Curriculum for S. Y. B. Tech. Sem-I

Computer Engineering (Software Engineering) (2023 NEP Pattern)

College Vision and Mission

Vision:

To be a globally acclaimed Institute in Technical Education and Research for holistic socio-economic development

Mission:

- To ensure that 100% of students are employable and employed in industry, higher studies, entrepreneurship, civil or defence services, government jobs, and other areas like sports and arts.

- To strengthen Academic Practices in Curriculum, Pedagogy, Assessment and Faculty Competence.
- To Promote Research Culture among Students and Faculty through Projects and Consultancy
- To make students Socially Responsible Citizens

Department Vision and Mission

Vision:

- Empowering Industry through Comprehensive Software Engineering Services to Achieve Excellence.

Mission:

- To produce industry-ready software engineering graduate with a blend of technical expertise and ethical responsibility

- To provide software engineering students with the utmost quality in developing technical, social, innovative, and entrepreneurial skills

Program outcomes (POs)-12 Graduate Attributes

- PO1: Engineering Knowledge Apply the knowledge of Mathematics, Science, Engineering fundamentals, and Engineering specialization to the solution of Complex Engineering Problems.

- PO2: Problem Analysis Identify, formulate, research literature, and analyze engineering problems to arrive at substantiated conclusions using first principles of Mathematics, Natural, and Engineering Sciences.

- PO3: Design/Development of solutions Design solutions for complex engineering problems and design system components, processes to meet the specifications with consideration for the public health and safety, and the cultural, societal, and environmental considerations.

- PO4: Conduct investigations of complex problems Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions..

- PO5: Modern tool usage Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

- PO6: The Engineer and Society Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

- PO7: Environment and Sustainability Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- PO8: Ethics Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

- PO9: Individual and team work Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.

- PO10: Communication Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations, and give and receive clear instructions. PO11: Project Management and Finance Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments.

- PO12: Life long learning Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Sr. No.	Course Code	Course Title	Course Type	Theory	Lab	Tut	ES E Th (W)	ESE TH (O)	CVV	CP	LAB	GD/PPT/HA	MSE (W)	MSE (O)	T1 (W)	T1 (O)	T2 (W)	T2 (O)	ESE	PRACT+ CVV	CR
01	SE2003	FUNDAMENTALS OF DATA STRUCTURES	TH	3	2	0				30	10									40+ 20	4
02	SE2004	DATABASE MANAGEMENT SYSTEMS	TH	2	2	0	40		20	30	10										3
03	SE2005	OPERATING SYSTEMS	TH	2	0	1	40		30			GD/PPT									3

				L	T	P												
											30							
04	SE2006	OBJECT ORIENTED PROGRAMMING	TH	1	2	0		40		30	10	GD/PPT 20						2
05	SEM001	MDM:Agile Principles and methodology		2		1						HA 30			35	35		3
06	HS2002	HS2002: CTC (Satkam-I)		2		-		50					50					2
7	HS2001	HS2001: RAD		1		-											100	1
8	SE2001	SE2001 Design Thinking				1											100	1
9	SE2002	ENGINEERING DESIGN AND INNOVATION-III			2								30				70	2
	Total																	21

Nomenclature for Teaching and Examination Assessment Scheme

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	Test –1 Written/online	T1
15	Assessment	Test –2 Written/online	T2
16	Assessment	Mid Semester Examination (W/O) Written/online	MSE
17	Assessment	End Semester Examination	ESE
18	Assessment	Written Examination	W
19	Assessment	Multiple Choice Questions	MCQ
20	Assessment	Comprehensive Viva Voce	CVV

SE2003 Fundamentals of Data Structures

Teaching Scheme: Theory: 3 Hours/Week, Lab: 2 Hours/WeeK	Credits: 4	Examination Scheme: Course Project-30 LAB-10a PRAC 40 Comprehensive Viva Voce -20

