2	2		
CO2	3	1				1			2		2	2	1	
CO3	3	2	3								2	2		
CO4	3	3	3	3							2	2		
CO5	3	3	1			1	2		1		2	2	1	
CO6	3	3	3	2		1		1			2	2	1	
Average	**3**	**2.3**	**1.6**	**1**	**0**	**0.5**	**0.3**	**0.1**	**0.5**		**2**	**2**	**0.5**	**0**

Future Courses Mapping: Data structures, Design and analysis of algorithms, theory of computation, artificial intelligence, machine learning.

Job Mapping:

Wherever one wants to model a computer science problem concretely the use of discrete structures is essential. Due to abstract nature of the course, the principles learnt have wide applicability. In any job which requires algorithmic thinking, programming, use of data structures, the knowledge of discrete structures is very helpful.

Text Books:

1. *"Discrete Mathematics and its applications" by Kenneth Rosen (William C Brown Publisher)*
2. *"Applied Combinatorics" by Alan Tucker (Wiley Publishing company)*
3. *"Combinatorics: Topics, techniques, algorithms" by Peter J. Cameron (Cambridge University Press)*
4. *Graph Theory by Reinhard Diestel (Springer Verlag Publishing Company)*
5. *Introduction to Graph Theory by Douglas B. West (Prentice-Hall publishers)*

FF No. : 654

HS2002, HS2003: From Campus To Corporate – 1,2

Credits:.2 **Teaching Scheme: Lab: 2 hours/Week**

Introduction to the Corporate World Understanding organizational structure and hierarchy, Work culture differences: campus vs. corporate,
Employer expectations from fresh graduates, Time management and ownership in corporate settings

Professional Communication Skills: Verbal and non-verbal communication, Email and business writing etiquette, Presentation skills and use of visual aids, Listening skills and telephone etiquette,

Soft Skills and Interpersonal Effectiveness: Body language, grooming, and first impressions, Conflict resolution and negotiation skills, Team dynamics and collaboration, Assertiveness vs. aggressiveness

Resume Building and Job Preparation : Building an effective resume and cover letter, Identifying strengths and achievements, Preparing for technical and HR interviews, Handling rejections and feedback
Group Discussions and Personal Interviews : Group discussion formats and evaluation criteria, Strategies for initiating, contributing, and summarizing, Mock interviews with feedback, STAR technique for answering behavioral questions,
Corporate Etiquette and Workplace Ethics: Meeting and greeting protocol, Dining and social etiquette, Work ethics, punctuality, confidentiality, Respect for diversity and inclusion in the workplace
Adaptability and Emotional Intelligence: Handling pressure, deadlines, and ambiguity, Self-awareness and emotional regulation, Empathy and workplace relationships, Managing feedback and continuous learning,
Introduction to Project Management Basics : Understanding tasks, milestones, deadlines, Collaboration using tools like Trello, Slack, Teams, Basics of Agile/Scrum concepts, Reporting and escalation protocol

Faculty are suppose to do conduct following in the class

- Resume and LinkedIn profile workshops
- Mock interviews and GD sessions
- Role plays: workplace scenarios, conflict handling
- Business email writing exercises
- Presentation and elevator pitch sessions

Books:

1. Dale Carnegie, How to Win Friends and Influence People
2. Stephen R. Covey, 7 Habits of Highly Effective People
3. Shital Kakkar Mehra, Business Etiquette: A Guide for the Indian Professional
4. Peggy Klaus, The Hard Truth About Soft Skills

FF No. : 654

HS2001, HS2004: Reasoning and Aptitude Development – 3, 4

Credits:.1 **Tut: 1 Hour/Week**

Unit 1: English Language

Familiarity with English Language, Ability to understand written text, spoken word and effective communication through written documents; Coverage of vocabulary to cope up with general and specific terminology, syntax and sentence structure, prevention of incorrect use leading to distortion in communication; synonyms, antonyms and contextual vocabulary, Grammar – Error identification, sentence improvement and construction, Reading Comprehension

Unit 2: Logical Ability

Objective interpretation of things, ability to perceive and interpret trends to make generalizations; ability to analyze assumptions behind an argument or statement; Deductive reasoning: Assessment of ability to synthesize information and derive conclusions - Coding deduction logic, Data Sufficiency, Directional Sense, Logical word sequence, Objective reasoning, Selection and decision tables, puzzles; Inductive reasoning: Assessment of ability to learn by example, imitation or by trial – Analogy pattern recognition, Classification pattern recognition, Coding pattern recognition, Number series pattern recognition; Abductive reasoning: Critical thinking ability of seeing through logical weak links or loopholes in an argument or a group of statements; Critical reasoning: assessment of ability to think through and analyze logical arguments, assessment of ability to use logical constructs to offer reasoning in unfamiliar situations; Information Gathering and synthesis: Ability of locating information, information ordering, rule based selection and data interpretation, order and classify data, interpret graphs, charts, tables and make rule based deductions. Application of these approaches for using visual, numerical and textual data from single or multiple sources

Unit 3: Quantitative Ability

Basic numbers – decimals and fractions, factorization, divisibility: HCF, LCM, Odd, even, prime and rational numbers. Application of algebra to real world, direct and inverse proportion, common applications – Speed-time -distance, Profit-loss, percentage, age relations, mixtures, other miscellaneous quantitative combination, exponentials and logarithms, permutations and combinations, probability. Spatial reasoning: Inductive – Missing portions, Sequence and series; Deductive analysis.

Reference Books –

1. "English Grammar in Use" by Raymond Murphy, Cambridge University Press.
2. "Word Power Made Easy" by Norman Lewis, Goyal Publishers & Distributors.
3. "Objective General English" by S.P. Bakshi, Arihant Publications.
4. "English for Competitive Examinations" by K. Sinha, S. Chand Publishing.
5. "Essential English Grammar" by Philip Gucker, Wiley.
6. "English Idioms and Phrasal Verbs" by M.A. Yadav, Vikas Publishing House.
7. "The Oxford English Grammar" by Sidney Greenbaum, Oxford University Press.

8. "A Modern Approach to Verbal & Non-Verbal Reasoning" by R.S. Aggarwal, S. Chand Publishing, ISBN: 978-8121903409.
9. "Logical Reasoning and Data Interpretation for the CAT" by Nishit K. Sinha, Pearson India, ISBN: 978-8131709117.
10. "Logical Reasoning and Data Interpretation for the CAT" by Arun Sharma, McGraw Hill Education, ISBN: 978-0070709642.
11. "A New Approach to Reasoning Verbal and Non-Verbal" by B.S. Sijwali & Indu Sijwali, Arihant Publications, ISBN: 978-9311124692.
12. "Quantitative Aptitude for Competitive Examinations" by R.S. Aggarwal, S. Chand Publishing, ISBN: 978-8121900637.
13. "How to Prepare for Quantitative Aptitude for the CAT" by Arun Sharma, McGraw Hill Education, ISBN: 978-0070709642.
14. "The Pearson Guide to Quantitative Aptitude for Competitive Examination" by Pearson, Pearson India, ISBN: 978-8131709117.
15. "Quantitative Aptitude for Competitive Examinations" by Abhijit Guha, Tata McGraw Hill Education, ISBN: 978-0070666653.
16. "Data Interpretation & Data Sufficiency" by R.S. Aggarwal, S. Chand Publishing ISBN: 978-8121903515.
17. "Quantitative Aptitude for Competitive Examinations" by S. Chand, S. Chand Publishing, ISBN: 978-8121903423.

