MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL
(Formerly West Bengal University of Technology)
Syllabus of B. Sc. in IT
(Effective from 2023-24 Academic Sessions)
SEMESTER: IV

Paper: Principles of Software Engineering

Code: BSCITM401
Contacts Hours/Week: 4L
Credits: 4

COURSE OBJECTIVE

1. To provide students with a fundamental understanding of software engineering principles and methodologies.
2. To develop the ability to design and manage software development projects using various lifecycle models.
3. To enhance skills in requirement analysis, software testing, and quality assurance to deliver reliable software solutions.

COURSE OUTCOMES

CO No.	Course Outcome Description	Cognitive Level (Bloom's)
CO1	Explain the fundamental principles, process models, and methodologies of software engineering to address software development challenges.	**Remembering, Understanding**
CO2	Apply software lifecycle models, requirement analysis techniques, and project management tools to plan, design, and manage software projects.	**Applying**
CO3	Analyze and evaluate software systems through structured techniques like DFDs, COCOMO, and testing strategies to ensure functionality, maintainability, and cost-efficiency.	**Analyzing, Evaluating**
CO4	Design and develop effective test cases, validation techniques, and project plans to ensure reliable and efficient software solutions.	**Creating, Applying**
CO5	Assess and ensure software quality through quality assurance processes, reliability metrics, and adherence to industry standards like IEEE for statistical quality control.	**Evaluating**

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

(Formerly West Bengal University of Technology)

Syllabus of B. Sc. in IT

(Effective from 2023-24 Academic Sessions)

CO-Module Mapping Table

CO No.	Mapped Modules
CO1	M1, M2
CO2	M2, M3, M4
CO3	M3, M4, M5
CO4	M4, M5
CO5	M5, M6

Module-Wise Syllabus

Module No.	Name of the Topic	Hours	Marks
M1	**Introduction**	6	10
	Introduction to Software Engineering: The evolving role of software, changing nature of software, software myths. A Generic view of process: Software engineering- a layered technology, a process framework, the capability maturity model integration (CMMI), process patterns, process assessment.		
M2	**Software Life Cycles**	6	15
	Overview of classical models: Waterfall, Iterative Waterfall, Spiral, Prototype, and Evolutionary models. Comparative studies and their applicability to different scenarios.		
M3	**Software Requirement and Specification Analysis**	10	20
	Requirements principles, specification principles, design analysis using DFDs (physical and logical), decision tables, and trees. Concepts of module cohesion and coupling. Overview of software cost estimation using the COCOMO model.		
M4	**Software Planning and Project Management**	10	20
	Project planning fundamentals: scope, time, cost, and quality management. Work Breakdown Structure (WBS). Resource allocation, risk management, and monitoring progress. Use of tools like Gantt charts, PERT, and CPM for project tracking.		
M5	**Software Testing**	8	20
	Principles of verification and validation, testing objectives and principles, error detection, Software Testing : Objective, Pimples, Test case Design for conventional software (Unit testing, Integration Testing), path testing , cyclometic complexity, System testing (Recovery testing, Security testing, stress testing, Performance testing)		
M6	**Software Quality Assurance And Reliability Metrics**	5	15
	Concepts of quality and software reliability, reliability metrics, quality control vs quality assurance, IEEE standards for statistical quality assurance, and metrics for software quality.		

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

(Formerly West Bengal University of Technology)

Syllabus of B. Sc. in IT

(Effective from 2023-24 Academic Sessions)

Module-Wise Breakdown of Syllabus

Module No.	Name of the Topic	Hours	Marks	COs Covered	Remarks
M1	Introduction: The evolving role of software, software myths, and process frameworks.	6	10	CO1	Establishes foundational understanding of Software Engineering.
M2	Software Life Cycles: Waterfall, Iterative, Spiral, Prototype, and Evolutionary models.	6	15	CO1, CO2	Explains lifecycle models and their practical applicability.
M3	Software Requirement and Specification Analysis: DFDs, decision tables, and COCOMO model.	10	20	CO2, CO3	Provides tools for structured analysis and design techniques.
M4	Software Planning and Project Management: WBS, risk management, and tracking tools like Gantt charts.	10	20	CO2, CO3, CO4	Focuses on planning, resource management, and tracking tools.
M5	Software Testing: Unit testing, integration testing, path testing, and system testing techniques.	8	20	CO3, CO4, CO5	Strengthens skills in testing and validation for reliable systems.
M6	Software Quality Assurance and Reliability Metrics: IEEE standards, quality metrics.	5	15	CO5	Covers the standards and metrics for ensuring software quality.