Prerequisites

- Knowledge of programming fundamentals using C or Python

- Understanding of control structures, functions, and arrays

- Basic concepts of memory management and file handling

Course Objectives

1. To introduce students to linear and nonlinear data structures and their applications

2. To develop problem-solving skills by applying data structures to real-life scenarios

3. To understand and analyze the complexity of algorithms

4. To implement and apply various sorting and searching techniques

5. To develop efficient programs using stacks, queues, linked lists, trees, and graphs

6. To expose students to memory management, recursion, and performance trade-offs

SECTION I

Unit I: Introduction to Data Structures and Complexity (6 Hours)

Abstract Data Types, need of data structures, classification of data structures, complexity analysis of algorithms using Big-O, Big-Ω and Big-Θ notations, time-space trade-off, recursion basics and applications

Unit II: Arrays and Searching & Sorting Techniques (6 Hours)

1D and 2D arrays, sparse matrix representation, linear search, binary search, bubble sort, selection sort, insertion sort, merge sort, quick sort, time complexity of all techniques

Unit III: Stacks and Queues (6 Hours)

Stack operations and applications like expression evaluation, infix to postfix conversion, recursion using stack; queue operations, circular queue, priority queue, deque, applications of queues in real-world problems

SECTION II

Unit IV: Linked Lists (6 Hours)

Singly linked list, doubly linked list, circular linked list, operations (insert, delete, traverse, search), applications such as polynomial operations, memory management using dynamic allocation

Unit V: Trees (6 Hours)

Tree terminologies, binary trees, binary search trees (BST), tree traversals (inorder, preorder, postorder), AVL trees, expression trees, heap trees, applications in decision making and file systems

Unit VI: Graphs and Hashing (6 Hours)

Graph representation using adjacency list and matrix, BFS, DFS, applications of graphs in networking and social media, hash tables, collision resolution techniques: chaining and open addressing

Lab Assignments

1. Implement recursive functions and analyze complexity

2. Implement array operations including sparse matrix

3. Implement linear and binary search techniques

4. Implement and compare sorting algorithms (bubble, insertion, selection, merge, quick)

5. Implement stack operations and applications (expression evaluation)

6. Implement queue, circular queue, and dequeue

7. Implement singly, doubly, and circular linked list

8. Implement polynomial addition using linked list

9. Implement binary tree creation and traversals

10. Implement binary search tree and AVL tree operations

11. Implement BFS and DFS for graph traversal

12. Implement hashing with collision resolution

Course Outcomes

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

1. Analyze and compute time and space complexity of algorithms

2. Apply searching and sorting algorithms to solve computational problems

3. Implement and apply stack and queue operations in applications

4. Develop and use various types of linked lists to manage dynamic data

5. Apply tree data structures for hierarchical data representation and operations

6. Demonstrate understanding of graph and hashing techniques and their applications

Textbooks & References

Textbooks (Print - Available in Pune)

1. E. Horowitz, S. Sahni, D. Mehta; *Fundamentals of Data Structures in C*; 2nd Edition; Universities Press; 2008

2. Y. Langsam, M. Augenstein, A. Tannenbaum; *Data Structures Using C and C++*; 2nd Edition; Pearson Education; 2006

Reference E-Books (Open Source / Freely Available)

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein; *Introduction to Algorithms*; 3rd Edition; MIT Press; https://mitpress.mit.edu/9780262033848/

2. Brad Miller, David Ranum; *Problem Solving with Algorithms and Data Structures using Python*; Open Book Project; Accessed: May 2025; https://runestone.academy/ns/books/published/pythonds/index.html

MOOCs and Learning Resources

1. Prof. Naveen Garg; *Data Structures and Algorithms*; NPTEL – IIT Delhi; https://nptel.ac.in/courses/106/102/106102064

2. UC San Diego, National Research University Higher School of Economics; *Data Structures*; Coursera; https://www.coursera.org/learn/data-structures

CO-PO Mapping Matrix: Fundamentals of Data Structures

CO \ PO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12
CO1	3	3										
CO2	3	2	2		1							
CO3	3	2	2		2							
CO4	3	2	2		2							
CO5	3	3	2		2							
CO6	3	3	2		3							