Course Outcomes:

Upon completion of the course, the student will be able to –

1. Improve the reading, writing and verbal skills, and enhance comprehension and articulation abilities
2. Develop logical reasoning abilities, enabling them to make sound decisions in problem-solving scenarios
3. Develop mathematical aptitude as well as data interpretation abilities and use them in test cases and real world problems
4. Learn to apply approaches for optimum time-management, prioritization maximizing the accuracy
5. Learn data interpretation, apply mathematical skills to draw accurate conclusions
6. Apply their knowledge of English, reasoning and quantitative skills for planning, critical thinking and real world problems

CO-PO Map

	Program Outcomes (PO)												PSO			
CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
CO1	2	2	2	2					3		2	2				3
CO2	2	2	3	2	2		2		3		2	2	3		3	3
CO3	2	2	3	2	3		2		3		2	2	3		3	3
CO4	2	2	3	2	3	3		2	3		2	2	3	3	3	3
CO5	2	2	3	2	3	2			3		2	2	3		3	3
CO6	2	2	3	3	2				3		3	2	3		3	3
Average	**2.0**	**2.0**	**2.83**	**2.83**	**2.6**	**2.5**	**2.0**	**2.0**	**3.0**	**1.0**	**2.16**	**2.0**	**3.0**	**3.0**	**3.0**	**3.0**

FF No. : 654

CS2306, CS2313: Design Thinking – 1, 2

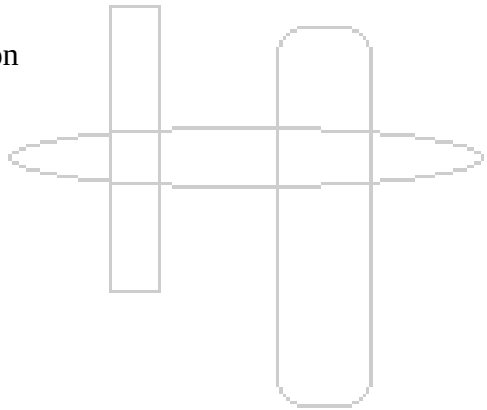
Credits: 1 **Teaching Scheme: Tutorial 01 Hour/week**

Course Prerequisites: Problem Based Learning, Project Centric Learning

Course Objective:
To provide ecosystem for students and faculty for paper publication and patent filing

Section 1: Topics/Contents

What is research?
Importance of Paper Publication and Patents
Structure of Paper
Journal Publication
Publication in conference
Literature Review
Research Paper Writing
Journal Ratings and Evaluation
How to rate a Journal?
Intellectual property (IP)
Research Ethics
Entrepreneurship



Section 2: Topics/Contents

Structure of The paper
Journal List (Top 50 Journals)
Selection of the journal
Use of various online journal selection tools
Plagiarism checking
Improving contents of the paper
Patent drafting
Patent search
Filing of patent
Writing answers to reviewer questions
Modification in manuscript
Checking of publication draft

Course Outcomes: **[**Publication of paper or patent]
The student will be able to
1. Understand the importance of doing Research

2. Interpret and distinguish different fundamental terms related to Research
3. Apply the methodology of doing research and mode of its publication
4. Write a Research Paper based on project work
5. Understand Intellectual property rights
6. Use the concepts of Ethics in Research
7. Understand the Entrepreneurship and Business Planning

CO-PO Map:

	Program Outcomes (PO)												PSO			
CO/PO	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
CO1	1	1	1	1	1	--	--	--	--	--	--	1	1	2	2	3
CO2	1	1	1	1	1	--	--	--	--	--	--	1	2	1	1	3
CO3	2	2	3	3	2	2	1	2	2	3	--	1	2	2	3	3
CO4	3	3	3	3	3	2	1	2	2	3	1	1	-	-	2	3
CO5	1	1	1	1	1	--	--	--	--	--	--	1	-	-	1	2
CO6	2	2	2	2	2	2	1	3	2	3	--	1	2	2	2	3
C07	1	1	1	1	1	--	--	--	--	--	--	1	1	1	1	1
Average	**1.57**	**1.57**	**1.71**	**1.71**	**1.57**	**2.0**	**1.0**	**2.33**	**2.0**	**3.0**	**1.0**	**1.0**	1.66	1.66	1.71	2.5

Syllabus Template **FF No. : 654**

CS2307, CS2314: Engineering Design And Innovation – 1, 2

Credits:2 **Teaching Scheme: 8 Hours/Week**

Course Prerequisites: Problem Based Learning

Course Objectives:

1. To develop critical thinking and problem solving ability by exploring and proposing solutions to realistic/social problems.
2. To Evaluate alternative approaches, and justify the use of selected tools and methods,
3. To emphasize learning activities those are long-term, inter-disciplinary and student-centric.
4. To engage students in rich and authentic learning experiences.
5. To provide every student the opportunity to get involved either individually or as a group so as to develop team skills and learn professionalism.
6. To develop an ecosystem to promote entrepreneurship and research culture among the students

Course Relevance: Project Centric Learning (PCL) is a powerful tool for students to work in areas of their choice and strengths. Along with course based projects, curriculum can be enriched with semester long Engineering Design and Development courses, in which students can solve socially relevant problems using various technologies from relevant disciplines. The various socially relevant domains can be like Health care, Agriculture, Defense, Education, Smart City, Smart Energy and Swaccha Bharat Abhiyan. To gain the necessary skills to tackle such projects, students can select relevant online courses and acquire skills from numerous sources under guidance of faculty and enrich their knowledge in the project domain, thereby achieving project centric learning. Modern world sustained and advanced through the successful completion of projects. In short, if students are prepared for success in life, we need to prepare them for a project-based world. It is a style of active learning and inquiry-based learning. Project centric learning will also redefine the role of teacher as mentor in the learning process. The PCL model focuses the student on a big open-ended question, challenge, or problem to research and respond to and/or solve. It brings students not only to know, understand and remember rather it takes them to analyze, design and apply categories of Bloom's Taxonomy.

Course Relevance: Project Centric Learning (PCL) is a powerful tool for students to work in areas of their choice and strengths. Along with course based projects, curriculum can be enriched with semester long Engineering Design and Development courses, in which students can solve socially relevant problems using various technologies from relevant disciplines. The various socially relevant domains can be like Health care, Agriculture, Defense, Education, Smart City, Smart Energy and Swaccha Bharat Abhiyan. To gain the necessary skills to tackle such projects, students can select relevant online courses and acquire skills from numerous sources under guidance of faculty and enrich their knowledge in the project domain, thereby achieving project centric learning. Modern world sustained and advanced through the successful completion of projects. In short, if students are prepared for success in life, we need to prepare them for a project-based world. It is a style of active learning and inquiry-based learning. Project based learning will also redefine the role of teacher as mentor in the

learning process. The PCL model focuses the student on a big open-ended question, challenge, or problem to research and respond to and/or solve. It brings students not only to know, understand and remember rather it takes them to analyze, design and apply categories of Bloom's Taxonomy.

Preamble - The content and process mentioned below is the guideline document for the faculties and students to start with. It is not to limit the flexibility of faculty and students; rather they are free to explore their creativity beyond the guideline mentioned herewith. For all courses of ED, laboratory course contents of "Engineering Design" are designed as a ladder to extend connectivity of software technologies to solve real word problem using interdisciplinary approach. The ladder in the form of gradual steps can be seen as below:

Industry Communication Standards, Single Board Computers and IoT, Computational Biology (Biomedical and Bioinformatics), Robotics and Drone, Industry 4.0 (Artificial Intelligence, Human Computer Interfacing, 5G and IoT, Cloud Computing, Big Data and Cyber Security etc).

Group Structure:

- There should be a team/group of 4-5 students.
- A supervisor/mentor teacher assigned to individual groups.
- It is useful to group students of different abilities and nationalities together.

Selection of Project/Problem:

- Students must focus to initiate the task/idea .The idea inception and consideration shall be from following areas as a real world problem:
- Health Care, Agriculture, Defense, Education, Smart City, Smart Energy, Swaccha Bharat Abhiyan, Environment, Women Safety.
- This is the sample list to start with. Faculty and students are free to include other areas which meet the society requirements at large.
- The model begins with the identifying of a problem, often growing out of a question or "wondering". This formulated problem then stands as the starting point for learning. Students design and analyze the problem/project within an articulated disciplinary subject frame/domain.
- A problem can be theoretical, practical, social, technical, symbolic, cultural, and/or scientific and grows out of students' wondering within different disciplines and professional environments. A chosen problem has to be exemplary. The problem may involve an interdisciplinary approach in both the analysis and solving phases.
- By exemplarity, a problem needs to refer back to a particular practical, scientific, social and/or technical domain. The problem should stand as one specific example or manifestation of more general learning outcomes related to knowledge and/or modes of inquiry.