List of Suggested Readings

1. ***Software Engineering: A Practitioner's Approach*** by Roger S. Pressman - McGraw Hill
2. ***Software Engineering*** **by Ian Sommerville** - Pearson Education
3. ***Fundamentals of Software Engineering*** by Rajib Mall - PHI
4. ***Software Testing Principles and Practices*** by Naresh Chauhan - Oxford University Press
5. ***Software Engineering Concepts*** by Richard Fairley - Tata McGraw Hill

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL
(Formerly West Bengal University of Technology)
Syllabus of B. Sc. in IT
(Effective from 2023-24 Academic Sessions)

Paper Name: Operating System with UNIX
Code: BSCITM402
Credits: 3

Weekly Class Structure

- **Theory Sessions**: 3 per week (1-hour each)
- **Lab Sessions**: 1 per week (2-hours each)

Course Objectives

1. *To understand the core concepts of Operating Systems with a focus on UNIX.*
2. *To explore UNIX-based OS components, file systems, and shell programming.*
3. *To provide practical experience with UNIX commands, utilities, and system programming.*
4. *To introduce process management, memory management, and inter-process communication in UNIX.*

Learning Outcomes (COs)

1. **CO1**: Explain the fundamental principles of Operating Systems and UNIX.
2. **CO2**: Demonstrate proficiency in using UNIX commands, scripting, and utilities.
3. **CO3**: Analyze process, memory, and file system management in UNIX environments.
4. **CO4**: Develop shell scripts and simple system programs in UNIX.
5. **CO5**: Apply Operating System concepts to real-world scenarios, focusing on UNIX-based platforms.

Sl. No	Course Outcome	Mapped Modules
1.	Explain the fundamental principles of OS and UNIX	Module 1: Introduction to Operating Systems and UNIX
2.	Proficiency in UNIX commands, scripting, and utilities	Module 2: UNIX Basics and Shell Programming
3.	Process, memory, file system management and deadlock in UNIX	Module 3 ,Module 4: Process and Memory Management, Module 5: File Systems in UNIX Module 6:
4.	Develop shell scripts and system programs	Module 2: UNIX Basics and Shell Programming,
5.	Apply OS concepts to UNIX-based platforms	Modules 1–6 (Theory and Practical Sessions)

Course Content

Module 1: Introduction to Operating Systems and UNIX

- Overview of Operating Systems: Functions : Types of Operating Systems (Batch, Time-Sharing, Distributed, Real-Time)
- Operating System Structure: Monolithic, Layered, Microkernel, Virtual Machines
- OS Evolution and the Role of OS in Computing

Module 2: UNIX Basics and Shell Programming

- Introduction to UNIX: Architecture, Kernel, Shell
- History and Evolution of UNIX
- Comparison of UNIX with other Operating Systems
- Basic UNIX Commands: File manipulation, Text Processing, Process Monitoring
- Shell Scripting: Variables, Control Structures, Functions
- UNIX Utilities: grep, sed, awk, tar, make
- Introduction to Regular Expressions

Module 3: Process Management in UNIX

- **Processes concept**:
    - Creation, Execution, and Termination
    - Process vs. Program
    - Process Control Block (PCB)
    - Process States: New, Ready, Running, Waiting, Terminated

- Context Switching
- Thread: Definition ,Various states, Benefits of threads, Types of threads ,Concepts of multithreadin

- **Process Scheduling and Priorities in UNIX:**
  - First-Come-First-Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), Priority Scheduling, Multi-Level Queue Scheduling
  - Comparison and Analysis of Scheduling Algorithms
- **Inter-Process Communication (IPC):**
  - Pipes
  - Critical section, Mutual exclusion, Race Condition,
  - Peterson's Solution, Producer Consumer problem, Reader's Writer's problem, Dinning Philosopher Problem
  - Shared Memory, Message Passing, Monitors
  - Synchronization of Processes: Critical Section Problem, Semaphores, Mutexes
- **Signals and Signal Handling**