Justification for Mapping for all COs

CO1: Analyze and compute time and space complexity of algorithms

- **PO1 (Engineering Knowledge)**: Core computational understanding

- **PO2 (Problem Analysis)**: Analyzing efficiency of algorithms

CO2: Apply searching and sorting algorithms to solve computational problems

- **PO1**: Implementation of classic algorithmic solutions

- **PO2**: Choosing correct approach

- **PO3**: Algorithm design decisions

- **PO5**: Use of IDEs and visualization tools

CO3: Implement and apply stack and queue operations in applications

- **PO1**: Data management strategies

- **PO2**: Simulating real-life stack/queue behavior

- **PO3**: Programming logic

- **PO5**: Debugging, analysis tools

CO4: Develop and use various types of linked lists

- **PO1, PO2, PO3**: Linked structure implementation

- **PO5**: Usage of pointers and dynamic memory

CO5: Apply tree data structures

- **PO1, PO2**: Problem representation using trees

- **PO3**: Designing traversals

- **PO5**: Use of debuggers to trace tree logic

CO6: Demonstrate understanding of graphs and hashing

- **PO1, PO2**: Network and associative data models

- **PO3**: Efficient implementation

- **PO5**: Graph plotting tools or simulations

SE2004 **Database Management Systems**		
Teaching Scheme: Theory: 2 Hours/Week, Lab: 2 Hours/WeeK	**Credits**: 3	Examination Scheme: ESETH(Written):40 Course Project-30 LAB-10/ Comprehensive Viva Voce -20

Prerequisites

- Basic programming knowledge (C/Python)
- Understanding of data structures and algorithms
- Logical reasoning and problem-solving skills
- Basic knowledge of file systems

Course Objectives

1. To introduce database concepts and data modeling using Entity Relationship models
2. To develop skills for formulating queries using SQL for data manipulation
3. To understand relational database design using normalization techniques
4. To explore transaction management, concurrency control, and recovery in database systems
5. To provide hands-on experience using open-source RDBMS tools
6. To prepare students to implement real-world database applications

Unit I: Introduction to Databases and Data Models (6 Hours)

Database concepts, characteristics of database systems, three-schema architecture, data independence, database users and DBA roles, data models: hierarchical, network, relational, object-oriented, entity types, entity sets, attributes, types of attributes, keys, relationship types, ER diagram, ER to relational mapping, Real-world case study: Railway Reservation

Unit II: Relational Model and SQL (6 Hours)

Relational model concepts, domains, attributes, tuples, relational algebra operations: select, project, union, set difference, Cartesian product, rename, joins, SQL: DDL, DML, DCL commands, integrity constraints, built-in functions, subqueries, views, joins, set operations, indexing, Modern SQL practices: JSON support in PostgreSQL

Unit III: Relational Database Design and Normalization (6 Hours)

DBMS Utilities for Data Warehousing :schema design, problems in database design, functional dependencies, closure of attributes, canonical cover, decomposition, lossless join and dependency preservation, normal forms: 1NF, 2NF, 3NF, Data Transformation and Database Privacy and Access Control (Basic), Case Study: Transaction management in E-Commerce portals ,

Unit IV: Transactions, Concurrency Control, and Recovery (6 Hours)

Concept of transactions, properties of transactions (ACID), schedule and serializability, concurrency control techniques: locking protocols, timestamp ordering, deadlock prevention and detection, crash recovery, log-based recovery, shadow paging, checkpointing. Hands-on introduction to MongoDB, CAP theorem, use cases for NoSQL, Case Study: Firebase or Amazon DynamoDB for mobile apps

Lab Assignments

1. Installation and configuration of MySQL/PostgreSQL

2. Creating a Data Model : Conceptual, Logical and Physical data model .Prepare the exhaustive glossary list ,Identify entities and attributes from the glossary list, Prepare Conceptual Data Model ,Transform into Logical Data Model and Physicalize the Data Model into a Database Structure.