Teacher's Role in PCL :

- Teacher is not the source of solutions rather he will they act as the facilitator and mentor.
- To utilize the principles of problems solving, critical thinking and metacognitive skills of the students.
- To aware the group about time management.
- Commitment to devote the time to solve student's technical problems and interested in helping students to empower them better.

Student's Role in PCL:

- Students must have ability to initiate the task/idea .they should not be mere imitators.
- They must learn to think.
- Students working in PCL must be responsible for their own learning.
- Students must quickly learn how to manage their own learning, Instead of passively receiving instruction.

- Students in PCL are actively constructing their knowledge and understanding of the situation in groups.
- Students in PCL are expected to work in groups.
- They have to develop interpersonal and group process skills, such as effective listening or coping creatively with conflicts.

Developing Inquiry Skills:

- Students in PCL are expected to develop critical thinking abilities by constantly relating: What they read to do? What they want to do with that information?
- They need to analyze information presented within the context of finding answers.
- Modeling is required so that the students can observe and build a conceptual model of the required processes.
- Use the following mechanism to maintain the track of moving towards the solution.
  How effective is …….? How strong is the evidence for ……? How clear is ……?
- What are the justifications for thinking? Why is the method chosen?
- What is the evidence given to justify the solution?

Literature Survey – To avoid reinvention of wheel:

- It is integral part of self- directed learning
- Identify the information needed to solve a given problem or issue
- Be able to locate the needed information
- Use the information to solve the given problem effectively.
- Skills required by students in information literacy include:
- · How to prepare the search? How to carry out the research
- Sorting and assessing of information in general

Use of Research Methodology: - investigation, collaboration, comprehension, application, analysis, synthesize and evaluation

Focus on following skills while working in a team to reach to solution:

- Collaborative learning
- Interpersonal Skills
- Resources Evaluation
- Metacognitive Skills
- Reflection Skills

EDD Sample Case Studies : -

With the adaptation of industry communication standards, Raspberry Pi and Sensors, following projects can be taken up:

1) Design a deployable product for soil moisture detection
2) Design a deployable product for temperature detection
3) Design a deployable product for pressure detection
3) Design a deployable product smoke detection
4) Design a deployable product for motion detection
5) Design a deployable product for collision detection
6) Design a deployable product for sound detection

...not limited to.....Faculty and students are free to include other areas which meet the society requirements at large.

Suggest an assessment Scheme:

Suggest an Assessment scheme that is best suited for the course. Ensure 360 degree assessment and check if it covers all aspects of Bloom's Taxonomy.

To focus on the higher levels of the Booms Taxonomy analyze, apply, evaluate and create.

Text Books: ***(As per IEEE format)***

1.A new model of problem based learning. By Terry Barrett. All Ireland Society for higher education (AISHE). ISBN:978-0-9935254-6-9; 2017

2.Problem Based Learning. By Mahnazmoallem, woei hung and Nada Dabbagh, Wiley Publishers. 2019.

3. Stem Project based learning and integrated science, Technology, Engineering and mathematics approach. By Robert Robart Capraro, Mary Margaret Capraro

Reference Books: ***(As per IEEE format)***

1.De Graaff E, Kolmos A., red.: Management of change: Implementation of problem-based and project-based learning in engineering. Rotterdam: Sense Publishers. 2007.

2.Project management core textbook, second edition, Indian Edition , by Gopalan.

3. The Art of Agile Development. By James Shore & Shane Warden.

MOOCs Links and additional reading material: www.nptelvideos.in

https://worldwide.espacenet.com/

Course Outcomes:

On completion of the course, learner will be able to–

1. Identify the real life problem from societal need point of view
2. Choose and compare alternative approaches to select most feasible one
3. Analyze and synthesize the identified problem from technological perspective
4. Design the reliable and scalable solution to meet challenges
5. Evaluate the solution based on the criteria specified
6. Inculcate long life learning attitude towards the societal problems

CO PO Map

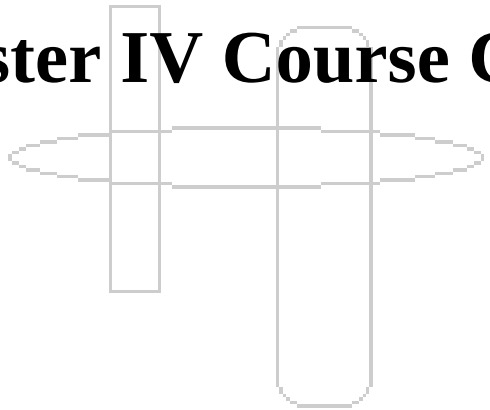
	Program Outcomes (PO)												PSO			
CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
CO1	2	2	2	2					3		2	2				3
CO2	2	2	3	2	2		2		3		2	2	3		3	3
CO3	2	2	3	2	3		2		3		2	2	3		3	3
CO4	2	2	3	2	3	3		2	3		2	2	3	3	3	3
CO5	2	2	3	2	3	2			3		2	2	3		3	3
CO6	2	2	3	3	2				3		3	2	3		3	3
Average	**2.0**	**2.0**	**2.83**	**2.83**	**2.6**	**2.5**	**2.0**	**2.0**	**3.0**	**1.0**	**2.16**	**2.0**	**3.0**	**3.0**	**3.0**	**3.0**

CO attainment levels

CO1 -4 CO2 –2 CO3-4 CO4-5 CO5 -1 CO6-3

S. Y. B. Tech. Computer Engineering
AY 2025-26

Semester IV Course Content



FF No. : 654

CS2308: Data Structures-II

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Lab: 2 Hours/Week

Course Prerequisites: Data Structures, Basic programming Skills (C/C++).

Course Objectives:

1. To Learn how and when to use data structures like AVL trees, red-black trees, B-trees, Fibonacci heaps, and others.
2. To Apply advanced data structures to solve complex computational problems efficiently.
3. To Gain hands-on experience in implementing and using these structures through programming assignments.
4. To Foster the ability to design new data structures or improve existing ones for specialized needs.
5. To prepare students for algorithm-heavy roles that require knowledge of advanced structures.

Course Relevance:

The course on Advanced Data Structures is highly relevant for several reasons, particularly in the context of computer science, software engineering, and related fields. The course is essential for anyone aiming to build efficient, scalable, and optimized software systems or preparing for roles in software development, data science, and academia. Advanced data structures (e.g., segment trees, tries, B-trees, red-black trees, Fibonacci heaps) are essential for optimizing time and space complexity in complex algorithms. Understanding advanced data structures is crucial for solving high-level problems in coding competitions and technical interviews. Used in databases (e.g., B-trees in indexing), compilers (e.g., syntax trees), operating systems (e.g., scheduling queues), and networks (e.g., tries in routing). Key for designing scalable systems (e.g., caching systems, real-time data processing) where performance and memory management are critical. Advanced structures are often the basis for innovations in algorithms, machine learning, data mining, and bioinformatics.

Syllabus

Theory

Section 1: Advanced Trees, Priority Queues and DS for Strings.

Unit 1: Advanced Trees and Applications (8 Hours)

Threaded Binary Tree, AVL Tree, Red-Black Tree, Heap Tree, Huffman Tree. B-Tree, B+-Tree, Splay Tree, Van Emde Boas Tree, Fusion Tree, Dynamic Finger Search Trees with Time and Space Complexity Analysis

Unit 2: Priority Queues and Heaps **(4 Hours)**

Double Ended Priority queues, Leftist Trees, Binomial Heaps, Fibonacci Heaps, skew heaps, pairing heaps

Unit 3: Data Structures for Strings **(4 Hours)**

String Searching: preliminaries, the DAWG, the position Heaps, tries and compressed tries, Suffix Trees and suffix arrays, Dictionaries Allowing Errors in Queries.