Module 4: Memory Management in UNIX

- **Memory Hierarchy**
  - Cache Memory, Main Memory, Virtual Memory
- **Memory Allocation Techniques**
  - Contiguous Allocation, Paging, Segmentation
  - Fragmentation: External and Internal Fragmentation
  - Compaction
- **Virtual Memory**
  - Page Replacement Algorithms: FIFO, LRU, Optimal
  - Thrashing
  - Demand Paging and Page Faults
- **UNIX Memory Architecture**
- **Swapping and Process Address Space**
- **Case Study: Memory Management in Linux**

Module 5: File System Management in UNIX

- **File Concepts**
  - File Attributes, Operations, and Types
  - File Access Methods: Sequential, Random, Direct Access
- **File System Implementation**
  - Directory Structure, File Allocation Methods (Contiguous, Linked, Indexed)
  - File System Mounting
- **I/O Management**
  - I/O Devices and Controller, Device Drivers
  - Disk Scheduling Algorithms: FCFS, SSTF, SCAN, C-SCAN,LOOK
  - Disk Reliability , Disk Formatting, Boot block, Bad blocks
- **System Calls for File Handling**:
  - open, read, write, close, etc.

Module 6: Deadlock and Security

- **Deadlock**
  - Conditions for Deadlock (Mutual Exclusion, Hold and Wait, No Preemption, Circular Wait)
  - Deadlock Prevention, Avoidance, Detection, and Recovery
  - Banker's Algorithm for Deadlock Avoidance
- **Operating System Security**
  - User Authentication, Access Control, Encryption
  - Security Threats: Malware, Viruses, Trojans
  - System Hardening Techniques

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

(Formerly West Bengal University of Technology)

Syllabus of B. Sc. in IT

(Effective from 2023-24 Academic Sessions)

Module-Wise Breakdown of Syllabus

Module No.	Content	Total Hours	% of Questions Covered	CO Covered	Bloom's Level	Remarks (if any)
1	Introduction to Operating Systems and UNIX	6	10%	CO1	Remembering, Understanding	
2	UNIX Basics and Shell Programming	8	10%	CO2, CO4	Understanding, Applying	
3	Process Management in UNIX	10	25%	CO3, CO5	Analyzing, Applying	
4	Memory Management in UNIX	8	15%	CO3, CO4	Analyzing	
5	File System Management in UNIX	10	20%	CO3, CO4	Applying, Evaluating	
6	Deadlock and Security	8	20%	CO5	Applying, Creating	

Recommended Textbooks and References

1. **"Operating System Concepts"** by Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne
2. **"Modern Operating Systems"** by Andrew S. Tanenbaum
3. **"UNIX Shell Programming"** by Yashwant Kanetkar
4. **"Advanced Programming in the UNIX Environment"** by W. Richard Stevens
5. Linux and BSD official documentation

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL
(Formerly West Bengal University of Technology)
Syllabus of B. Sc. in IT
(Effective from 2023-24 Academic Sessions)

Lab Paper: UNIX lab
Code: BSCITM492
Credits: 1

Weekly Lab Structure

- **Number of Sessions**: 1 session per week (2 hour).
- **Tools Required**: Linux OS (Ubuntu or similar), Bash Shell, GCC Compiler.

Lab Objectives

1. To practice basic and advanced UNIX commands.
2. To write simple shell scripts for automating tasks.
3. To understand basic system programming concepts like process management and file handling.
4. To learn to work with file systems, permissions, and memory management in UNIX.

Lab Outcomes

By the end of the lab, students will be able to:

1. Use UNIX commands effectively for basic file and process management.
2. Write simple shell scripts to perform routine tasks.
3. Implement basic programs for process and file management in UNIX.
4. Understand and manipulate file systems, memory, and processes in UNIX.

Lab Experiments

Section 1: Basic UNIX Commands

1. **Navigating the File System**: Learn commands like cd, ls, pwd.
2. **Basic File Operations**: Use commands like cp, mv, rm, mkdir, and touch.
3. **Viewing File Content**: Practice with cat, more, less, head, and tail.
4. **File Permissions**: Learn to change file permissions with chmod and chown.
5. **Managing Processes**: Use ps, top, kill, and jobs to manage processes.