3. Population of Database :Creating databases and tables using DDL commands

4. Extraction of Database : Performing DML operations: INSERT, UPDATE, DELETE Querying data using SELECT with WHERE, ORDER BY, and aggregate functions Implementing inner, outer, and self joins Working with nested queries and subqueries Creating views and indexing table

5. Implementing stored procedures and triggers

6. Mini-project: Develop a CRUD-based application using a real-world scenario (Database :Restaurant, Bank, Library, Air Line reservation etc)

    **Use Agile Methodologies for each of the Assignments**

Course Outcomes

Upon successful completion, the student will be able to:

1. Explain fundamental database concepts and data modeling

2. Apply relational algebra and SQL to solve query-based problems

3. Design normalized relational schemas for effective data storage

4. Implement transaction and concurrency control mechanisms

5. Use open-source RDBMS tools to manage databases

6. Develop small-scale database applications with practical interfaces

Textbooks & References

Textbooks:

1. A. Silberschatz, H. Korth, S. Sudarshan; *Database System Concepts*; 7th Ed.; McGraw-Hill Education; 2020

2. R. Elmasri, S. Navathe; *Fundamentals of Database Systems*; 7th Ed.; Pearson; 2017

References:

1. D. Kroenke, D. Auer; Database Processing: Fundamentals, Design and Implementation; Pearson

2. Dr. S. Thakur; Database Systems – Concepts and Case Studies; Oxford Univ. Press; Available via NPTEL

Open-Source E-Resources:

1. R. Elmasri, S. Navathe; *Fundamentals of Database Systems*; Pearson; https://archive.org/details/fundamentals-of-db-elmasri-navathe

2. NPTEL Course: *Database Management Systems*; Prof. P. Dasgupta (IIT Kharagpur); https://nptel.ac.in/courses/106105175

3. Stanford Online; *Databases* by J. Widom; https://online.stanford.edu/courses/soe-ycs-databases

4. Coursera – *Databases Specialization* by University of Michigan; https://www.coursera.org/specializations/databases

5. J. Widom; *Databases: Relational Databases and SQL*; Stanford Online via edX; https://online.stanford.edu/courses/soe-ycs-databases; Accessed: May. 20, 2025

6. J. Ullman, J. Widom; *Databases*; Coursera; https://www.coursera.org/specializations/databases; Accessed: May. 20, 2025

7. Free SQL tutorials and playground: https://sqlzoo.net, https://sqliteonline.com

8. MongoDB University; *MongoDB for Developers*; https://university.mongodb.com

CO-PO Mapping Matrix: Fundamentals of Database Management Systems

CO\PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2									
CO2	3	3	2		2						
CO3	3	3	3	2	2						
CO4	3	2	2	3	2						
CO5	3	2	2	2	3				1	2	1
CO6	3	2	3	2	3				3	2	2

Justification for Mapping

CO1: Understanding fundamental database concepts and ER modeling

- **PO1**: Strong understanding of database architecture and components

- **PO2**: Ability to analyze data modeling problems

CO2: Writing SQL queries and relational algebra operations

- **PO1**: Requires sound conceptual and theoretical understanding

- **PO2**: Identifies complex query issues

- **PO3**: Designing queries to extract meaningful data

- **PO5**: Using RDBMS tools like MySQL/PostgreSQL

CO3: Designing normalized databases

- **PO1**: Deep conceptual understanding of dependencies

- **PO2**: Ability to analyze anomalies and redundancies

- **PO3**: Effective design using normalization

- **PO4**: Applying functional dependencies and logical inference

CO4: Understanding transaction, concurrency, and recovery

- **PO1**: Solid grasp of theoretical constructs

- **PO2**: Analyzing transaction anomalies and deadlocks

- **PO3**: Implementing ACID properties

- **PO4**: Investigating concurrency issues and system crashes

CO5: Applying open-source tools for managing databases

- **PO5**: Strong tool usage, integration, and problem-solving skills

- **PO9**: Working in teams for mini-projects

- **PO10**: Writing documentation

- **PO11**; Managing real-world use cases

CO6: Developing real-world DB applications

- **PO1-PO3**: Knowledge from schema to queries

- **PO9**: Project collaboration, communication, ethics, and management

- **PO10**: Communication, ethics, and management

- **PO11**: Management

SE2005 Operating Systems Syllabus		
Teaching Scheme: Theory: 2 Hours/Week, Tut: 1 Hours/WeeK	**Credits**: 3	Examination Scheme: ESE-TH(Written):40 Comprehensive Viva Voce -30 Group Discussion/Presentation-30