Section 2: Randomized, Multidimensional and Miscellaneous Data Structures

Unit 4: Randomized Data Structures **(6 Hours)**

Preliminaries of randomized algorithm and probability theory, Skip Lists: Structural Properties of Skip Lists, Space Complexity of skip list. Treap: A Randomized Binary Search Tree

Unit 5: Multidimensional Spatial Data Structures **(5 Hours)**

Introduction, point data, region data and Rectangle data. Quad trees and Octrees: Quad trees for point data, spatial queries with region quad tree. Interval trees, Segment trees, Range trees, and Priority Search Trees. Binary Space Partitioning Trees, R-trees.

Unit 6: Miscellaneous Data Structures **(3 Hours)**

Google's Big Table, Data Structures for Sets: The Disjoint Set Union-Find Problem, Concurrent Data structures, Succinct Representation of Data Structures: Bit vector, Succinct Dictionaries, Tree Representations. Persistent data structures. Cache-Oblivious Data Structure.

Syllabus
Laboratory

List of Experiments

1. Assignment based on TBT creation and performing Stackless Traversals.
2. Assignment based on operations on AVL and RED-Black trees
3. Assignment based on B Trees and B+ Trees.
4. Assignment based on Priority Queues Application
5. Assignment based on tries.
6. Assignment based on Suffix Trees.
7. Assignment Based on Randomized Data Structures.
8. Assignment based on Quad trees and Oct trees
9. Assignment based on Interval trees, Segment trees, Range trees.
10. Assignment based on R-trees.
11. Assignment Based on Disjoint Set data structures.

12. Assignment based on concurrent data structures.
13. Assignment based on Succinct data structures.

Syllabus
Course Project

List of Course Projects

1) Performance Comparison of AVL and Red-Black Trees on Dynamic Data Sets
2) Visualizing Rotations in AVL and Red-Black Trees
3) Designing a Priority Queue System Using Red-Black Trees
4) Implementing a Self-Balancing Dictionary Using AVL Trees
5) Implementation of Binomial Heaps and Their Use in Priority Queues
6) Comparative Study: Binomial Heap vs Binary Heap vs Fibonacci Heap
7) Task Scheduling Simulation Using Binomial Heaps
8) Analysis of Union and Decrease-Key Operations in Binomial Heaps
9) Implementing a Simple Database Index Using B-Trees
10) File System Directory Management Using B+ Trees
11) Comparison of Search Performance: B-Tree vs B+ Tree on Disk-Based Data
12) Building a Block-Oriented Key-Value Store Using B+ Trees
13) Implementation and Performance Analysis of R-Trees for Spatial Indexing
14) Design and Application of KD-Trees for Nearest Neighbor Search
15) Spatial Query Optimization Using Quadtrees in 2D Geographic Data
16) Building a 3D Game World Using Octrees for Scene Management
17) Multidimensional Range Search Using Range Trees
18) Efficient Spatial Joins Using Grid-Based Indexing
19) Comparison of Spatial Indexing Techniques: R-Tree vs. QuadTree vs. KD-Tree
20) Real-Time Object Tracking Using Spatial Hashing
21) Clustering Geospatial Data Using k-d Trees and Voronoi Diagrams
22) Designing a Mini GIS Engine with Support for Spatial Queries

Course Outcomes

The student will be able to –

1) Analyze and implement advanced binary tree structures.
2) Model the real world problem with the help of appropriate priority queues and heap data structure.
3) Construct and evaluate and string processing structures for real world applications.
4) Apply randomized data structures for optimized performance.
5) Model and query spatial and multidimensional data effectively.
6) Critically evaluate the specialized and modern data structures for complex real-world problems.

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PSO1**	**PSO2**	**PSO3**
CO1	2	2	3								2	3		2
CO2	2	3	2								2	3		2
CO3	3	3	3			2	2				2	3	2	2
CO4	3	3	3	3		2	2				2	3	2	2
CO5	3	3	2								2	3		2
CO6	2	3	3								2	3		2
Average	2.5	3.0	2.66	3.0		2.0	2.0				2.0	3.0	2.0	2.0

CO Attainment levels:

Weights for attainment levels: L1 - Easy-0.75 L2 - Comfortatble-0.7 L3 – Medium – 0.65
L4 – Somewhat difficult – 0.6 L5 – Difficult – 0.55

CO1 – L3, CO2– L3, CO3 – L2, CO4 – L4, CO5 – L4 and CO6 – L5

Future Courses Mapping:

Design and Analysis of Algorithms, Computational Geometry, Data Mining and Information Retrieval, Machine Learning and Data Science, Parallel and Distributed Computing, Compiler Design, Advanced Topics in Data Structures and Algorithms.

Job Mapping:

Students who have completed a course in Advanced Data Structures has a bright career paths and roles where the knowledge and skills from the course are directly applicable are Software Developer / Software Engineer, Data Engineer, Backend Developer, Database Developer / Administrator, Algorithm Engineer, Competitive Programmer , Coding Coach, Computer Science Researcher, System Programmer, Operating System Developer, Machine Learning Engineer / AI Engineer, Big Data Engineer / Cloud Engineer, Cyber security Analyst / Cryptography Engineer, Game Developer / Graphics Programmer.

This job mapping shows that Advanced Data Structures is a core competency across multiple high-demand domains in both industry and academia.

Text Books:

1. Sartaj Sahni, Dinesh P. Mehta; Handbook of Data Structures and Applications;

2nd edition, Chapman & Hall/CRC.

2. Fundamentals of Data Structures in C", E. Horwitz, S. Sahani, Anderson-Freed, Second Edition, Universities Press.

Reference Books:

1. T. Cormen, R.Rivest, C. Stein, C. Leiserson, "Introduction to Algorithms", Second Edition, PHI publication.

2. Peter Brass, Advanced Data Structures, First Edition, Cambridge University Press.

For MOOCs and other learning Resources

1. *www.nptelvideos.in,*
2. *www.geeksforgeeks.org*
3. *https://www.youtube.com/watch?v=244YpoG1pqA&list=PLrikLQMZHuSonRoDheibeb9ffd9phWIyu&index=5*
4. *https://classroom.volp.in/*
5. *https://www.coursera.org/*

FF No. : 654

CS2309: Theory of Computation

Credits: 2 **Teaching Scheme Theory: 2 Hours/Week**

Course Prerequisites: Introduction to discrete mathematics, proof techniques, basic familiarity with programing/computing.

Course Objectives:

1. Students will learn basic concepts such as alphabet, strings, Languages, Decision problems, etc and will be able to work with the abstract formal setup.
2. Students will be able to design deterministic/nondeterministic automata for regular languages, also he will be able to prove non regularity of languages through application of Pumping Lemma and Myhill-Nerode theorem.
3. Students will gain understanding of the role of non-determinism in Automata theory.
4. Students will be able to design Context free grammars, Push down automata for context Free Languages
5. Students will be able to design Turing Machines for various computational problems and see the equivalence of TM models with high level programming languages.
6. Students will be able to comprehend the meaning of undecidability in the context of Turing Machine Model and understand the inherent limits of computation.

Course Relevance:

This is a foundational course for Computer Science and Engineering. The central theme of the course is to study what makes certain computational problems very hard and the others easy? Is there some concrete theoretical evidence for the exhibited hardness of the problems? The course explores these questions, first by introducing students to the abstract notion of computation and models of computation. Starting from very simple model of state machines to finally cumulating into the Turing machine model (which is a foundation of modern-day computers), several models in between are studied. For every model, questions such as, which computational problems can be/cannot be solved in the model? How efficiently a problem can be solved in a particular model? various closure properties of the model are studied. Throughout the course emphasis is given to proving things with concrete mathematical arguments.