Section 2: Shell Scripting

6. **Simple Shell Scripts**: Write scripts that print text, perform calculations, and take input.
7. **Control Flow in Scripts**: Use if, for, and while loops to control script behaviour.
8. **Working with Files in Scripts**: Use grep, sed, awk for simple text processing.
9. **Automating Tasks**: Write a script to back up files or check system health.

Section 3: Process Management

10. **Creating Processes**: Write a simple C program to create a new process using fork().
11. **Inter-Process Communication (IPC)**: Practice using pipes (|) for communication between commands.
12. **Signals**: Write a program that handles signals using signal() in C.

Section 4: File System Management

13. **Exploring File Systems**: Check disk usage with df and file space with du.
14. **Creating and Reading Files**: Use open(), read(), write(), and close() system calls in C.
15. **Modifying File Permissions**: Write a program that changes file permissions using chmod.

Section 5: Memory Management

16. **Viewing Memory Usage**: Use free and vmstat commands to view memory usage.
17. **Simulating Memory Allocation**: Write a C program to allocate memory dynamically and manage it using malloc() and free().

Recommended Tools and Resources

1. **Operating System Environment**: Linux (Ubuntu, Fedora, or CentOS).
2. **Editor**: Vim, Nano, or VS Code.
3. **Compilers**: GCC for C programming.
4. **Books**:
   - **"The Linux Command Line"** by William E. Shotts.
   - **"Advanced Programming in the UNIX Environment"** by W. Richard Stevens.
   - Linux man pages (man command).

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL
(Formerly West Bengal University of Technology)
Syllabus of B. Sc. in IT
(Effective from 2023-24 Academic Sessions)

Theory Paper: OOPs with JAVA (Code: BSCITM403)
lab Paper name: OOPs with JAVA lab (Code: BSCITM493)

CREDIT: 3 (Theory) + 2 (Practical)
CONTACT HOURS/WEEK: 4H(L) + 2H(P)

COURSE OBJECTIVE:

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

1. Provide an in-depth understanding of various concepts of object-oriented programming.
2. Develop the ability to read, understand, and trace the execution of Java programs.
3. Enhance debugging skills for programming.
4. Gain the skill of writing Java programs to solve real-world problems.
5. Introduce students to a powerful programming language.
6. Enable students to design algorithms and flowcharts for given problems.

COURSE OUTCOME:

- **CO1:** Explain the fundamentals and advantages of object-oriented programming and its differences from conventional programming.
- **CO2:** Design and implement object-oriented concepts like classes, objects, inheritance, encapsulation, and polymorphism using Java.
- **CO3:** Apply control statements, loops, arrays, and basic string handling techniques in Java programs.
- **CO4:** Develop reusable code using inheritance, interfaces, and packages.
- **CO5:** Handle exceptions and implement multithreading to write robust and efficient Java programs.

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

(Formerly West Bengal University of Technology)

Syllabus of B. Sc. in IT

(Effective from 2023-24 Academic Sessions)

UNITS: COURSE CONTENT

Unit No.	Unit Name	Description	Lecture Hours	Marks
Unit 1	**Object-Oriented Design**	- Introduction to object-oriented programming languages. - Major and minor elements of OOP. - Objects and classes. - Relationships among objects and classes: aggregation, association, instantiation, grouping constructs.	6	15
Unit 2	**Object-Oriented Concepts**	- Differences between OOP and conventional programming. - Advantages and disadvantages of OOP. - Key OOP concepts: Class, Object, Message Passing, Inheritance, Encapsulation, Polymorphism.	8	20
Unit 3	**Basic Concepts of Object-Oriented Programming Using Java**	- Implementation of OOP concepts using Java. - Introduction to Java programming language.	6	10
Unit 4	**Class and Object Properties**	- Java basics: advantages, byte-code, JVM. - Data types, access specifiers, operators, control statements, loops. - Arrays, classes, objects, constructors. - Garbage collection, method overloading, this keyword, static variables/methods, nested and inner classes. - String handling: String and StringBuffer class methods. - Command-line arguments.	8	20
Unit 5	**Reusability**	- Super class and subclasses, multilevel	8	20