Prerequisite:

Data structures(stack, queue, linked list, tree, graph), hashing, File structures, Any structured Programming Language.

Course Objectives

- o **To provide knowledge about the services rendered by operating systems**

- o **To provide a detailed discussion of the various memory management techniques**

- o **To discuss the various file-system design and implementation issues**

- o **To discuss how the protection domains help to achieve security in a system**

Unit 1: Introduction

Operating Systems Functionalities - Formal Definition - Evolution – Types of operating system, Services, Operating system Design and Implementation, Operating System Structure.

Unit 2: Process Management

Process concept - Process control block, Process Hierarchy, Threads – Single Thread and Multi Thread Model, IPC models: shared memory and message passing. CPU Scheduling algorithms, Multiprocessor Scheduling, Process Synchronization - Peterson's Solution, Process Synchronization - Semaphores, Critical Regions, Monitors - Deadlock prevention- Deadlock avoidance and Deadlock Detection and Recovery - Bankers Algorithm.

Unit 3: Memory Management

Overview of Swapping - Multiple Partitions – Paging, Page table, Segmentation, Demand paging- Fragmentation & Compaction- Page replacement algorithms, Memory allocation algorithms: first fit, Best fit, worst fit.

Unit 4 : File System

Access Methods, Contiguous-Sequential and Indexed Allocation, File system interface - File System implementation, Secondary Storage Structure.

List of Experiments

1. Study of Basic commands of Linux/UNIX.

2. Study of Advance commands and filters of Linux/UNIX.

3. Write a shell script to generate marksheet of a student. Take 3 subjects, calculate and display total

marks, percentage and Class obtained by the student.

4. Write a shell script to find factorial of given number n.

5. Write a shell script which will accept a number b and display first n prime numbers as output.

6. Write a shell script which will generate first n fibonnacci numbers like: 1, 1, 2, 3, 5, 13,…

7. Write a menu driven shell script which will print the following menu and execute the given task.

8. MENU

9. Display calendar of current month

10. Display today's date and time

11. Display usernames those are currently logged in the system

12. Display your name at given x, y position

13. Display your terminal number

14. Exit

15. Write a shell script to read n numbers as command arguments and sort them in descending order.

16. Write a shell script to display all executable files, directories and zero sized files from current directory.

17. Write a shell script to check entered string is palindrome or not.

18. Shell programming using filters (including grep, egrep, fgrep)

19. Study of Unix Shell and Environment Variables.

20. Write a shell script to validate the entered date. (eg. Date format is : dd-mm-yyyy).

21. Write an awk program using function, which convert each word in a given text into capital.

22. Write a program for process creation using C. (Use of gcc compiler).

Assignment – It should consist of theoretical and analytical questions covering the whole syllabus.

Course Outcomes

- o **Ability to comprehend the techniques used to implement the process manager**

- o **Ability to comprehend virtual memory abstractions in operating systems**

- o **Ability to design and develop file system interfaces, etc.**

- o **Technical knowhow of the working principle of various types of operating systems**

Textbooks

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, "Operating System Concepts", 9th Edition, John Wiley & Sons Inc. 2012.