The course is very important for understanding the concept of computation in a more abstract set-up. Wherever one wants to formally talk about the underlying model, the restrictions imposed by the model, what is the power and limitation of the model, the principles learnt in this course are useful. Due to the abstract nature of the course, the principles learnt have wide applicability. The course is an essential prerequisite for several advanced courses such as Computational Complexity, Advanced Algorithms, Foundation of Logic, Quantum Computation, Parallel computation, Circuit Complexity etc. On a more applied side: The Automata theoretic models, concept of Context Free Grammar and

Pushdown Automata studied in the course are very important for Compiler design. The models discussed during the course have direct applications to several machine learning models, Natural Language processing, Artificial Intelligence, Functional Programming.

Once the student gains expertise in thinking abstractly about underlying models of computation it facilitates systematic study of any other domain (in computer science or otherwise) which demands logical thinking and abstraction.

This course is also relevant for students who want to pursue a research career in theory of computing, computational complexity theory, Natural Language Processing, advanced algorithmic research.

Syllabus
Theory

Section 1: Topics/Contents

Unit 1: Finite Automata (4 Hours)

Introduction to Automata, Computability and Complexity theory, Automaton as a model of computation, Central Concepts of Automata Theory: Alphabets, Strings, Languages. Decision Problems Vs Languages. Finite Automata, Structural Representations, Deterministic Finite Automata (DFA)-Formal Definition, Simplified notation: State transition graph, transition table, Language of DFA, construction of DFAs for Languages and proving correctness, Product construction, Nondeterministic finite Automata (NFA), NFA with epsilon transition, Language of NFA, Conversion of NFA with epsilon transitions to DFA, Applications and Limitation of Finite Automata.

Unit 2: Regular and Non-Regular Languages (6 Hours)

Regular expression (RE), Definition, Operators of regular expression and their precedence, Algebraic laws for Regular expressions, Kleene's Theorem: Equivalence Regular expressions and DFAs (without proof), Closure properties of Regular Languages (union, intersection, complementation, concatenation, Kleene closure), Decision properties of Regular Languages, Applications of Regular expressions. Myhill-Nerode theorem and applications: proving non-regularity, lower bound on number of states of DFA, State Minimization algorithm, Equivalence testing of DFAs. Non-Regular Languages, Revisiting Pigeon-Hole principle, Pumping Lemma for regular Languages.

Unit 3: Context Free Grammars (4 Hours)

Context Free Grammars: Definition, Examples, Derivation, Languages of CFG, Constructing CFG, correctness proof using induction. Closure properties of CFLs (Union, Concatenation, Kleene closure, reversal). Derivation trees, Ambiguity in CFGs, Removing ambiguity, Inherent ambiguity. Normal forms for CFGs: CNF and GNF (without proof). Decision Properties of CFLs (Emptiness, Finiteness and Membership). Applications of CFG.

Section 2: Topics/Contents

Unit 4: Push Down Automata (5 Hours)

Description and definition, Language of PDA, Acceptance by Final state, Acceptance by empty stack, Deterministic, Non-deterministic PDAs, CFG to PDA construction (with proof). Equivalence of PDA and CFG (without proof). Intersection of CFLs and Regular language. Pumping lemma for CFLs, non-Context Free Languages, Chomsky hierarchy**.**

Unit 5: Turing Machines (5 Hours)

Basic model, definition, and representation, Instantaneous Description, Language acceptance by TM. Robustness of Turing Machine model and equivalence with various variants: Two-way/One-way infinite tape TM, multi-tape TM, non-deterministic TM, Universal Turing Machines. TM as enumerator. Recursive and Recursively Enumerable languages and their closure properties.

Unit 6: Introduction to Undecidability (4 Hours)

Church-Turing Thesis and intuitive notion of Algorithm, Encoding for Turing machines and countability of set of all Turing machines. Existence of Turing unrecognizable languages via Cantor's diagonalization. Undecidability of Halting problem. Examples of undecidable problems: Post Correspondence Problem, Hilbert's 10th Problem, Tiling problem (without proof). Example of Turing unrecognizable language. Decision properties of R, RE languages.

Course Outcomes:

The student will be able to –

1. Infer the applicability of various automata theoretic models for recognizing formal languages.
2. Discriminate the expressive powers of various automata theoretic and formal language theoretic computational models.
3. Illustrate significance of non-determinism pertaining to expressive powers of various automata theoretic models.
4. Comprehend general purpose powers and computability issues related to state machines and grammars.
5. Explain the relevance of Church-Turing thesis, and the computational equivalence of Turing machine model with the general-purpose computers.
6. Grasp the theoretical limit of computation (independent of software or hardware used) via the concept of undecidability.

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
CO1	2	3	3	1							2	3		
CO2	2	3	2								2	3		
CO3	2	3	2	2							2	3		
CO4	2	3									2	3		
CO5	2	3	2								2	3		
CO6	2	3		1							2	3		
Average	2	3	2.25	1.33							2	3		

Future Courses Mapping:

Compiler design, Computational Complexity theory, Computability theory, Advanced Algorithms, Natural Language Processing, Artificial Intelligence

Job Mapping:

The principles learnt in the course have wide applicability, in domains like Compiler design, Programming languages design, Machine learning, Natural Language processing, etc. Any job that involves modeling and systematic study of some systems, background of Theory of Computation is useful. Understanding of the course content is helpful in developing a systematic and structured approach towards programming. The programming/algorithm design abilities lie at the heart of computer science and are useful for several job profiles in the computer industry. If a student wants to pursue higher education/ research in Computer Science, this course is essential.

Books and E-Resources

Text Books: ***(As per IEEE format)***

1.Hopcroft J, Motwani R, Ullman, Addison-Wesley, "Introduction to Automata Theory, Languages and Computation", Second Edition, ISBN 81-7808-347-7.
2. Michael Sipser, Course Technology, "Introduction to Theory of Computation", Third Edition,ISBN-10: 053494728X.
3.. "Discrete Mathematics and its applications" by Kenneth Rosen (William C Brown Publisher)

Reference Books: ***(As per IEEE format)***

1. J. Martin, "Introduction to Languages and the Theory of Computation", Third edition, Tata McGraw-Hill, ISBN 0-07-049939-x, 2003.
2. Daniel I. A. Cohen, "Introduction to Computer Theory", Wiley-Second Edition, ISBN-10 : 04711377

For MOOCs and other learning Resources

www.nptelvideos.in

FF No. : 654

CS2310: Operating System

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Lab: 2 Hours/Week

Course Prerequisites:

1. Computer Architecture
2. Computer organization
3. Data Structure

Course Objectives:

1. To learn functions of Operating System
2. To learn the importance of concurrency and how to implement concurrent abstractions correctly in an OS.
3. To learn OS scheduling policies and mechanisms.
4. To deal with deadlock.
5. To learn memory management schemes in various ways to improve performance, and how this impacts system complexity.
6. To learn design & develop the Operating system from a scratch.

Course Relevance:

Operating Systems serve as the fundamental backbone of all computing systems, from mobile phones and laptops to enterprise servers and embedded systems. Understanding how operating systems function is essential for every computer engineer, as it enables them to efficiently manage hardware resources, develop robust software, and ensure system stability and performance.

Syllabus

Theory

Section 1: Topics/Contents

Unit 1: Introduction to OS **(4 Hours)**

What is OS, Interaction of OS and hardware, Goals of OS, Basic functions of OS, OS Services, System Calls, and Types of system calls. Types of OS: Batch, Multiprogramming, Time sharing, Parallel, Distributed & Real-time OS.