	Properties	hierarchy. - Constructor calling in inheritance. - Use of super and final keywords. - Dynamic method dispatch. - Abstract classes and interfaces. - Creating and importing packages.		
Unit 6	**Exception Handling and Multithreading**	- Basics of exception handling: try, catch, throw, throws, and finally. - User-defined exception classes. - Basics of multithreading: thread lifecycle, thread priorities, synchronization, deadlocks, inter-thread communication.	8	15

List of Suggestive Lab/Practical:

1. Write a program to print "Hello World" in Java.
2. Write a program to calculate the factorial of a number using recursion.
3. Create a class Employee with properties name, id, and salary. Implement methods to input and display employee details.
4. Write a program to implement single and multilevel inheritance with constructors.
5. Demonstrate polymorphism using method overriding and dynamic method dispatch.
6. Write a program to implement interfaces and abstract classes.
7. Create and use a custom package in Java with multiple classes.
8. Write a program to demonstrate the use of constructors and the this keyword.
9. Create a program that demonstrates the use of static variables and static methods.
10. Write a program to perform basic string operations such as concatenation, comparison, and finding substrings.
11. Demonstrate the use of StringBuffer methods such as append, insert, and reverse.
12. Write a program to check if a given string is a palindrome.
13. Implement exception handling using try, catch, throw, and finally for division by zero.
14. Create a custom exception for validating user input (e.g., age should be greater than 18).
15. Write a program to create threads using the Runnable interface and Thread class.

16. Demonstrate thread synchronization using a producer-consumer problem.
17. Write a program to demonstrate the use of arrays and perform operations like searching and sorting.
18. Create a program to implement method overloading for mathematical operations.
19. Write a program to demonstrate nested and inner classes in Java.
20. Implement multithreading to calculate the sum of elements in an array using multiple threads.
21. Implement garbage collection and finalize methods in Java with object lifecycle.
22. Write a program to demonstrate method calls using call by value and call by reference.
23. Create a program to implement constructor chaining using the super keyword.
24. Write a program to create and handle multiple exceptions in a single program.
25. Write a program to print the sum, difference, product, and quotient of two numbers.
26. Write a program to check if a given number is odd or even.
27. Write a program to find the largest of three numbers using conditional statements.
28. Write a program to swap two numbers without using a third variable.
29. Write a program to calculate the area and perimeter of a rectangle.
30. Write a program to find whether a given year is a leap year or not.
31. Write a program to reverse a given number.
32. Write a program to check if a given number is a palindrome.
33. Write a program to calculate the sum of the digits of a number.
34. Write a program to print the multiplication table of a given number.
35. Write a program to check if a given number is prime.
36. Write a program to find all prime numbers between 1 and 100.
37. Write a program to calculate the GCD (Greatest Common Divisor) and LCM (Least Common Multiple) of two numbers.
38. Write a program to print the Fibonacci series up to a given number of terms.
39. Write a program to find the factorial of a number without recursion.
40. Write a program to sort an array in ascending and descending order.
41. Write a program to search for an element in an array using linear search.
42. Write a program to find the second largest element in an array.
43. Write a program to merge two arrays into one.
44. Write a program to calculate the average of elements in an array.

45. Write a program to count the occurrences of a specific character in a string.
46. Write a program to replace all occurrences of a specific character in a string.
47. Write a program to check if two strings are anagrams of each other.
48. Write a program to find the smallest and largest word in a given string.
49. Write a program to demonstrate the use of basic for, while, and do-while loops.
50. Write a program to calculate the power of a number using a loop.

Reference books:

- **E. Balagurusamy** – Programming with Java, McGraw-Hill Education.
- **Kathy Sierra, Bert Bates** – Head First Java, O'Reilly Media.
- **Herbert Schildt** – Java: The Complete Reference, McGraw-Hill Education.
- **Y. Daniel Liang** – Introduction to Java Programming and Data Structures, Pearson.
- **Paul Deitel, Harvey Deitel** – Java: How to Program, Pearson.
- **R. Nageswara Rao** – Core Java: An Integrated Approach, Dreamtech Press.
- **T. Budd** – Understanding Object-Oriented Programming with Java, Pearson.