2. Andrew S Tanenbaum, "Modern Operating Systems", 4th Edition, Prentice Hall. 2014

3. William Stallings, "Operating System: Internals and Design Principles", 9th Edition, Pearson, 2018.

Reference Books

1. Harvey M. Deitel, Paul J. Deitel, David R. Choffnes, "Operating System", 3rd Edition, Pearson, 2013.

2. D M Dhamdhere, "System Programming and Operating Systems", 2nd Edition, Tata McGraw Hill, 2009.

3. Gary Nutt, " Operating Systems: A Modern Perspective", 2nd Edition, Addison Wesley, 2001.

4. Achyut S Godbole, "Operating Systems", 3rd Edition, Tata McGraw Hill, 2010.

SE2006 Object Oriented Programming		
Teaching Scheme: Theory: 1 Hours/Week, lab: 2 Hours/Week	**Credits**: 2	Examination Scheme: ESE-TH(Written):40 Course project-30 LAB-10 Group Discussion/Presentation-20

Prerequisites

● Basic Programming

Course Objectives

● To introduce the fundamentals of object-oriented programming and familiarize students with the basic syntax and features of C++.

● To develop the ability to implement modular, maintainable, and reusable code using classes, objects, constructors, destructors, and operator overloading.

● To explain and apply core OOP concepts such as inheritance, polymorphism, function overloading, and virtual functions for building hierarchical and dynamic programs.

● To enable students to perform file operations (text and binary) and handle runtime errors using exception handling mechanisms in C++.

● To introduce templates for generic programming and promote writing efficient and type-independent code.

Course Relevance

This course equips CSE-IoTCSBT students with essential object-oriented programming skills using C++, which are crucial for developing efficient, modular, and reusable software for IoT systems. Concepts like classes, inheritance, and polymorphism help in designing hardware-abstracted and scalable code. File handling and exception management enable robust data processing and error control in real-time IoT applications. Templates promote reusable code for drivers and protocols. Overall, it builds a strong foundation for embedded programming and advanced IoT development.

SECTION I – Foundations of OOP

Unit 1: Introduction to OOP

Fundamentals of OOPS, Introduction to Programming and C++, How C++ differs from C ,Variables, Data Types, and Operators , Control Structures, Loops and Iteration, Functions and Modular Programming, Basics of Console Input and Output Class, Dynamic Memory Allocation Overview of OOPs Principles , Introduction to classes & objects , Creation & destruction of objects , Data Members , Member Functions , Access Specifier, this Pointer , Constructor &Destructor , Static class member ,Friend class and functions , Function Overloading, Operator Overloading Namespace.

Unit 2: OOP Principles

Introduction to inheritance, Base and Derived class Constructors , Types of Inheritance, Down casting and up casting, Function overriding, Virtual functions, Polymorphism, Pure virtual functions, Virtual Base Class, C++ Class Hierarchy , File Stream ,Text File Handling , Binary File Handling , Error handling during file operations , Overloading << and >> operators, Introduction to Exception, Benefits of Exception handling, Try and catch block, Throw

statement, Pre-defined exceptions in C++, Writing custom Exception class, Stack Unwinding, Function Templates , Class Templates

Lab Assignments

1. Basics of C++
    a) Write a C++ program to calculate the area of a rectangle given its length and width.
    b) Develop a C++ program that converts temperature from Celsius to Fahrenheit using the formula: Fahrenheit = (Celsius * 9/5) + 32.
    c) Create a program that takes a user's age as input and determines if they are eligible to vote or not.
    d) Implement a C++ program that generates the Fibonacci sequence up to a given number 'n' using loops.

2. Functions and Modular Programming
    a) Design a program that calculates the factorial of a given positive integer using a recursive function.
    b) Develop a modular program that checks whether a given number is prime or not, utilizing a function for prime number testing.

3. Object-Oriented Programming (OOP)
    a) Create a C++ class named Rectangle that has attributes for length and width. Implement methods to calculate the area and perimeter of the rectangle.

    b) Design a program using OOP concepts to simulate a basic banking system. Implement classes for customers and accounts, allowing for deposits and withdrawals.

    c) Write a program to use static data members and member functions.

    d) Use the this pointer to resolve naming conflicts.

    e) Create a program to overload arithmetic operators using friend functions.

    f) Implement overloading of comparison operators (==, <, etc.).