Unit 2: Process Management **(6 Hours)**

Process Concept, Process States: 2, 5, 7 state models, Process Description, Process Control. **Threads:** Multithreading models, Thread implementations – user level and kernel level threads, Symmetric Multiprocessing. **Concurrency:** Issues with concurrency, Principles of **Concurrency Mutual**

Exclusion: H/W approaches, S/W approach, OS/Programming Language support: Semaphores, Mutex and Monitors. **Classical Problems of Synchronization:** Readers-Writers problem, Producer Consumer problem, Dining Philosopher problem

Unit 3: Process Scheduling (4 Hours)

Uniprocessor Scheduling: Scheduling Criteria, Types of Scheduling: Pre-emptive, Non pre-emptive, Long-term, Medium-term, Short-term. **Scheduling Algorithms:** First Come First Serve, Shortest Job First, Round Robin, Priority.

Section 2: Topics/Contents

Unit 4: Deadlocks (4 Hours)

Principles of deadlock, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Deadlock Recovery

Unit 5: Memory Management (6 Hours)

Fixed, Dynamic Partitioning, Buddy Systems, Fragmentation, Paging, Segmentation, Address translation. **Placement Strategies:** First Fit, Best Fit, Next Fit and Worst Fit. **Virtual Memory:** Concepts, Swapping, VM with Paging, Page Table Structure, Inverted Page Table, Translation Lookaside Buffer, Page Size, VM with Segmentation with combined paging and segmentation. **Page Replacement Policies:** FIFO, LRU, Optimal, Clock. Swapping issues: Thrashing

Unit 6: I/O and File Management (4 Hours)

I/O management: I/O Devices - Types, Characteristics of devices, OS design issues for I/O management, I/O Buffering. Disk Scheduling: FCFS, SCAN, C-SCAN, SSTF. List of Tutorials (13): File Management: Concepts, File Organization, File Directories, File Sharing. Record Blocking, Secondary Storage Management, Free Space management, Security.

Syllabus
Laboratory

List of Experiments

1. Execution of Basic & Advanced Linux Commands.
2. Write shell script covering – basic arithmetic, control structures, loops, execution of Linux command in shell, command line arguments, functions and arrays. (Title)
3. Write a C /C++ program to solve synchronization problem – Reader writer problem mutex & semaphore.
4. Write a C /C++ program to solve synchronization problem - Producer consumer problem mutex & semaphore.

5. Write a C /C++ program to solve synchronization problem - dinning philosopher problem using mutex & semaphore.
6. Implement CPU scheduling algorithms – FCFS , SJF , Round Robin and priority.
7. Implement Banker's algorithm
8. Implement deadlock detection algorithm
9. Implement placement strategies – First fit, Best Fit, Next Fit and Worst Fit.
10. Implement buddy system.
11. . Implement page replacement algorithm – FIFO, LRU and Optimal.
12. Write a C/ C++ program to convert logical/ virtual address into physical address using paging and segmentation.
13. . Implement disk scheduling algorithm – FCFS, SSTF, SCAN, C-SCAN.

Syllabus
Course Project

List of Course Projects

1. Design and implementation of a
   i. CPU/ Machine Simulation
   ii. Supervisor Call through interrupt Design multi programming operating system phase 1
2. Design and implementation of a Multiprogramming Operating System: Stage II
   i. Paging
   ii. Error Handling
   iii. Interrupt Generation and Servicing
   iv. Process Data Structure
3. Design and implementation of a Multiprogramming Operating System: Stage III
   i. I/O Channels & I/O buffering
   ii. Multiprogramming
   iii. I/O Spooling
4. Design multi programming operating system phase 1 with arithmetic & logical instructions
5. Design multi programming operating system phase 3 with swapping.

Course Outcomes

The student will be able to

1. Discuss the functions of a contemporary Operating system with respect to convenience, efficiency and the ability to evolve
2. Implement concurrent abstractions correctly in an OS to solve real world problems.
3. Use various CPU scheduling algorithms to construct solutions to real world problems.

4. Corelate the mechanisms related to deadlock handling in real life situations.
5. Distinguish memory management schemes & file management systems in various ways to improve performance, and analyze the impact of it on system complexity.
6. Design & develop the Operating system from a scratch

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
CO1	2	2											3	3
CO2	2	2	3	2			2	2					3	3
CO3	2	3	3	2			2	2					3	3
CO4	2	3		2									3	3
CO5	2	3		1									3	3
CO6	2	2	3	2		2	2	2	3	3	3		3	3
Average	**2**	**2.5**	**3.0**	**1.8**		**2.0**	**2.0**	**2.0**	**3.0**	**3.0**	**3.0**		**3.0**	**3.0**

Future Courses Mapping:
Advanced Operating System, Distributed Operating System, Parallel Computing.

Job Mapping:
System Administrator, System Analyst

Books and E-Resources

For Reference Print Book -

1. W. Stallings; 'Operating Systems: Internals and Design Principles'; 9th Edition; Pearson; 2017 .
2. A. Silberschatz, P. B. Galvin, G. Gagne ; 'Operating System Concepts'; 10th Edition ; Wiley; 2021.

For Reference Electronic Book –

R.H. Arpaci-Dusseau,A. C. Arpaci-Dusseau; 'Operating Systems: Three Easy Pieces'; 1st Edition; CreateSpace Independent Publishing Platform (Arpaci-Dusseau Books, LLC) , 2018. Accessed: June. 9, 2025. [Online]. Available: https://pages.cs.wisc.edu/~remzi/OSTEP/

For MOOCs and other learning Resources

R. Ahuja,; 'Hands-on Introduction to Linux Commands and Shell Scripting'; Coursera; https://www.coursera.org/learn/hands-on-introduction-to-linux-commands-and-shell-scripting/home; Accessed: June. 9, 2025

FF No. : 654

CS2311: Software Engineering

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Lab: 2 Hours/Week

Course Prerequisites: Basic Programming Skills, Object-Oriented Programming Concepts, Problem Solving and Logic Building

Course Objectives:

1. **To introduce foundational concepts and various software development lifecycle models**, including traditional and agile approaches such as Scrum, along with their roles and artifacts.
2. **To develop the ability to elicit, analyze, and document software requirements**, and to understand the structure of Software Requirement Specification (SRS) and use case modeling.
3. **To explain key principles of software design and differentiate between high-level and low-level designs**, focusing on modularity, reusability, and design patterns.
4. **To familiarize students with structural modeling using UML**, including class, object, and use case diagrams, and enable hands-on diagram creation using UML tools.
5. **To enable learners to represent dynamic behavior of software systems using UML behavioral models**, such as sequence, activity, and state diagrams, along with deployment and component diagrams.
6. **To impart practical knowledge of software project management**, including cost estimation, risk management, scheduling, quality assurance, and the use of tools like Trello and JIRA.

Course Relevance:

Software Engineering applies systematic principles and methodologies to design, develop, and maintain reliable, efficient, and scalable software systems, drawing on foundational knowledge from Computer Engineering. It encompasses the full software lifecycle, from requirements gathering and design to implementation, testing, and maintenance, ensuring quality and security within project constraints. The course emphasizes formal techniques, practical tools, and teamwork to build trustworthy software that meets real-world demands.

Syllabus

Theory

Unit 1: Introduction to Software Engineering (4 Hours)

Nature and scope of software engineering, Software characteristics and application domains, SDLC models: Waterfall, Spiral, Incremental, Rapid Application Development, Agile methodology overview: Scrum, roles, and artifacts.

Unit 2: Requirements Engineering (4 Hours)

Functional vs Non-functional requirements, Requirement elicitation techniques: interviews, observation, document analysis, Requirement analysis - MoSCoW technique, Writing Software Requirement Specification (SRS), Use case modeling.

Unit 3: Software Design and Design Principles (4 Hours)

High-level vs Software Design, Software design principles: modularity, cohesion, coupling, abstraction, reusability, High-level design, Low-level design, Characteristics of a good design, Design patterns,

Unit 4: Structural Modeling using UML (6 Hours)

Object-Oriented Modeling basics, Class diagram: classes, attributes, methods, associations, multiplicity, Object diagram, Use case diagram, Relationships: generalization, aggregation, composition, Hands-on drawing structural models using UML tools (e.g., Lucidchart & StarUML)

Unit 5: Behavioral Modeling using UML (6 Hours)

Sequence diagram: object interactions and message flow, Activity diagram: workflows and control flow, State diagram: system behavior modeling, Component and Deployment diagrams, Hands-on drawing behavioral models using UML tools (e.g., Lucidchart & StarUML).

Unit 6: Software Project Management (3 Hours)

Project planning and scheduling (Gantt, PERT), Software cost estimation (Function Point, Use Case Point), Risk management and mitigation, Quality assurance processes and metrics Tools: Trello, JIRA (introduction).

Syllabus
Laboratory

List of Experiments

List of experiments: (Any SIX)

1. To prepare a Statement Of Work (SOW) document, which addresses the vision, goals and objectives of the real-world problem.
2. To prepare a Software Requirement Specification (SRS) document, based on several types of system requirements, such as functional and non-functional requirements.
3. Create and manage use case diagrams for a real-world application (e.g., Online Shopping System) to capture functional requirements.
4. Design detailed class and object diagrams using UML tools (e.g., StarUML, Visual Paradigm) for a given problem statement such as a Student Management System.
5. Develop sequence diagrams illustrating object interactions and activity diagrams representing workflows for a banking transaction system.

6. Model the dynamic behavior of a system using state diagrams (e.g., ticket booking process in a cinema management system).
7. Prepare a software project plan with scheduling using Gantt charts and PERT charts for a sample project.
8. Perform software cost estimation exercises using Function Point Analysis and Use Case Points for a given software project.
9. Identify potential risks and prepare mitigation strategies for a hypothetical software development project.
10. Hands-on with Project Management Tools: Practice project tracking, task assignment, and progress monitoring using Trello and JIRA on a simulated project.

Syllabus
Course Project

List of Sample Course Projects

Project 1: Comparative Study of Agile vs. Traditional SDLC Models in Real-World Software Development

Analyze project outcomes, productivity, and quality metrics using case studies or surveys.

Project 2: Automated Requirements Elicitation and Analysis Using NLP Techniques

Develop or enhance a tool that applies Natural Language Processing for extracting and validating requirements from stakeholder documents.

Project 3: Design Patterns Effectiveness in Large-Scale Software Systems: A Quantitative Analysis

Study the impact of specific design patterns on code maintainability, performance, and scalability through quantitative metrics.

Project 4: UML-Based Model-Driven Development Framework for Rapid Prototyping

Build and evaluate a framework or tool that automates code generation from UML models to speed up prototyping.

Project 5: Risk Assessment and Mitigation Strategies in Agile Project Management

Develop a risk management framework tailored for agile projects and validate its effectiveness through case studies or simulations.

Project 6: Software Cost Estimation Accuracy Improvement Using Machine Learning Models

Use historical project data to train predictive models for software cost and effort estimation; compare

with traditional techniques.

Course Outcomes

The student will be able to –

1) **Analyze and compare different software development lifecycle models**, including agile methodologies like Scrum, to determine their suitability for various projects.
2) **Apply requirements elicitation techniques and prepare a detailed SRS**, identifying both functional and non-functional requirements effectively
3) **Evaluate software design principles and construct modular, cohesive, and reusable designs**, applying basic design patterns.
4) **Construct structural UML diagrams (class, object, use case)** using appropriate tools for visualizing the software architecture.
5) **Develop behavioral UML diagrams (sequence, activity, state, component, deployment)** to model system workflows and interactions.
6) **Plan and manage software projects using estimation techniques and project management tools**, addressing risks and ensuring quality assurance.

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
CO1	3	3	3	3	3	3	3	3	3	3	3	3	3	3
CO2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
CO3	2	2	2	2	2	2	2	2	2	2	2	2	2	2
CO4	1	1	1	1	1	1	1	1	1	1	1	1	1	1
CO5	1	1	1	1	1	1	1	1	1	1	1	1	1	1
CO6	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Average	2.5	2.5	2.66	1.83	2.33	2		2	2.25	2.5	2	2.83	2.83	

CO Attainment levels:

Weights for attainment levels: L1 - Easy - 0.75, L2 - Comfortable - 0.7, L3 – Medium – 0.65, L4 – Somewhat difficult – 0.6, L5 – Difficult – 0.55.

CO1- (L2), CO2 -(L3), CO3 - (L2), CO4 -(L3), CO5 -(L3), CO6 -(L4)

Future Courses Mapping:

The foundational knowledge and skills developed in *Software Engineering* directly support the learning outcomes of the following advanced and interdisciplinary courses:

Software Architecture and Design Patterns, Software Testing and Quality Assurance, Project Management in Software Development, DevOps and Agile Practices, Cloud Computing and Microservices Architecture, Secure Software Development, Human-Computer Interaction, Machine Learning Systems Engineering.

Job Mapping:

Students who complete a course in Software Engineering are well-prepared for careers where planning, designing, developing, and managing complex software systems is essential. Key job roles include:

Software Developer / Software Engineer, Requirements Analyst / Systems Analyst, Software Architect, DevOps Engineer, QA Engineer / Test Automation Engineer, Project Manager /, Scrum Master, Agile Coach, Product Manager (Technical), Software Consultant, Full Stack Developer, Application Support Engineer, R&D Engineer / Academic Researcher

Text Books:

1. ***Ian Sommerville**, Software Engineering, 10th Edition, Pearson Education.*
2. ***Roger S. Pressman & Bruce R. Maxim**, Software Engineering: A Practitioner's Approach, 8th Edition, McGraw-Hill.*
3. *Tom Pender, _UML Bible', John Wiley & Sons, 2003, ISBN - 0764526049*
4. *Kenneth Rubin, _Essential SCRUM: A Practical Guide To The Most Popular Agile Process', Addison-Wesley, 2012, ISBN-13: 978-0-13-704329-3.*

Reference Books:

1. *SorenLauesen, _Software Requirements: Styles and Techniques, Addison Wesley, 2002, ISBN 0201745704.*
2. *Pankaj Jalote, An Integrated Approach to Software Engineering, 3rd Edition, Springer.*
3. *S. L. Pfleeger & Joanne M. Atlee, Software Engineering: Theory and Practice, 4th Edition, Pearson.*
4. *Grady Booch, James Rambaugh, Ivar Jacobson, _Unified Modeling Language User's Guide', 2nd Edition, AddisonWesley 2005, ISBN – 0321267974.*

For MOOCs and other learning Resources

1. *NPTEL – Software Engineering- https://nptel.ac.in/courses/106105087*
2. *GeeksforGeeks – Software Engineering- https://www.geeksforgeeks.org/software-engineering/*
3. *Coursera – Software Process and Agile Practices*
4. *Classroom VolP*

FF No. : 654

CS2312: System and Application Development

Credits: 1 **Teaching Scheme Lab: 2 Hours/Week**

Course Prerequisites:

Basic knowledge of programming (any language)

Course Objectives:

1. Understand the basics of Dart and Flutter for cross-platform mobile development
2. Design responsive user interfaces with Flutter widgets
3. Implement navigation, state management, and API integration
4. Build, test, and deploy simple mobile applications

Course Relevance:

The course equips practical skills in cross-platform app development using Flutter, a framework in high demand across industries. It bridges theoretical programming concepts with real-world mobile application design, enhancing students' readiness for internships and projects.