4. Inheritance and Polymorphism

    a) Build a hierarchy of classes representing different shapes (e.g., Circle, Triangle, Rectangle) with a common base class. Implement a virtual function for calculating the area of each shape.

    b) Extend the banking system to include different types of accounts (Savings, Checking) that inherit from a common Account class. Implement polymorphic behavior for interest calculations.

5. Write a program using try, catch, and throw for arithmetic and input errors.
6. Write a function template for finding the maximum of two values.

Course Outcomes

Students will be able to:

- CO1: Demonstrate understanding of object-oriented programming fundamentals and implement modular programs in C++ using classes, objects, constructors, destructors, and operator overloading.
- CO2: Develop C++ applications using advanced OOP features like inheritance, polymorphism, file handling, exception handling, and templates to ensure robustness and reusability.

Textbooks & References

1. *Behrouz A. Forouzan, Richard F. Gilberg, "COMPUTER SCIENCE – A Structred Programming approach using C", Indian Edition, Thomson, 3rd edition*
2. *BjarneStroustrup, ― The C++ Programming language‖, Third edition, Pearson Education. ISBN 9780201889543*
3. *Kernighan, Ritchie, "The C Programming Language", Prentice Hall of India*
4. *Robert Lafore, ―Object-Oriented Programming in C++, fourth edition, Sams Publishing, ISBN:0672323087 (ISBN 13: 9780672323089)*
5. *Herbert Schildt, ―C++ The complete reference‖, Eighth Edition, McGraw Hill Professional, 2011, ISBN:978-00-72226805*
6. *E. Balagurusamy-- Object-oriented programming with C++, fourth edition, Mc Hill Professional,2008, ISBN 978-0-07-066907-9*

CO-PO Mapping Matrix: Object Oriented Programming

CO\PO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO1 0	PO1 1	PO1 2
CO1	3	2	3	1	1							
CO2	3	3	3	2	2							

Scale:

- 3 – Strongly Mapped
- 2 – Moderately Mapped
- 1 – Slightly Mapped
- Blank – Not Mapped

☐ **Justification for Mapping**

CO1: Understanding and Implementation of OOP Fundamentals

- **PO1 (Engineering Knowledge):** Strong correlation (3) because CO1 requires applying core engineering principles of programming and software design using C++ and OOP concepts.
- **PO2 (Problem Analysis):** Moderate correlation (2) as students analyze programming problems and devise modular object-oriented solutions.
- **PO3 (Design/Development):** Strong correlation (3) since CO1 involves designing classes and objects, constructors, destructors, and operator overloading to develop reusable software components.
- **PO4 (Investigation of Problems):** Slight correlation (1) because debugging and basic problem investigation are part of learning OOP fundamentals.

- **PO5 (Modern Tool Usage):** Slight correlation (1) as students use IDEs and debuggers for C++ programming, but advanced tools are less emphasized here.

CO2: Advanced OOP Features and Robust Application Development

- **PO1 (Engineering Knowledge):** Strong correlation (3) since CO2 extends knowledge to advanced concepts like inheritance, polymorphism, file handling, and templates in C++.
- **PO2 (Problem Analysis):** Strong correlation (3) because CO2 involves analyzing complex software problems and designing polymorphic and template-based solutions.
- **PO3 (Design/Development):** Strong correlation (3) as CO2 requires designing robust and reusable applications with advanced OOP principles and exception handling.
- **PO4 (Investigation of Problems):** Moderate correlation (2) since handling exceptions and file errors requires deeper investigation and understanding of runtime problems.
- **PO5 (Modern Tool Usage):** Moderate correlation (2) because students use file handling libraries, templates, and exception handling features which are part of modern C++ tools and standards.

SEM001 **Agile Principles and Methodology**

Teaching Scheme: Theory: 2 Hours/Week, Tut: 1 Hours/Week	**Credits**: 3	Examination Scheme: Home assignment -30 T1(Online)-35 T2(Online)-35

Prerequisites

Basics of Software Engineering

Course Objectives

● To introduce characteristics of an agile development process.
● To understand agile software development process models and plan driven process models.
● To understand software project characteristics that would be suitable for an agile process.
● To impart and Identify software project characteristics that would not be suitable for an agile process.