Syllabus
Laboratory

List of Experiments

Lab 1: Introduction to Flutter & Dart

- Setting up Flutter SDK and IDE (Android Studio / VS Code)
- Running the first Flutter app: "Hello World"
- Basic Dart syntax: variables, functions, control structures

Lab 2: Flutter Project Structure & Widgets

- Exploring project folders and key files (pubspec.yaml, main.dart)
- Understanding Stateless and Stateful widgets
- Building a basic UI with common widgets (Text, Container, Row, Column)

Lab 3: Layouts and Styling

- Using layout widgets: Stack, Expanded, SizedBox
- Styling UI elements with themes, colors, and fonts
- Designing a basic login screen

Lab 4: User Input & Forms

- Working with TextField for user input
- Creating forms using Form and TextFormField
- Implementing basic input validation

Lab 5: Navigation and Routing

- Navigating between multiple screens using Navigator
- Passing data between screens
- Implementing a simple multi-page app

Lab 6: State Management Basics

- Introduction to state management: setState, Stateful widgets
- Handling user interactions and updating the UI
- Building a counter app with dynamic state

Lab 7: Lists and Dynamic Content

- Displaying dynamic lists using ListView and ListView.builder
- Creating simple item cards with Card widget

Lab 8: Fetching Data from APIs

- Using the http package to make network requests
- Fetching and displaying data from a REST API
- Basic JSON parsing in Flutter

Lab 9: Local Data Storage

- Introduction to local storage: shared_preferences
- Storing and retrieving simple data (e.g., login status, preferences)
- Basics of offline support

Lab 10: Final App Build and Deployment

- Consolidating concepts to build a mini project/app
- Building the APK for deployment
- Debugging and best practices recap

Course Outcomes

1. Understand the architecture and basic components of Flutter and Dart to build cross-platform apps.
2. Design and implement responsive UIs using Flutter widgets and layout structures.
3. Apply form validation and user interaction handling techniques to develop functional mobile apps.
4. Implement navigation and routing to create multi-screen apps with data transfer.
5. Integrate REST APIs and local storage (shared preferences) for data handling in apps.
6. Develop and deploy a basic Flutter-based mobile application meeting user requirements.

CO-PO Mapping

	Program Outcomes (PO)												PSO			
CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	2	2	3									2	3		2	
CO2	2	3	2									2	3		2	
CO3	3	3	3				2	2				2	3	2	2	2
CO4	3	3	3	3			2	2				2	3	2	2	2
CO5	3	3	2									2	3		2	
CO6	2	3	3									2	3		2	
Average	2.5	3.0	2.66	3.0			2.0	2.0				2.0	3.0	2.0	2.0	2.0

Future Courses Mapping:
Mobile Computing, Web Technologies

Job Mapping:
Mobile Application Developer, Front-end Developer (Mobile-focused), UI/UX Developer for mobile apps, Freelancer / Independent App Developer, Mobile Solutions Engineer

Books and E-Resources
For Reference Print Book -

1. Alessandro Biessek, "Flutter for Beginners, Packt Publishing, 2020 (Second Edition), ISBN: 978-1800565997
2. Frank Zammetti, "Practical Flutter: Improve Your Mobile Development with Google's Latest Open-Source SDK", Apress, 2019, ISBN: 978-1484249717

For MOOCs and other learning Resources

1. SWAYAM Plus
2. The Complete Flutter Development Bootcamp with Dart | Udemy
3. Flutter and Dart: Developing iOS, Android, and Mobile Apps | Coursera

FF No. : 654

Multidisciplinary Minor**

MM0702: Microcontroller

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Tutorial: 1 Hour/Week

Course Prerequisites: Computer organization & Architecture**,**Logic Design, Microprocessor,

Course Objectives:

1. To understand the architecture and features of PIC microcontrollers.
2. To learn memory organization and I/O programming using TRIS and LAT.
3. To implement timers, interrupts, and serial communication in PIC.
4. To study the architecture and register a set of ARM Cortex-M processors.
5. To understand data types, and addressing modes and instructions set in ARM.
6. To design real-time embedded applications using sensors and communication protocols.

Course Relevance: This is a basic course for Computer Science and Engineering. The course plays an essential role while modeling problems in computer science and engineering.Microcontroller course is highly relevant for Computer Engineering especially in today's world of smart devices, automation, and embedded intelligence.

Syllabus
Theory

Section 1: Topics/Contents

Unit 1: Introduction to PIC Microcontrollers (4 Hours)

Evolution and features of PIC microcontrollers Overview of PIC18 family,Architecture and pin configuration,Memory organization: RAM, ROM, EEPROM,Configuration registers and status flags

Unit 2: Instruction Set and I/O Programming (5 Hours)

Arithmetic, logic, and branch instructions,Stack operations and program counter,I/O port programming using TRIS and LAT registers, Bit manipulation and time delay generation,Assembly and embedded C programming basics.

Unit 3: Timers, Interrupts, and Serial Communication (5 Hours)

Timer modules (Timer0, Timer1, Timer2), Interrupt types and handling,UART programming and RS232 interfacing,Basics of SPI and I2C communication

Section 2: Topics/Contents

Unit 4: ARM Microcontroller (5 Hours)

Overview of ARM processor families (Cortex-M); RISC vs CISC;Block and Functional Diagram of ARM Controller, Pin Configuration of ARM Microcontroller. ARM pipeline and instruction set basics.

Unit 5: ARM Data Organization (5 Hours)

Instruction Set of ARM Microcontroller, Addressing Modes, Data Types, Register Set, Applications of ARM Microcontroller.

Unit 6: Advanced Application (4 Hours)

Mini real-time embedded systems: digital thermometer, sensor node, home automation controller.ARM in IoT: connectivity and cloud integration

Syllabus
Tutorials

List of Tutorials

1. PIC microcontroller architecture and pin configuration
2. Memory organization: RAM, ROM, EEPROM, configuration registers
3. PIC instruction set
4. Stack operations and program counter behavior in PIC
5. I/O programming using TRIS and LAT registers
6. Bit manipulation and software delay generation
7. Timer configuration in PIC
8. Interrupt handling and flag management in PIC
9. UART programming and RS232 serial communication
10. SPI and I2C communication protocols
11. ARM Cortex-M microcontroller architecture and pipeline
12. ARM instruction Set
13. ARM Addressing modes

14. Real-time embedded applications

Course Outcomes

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

1. Describe the evolution, architecture, features, and memory organization of PIC microcontrollers.
2. Elaborate assembly and embedded C programs for arithmetic, logic, stack, and I/O operations using TRIS and LAT registers in PIC.
3. Describe timers, interrupts, UART, SPI, and I²C for real-time applications in PIC microcontrollers.
4. Discuss the architecture, processor families, and pipeline features of ARM Cortex-M microcontrollers.
5. Interprete the instruction set, addressing modes, data types, and register usage in ARM Cortex-M microcontrollers for embedded system development.
6. Discuss real world application of ARM

CO-PO Mapping

	Program Outcomes (PO)												PSO			
CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	3	1		1								1	1		3	1
CO2	3	2				1			2		2	1	2		3	2
CO3	3	2	2									1	2		3	2
CO4	3	2	2	3								1	2		3	2
CO5	3	2	1						1		2	1	2	1	3	2
CO6	3	2	1	1		1	1	1			2	1	2	3	3	2
Average	3	1.8	1	0.8	0	0.3	0.1	0.1	0.5	0	1.3	1	1.8	0.7	3	1.8

Future Courses Mapping: Parallel Computing, Operating System, IoT, Embedded System

Job Mapping:

Embedded Systems Engineer, Firmware Developer, IoT Developer, Automation Engineer, Robotics Engineer

Textbook

1. “PIC Microcontroller and Embedded Systems: Using Assembly and C for PIC18”, 1st Edition, Muhammad Ali Mazidi, Rolin D. McKinlay & Danny Causey (Pearson/Prentice Hall, 2008)
2. “Design with PIC Microcontrollers”, 1st Edition, John B. Peatman (Prentice Hall, 1997–98)
3. “Embedded C Programming and the Microchip PIC”, 1st Edition, Richard H. Barnett, Sarah Cox & Larry O’Cull (Delmar Cengage Learning, November 2003)
4. “PIC Microcontroller Project Book”, 2nd Edition, John Iovine (McGraw-Hill, March 29 2004)
5. Intel 8-bil Microcontroller manual.

6. Ajay Deshmukh, "Microcontrollers — (Theory and application)" , 2004, TMH,ISBN 0-07-058595-4
7. J. Yiu, The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors, Newnes

FF No. : 654

Multidisciplinary Minor**

MM0703: Internet of Things

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Tutorial: 1 Hour/Week