Unit I: FUNDAMENTALS OF AGILE (6 Hours)

The Genesis of Agile, Introduction and background, Agile Manifesto and Principles, Overview of Scrum, Extreme Programming, Feature Driven development, Lean Software Development, Agile project management, Design and development practices in Agile projects, Test Driven Development, Continuous Integration, Refactoring, Pair Programming, Simple Design, User Stories, Agile Testing, Agile Tools

Unit II: AGILE SCRUM FRAMEWORK (6 Hours)

Introduction to Scrum, Project phases, Agile Estimation, Planning Game, Product backlog, Sprint backlog, Iteration planning, User story definition, Characteristics and content of user stories, Acceptance tests and Verifying stories, Project velocity, Burn down chart, Sprint planning and retrospective, Daily scrum, Scrum roles Product Owner, Scrum Master, Scrum Team, Scrum case study, Tools for Agile project management.

Unit III: AGILE TESTING (6 Hours)

The Agile lifecycle and its impact on testing, Test-Driven Development (TDD), Unit framework and tools for TDD, Testing user stories - acceptance tests and scenarios, Planning and managing testing cycle, Exploratory testing, Risk based testing, Regression tests, Test Automation, Tools to support the Agile tester.

Unit IV: AGILE SOFTWARE DESIGN AND DEVELOPMENT (6 Hours)

Agile design practices, Role of design Principles including Single Responsibility Principle, Open Closed Principle, Liskov Substitution Principle, Interface Segregation Principles, Dependency Inversion Principle in Agile Design, Need and significance of Refactoring, Refactoring Techniques, Continuous Integration, Automated build tools.

Tutorials / Assignments

1. Study & Evaluate the historical and contextual factors that contributed to the rise of Agile software development methodologies.

2. Create & Evaluate the definition of user stories and explain their significance within the context of Agile software development methodologies.

3. Create a Scrum Team which defines & identifies the key roles within a Scrum Team based on Real Time Scenario.

4. Apply Design principle and Refactoring to achieve agility based on the above Real Time Scenario.

5. Implement Jira, a project management tool that helps teams plan, track, and manage work for Software Application.

6. Case study (Agile in Software Development vs. Other Industries (Manufacturing, Education, Healthcare)

7. Study and implement automated build tool Docker/ Kubernetes/ OpenShift for an Application.

8. Use Continuous Integration tool such as Jenkins for Software Implementation.

9. Perform Testing activities within an agile project using Jira.

Course Outcomes

At the end of the course the student should be able to:

CO1: Explain fundamentals of Agile methodology.

 CO2: Explain agile principles.

CO3: Apply Scrum principles.

CO4: Apply practices of XP and Incremental design.

Textbooks & References

Textbooks:

  1. Agile Development with Scrum, Ken Schwaber & Mike Beedle, Prentice Hall

  2. Integrating Agile Development in the Real World, Peter Schuh, Charles River Media
Reference Books:

  1. Agile Software Development – The Cooperative Game (2nd Edition), Alistair Cockburn.

  2. Succeeding With Agile, Software Development Using Scrum, Mike Cohn, Addison Wesley

ENGINEERING DESIGN AND INNOVATION-III

Teaching Scheme: PR: 4 Hours/Week	Credit: 2	Examination Scheme: MSE(Online)-30 ESE-70

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 Outcomes:

After completion of the course, students will be able to

1. Integrate knowledge across multiple domains to address complex problems.
2. Take ownership of their learning process through self-directed and peer-supported efforts.
3. Apply concepts learned in various courses to real-world scenarios, encouraging deeper understanding and sustained engagement.
4. Build confidence in managing projects from conception to implementation, fostering a habit of continuous learning and innovation.

Course Contents

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.

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.

Text books :

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 :

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