LESSON PLAN 2025-26(WINTER)

NAME OF THE TEACHER : DEEPAK KUMAR BARDA, LECT.(STAGE-II,CSE)					
Subject: **SOFTWARE ENGINEERING (Course Code:TH3)** Program: Diploma in Computer Science and Engineering Semester: 3rd Total Contact Hours: 60 Total Marks: 100 Assessment: Internal Assessment – 20, End Term – 80					
After completion of this course the student will be able to: CO1-Understand the concept of Software Engineering. CO2-Understand how costs, schedule and quality drive a software project. CO3-Understand the role of software process and a process model in a project. CO4-Understand planning and estimation of a software project. CO5-Understand the role of SRS in a project and how requirements are validated CO6-Know the key design concepts of software engineering. CO7-Learn the structured code inspection process. CO8-Learn how testing is planned and testing done.					
Period	**Topic**	**Learning Objectives**	**Activity**	**Homework**	**COURSE OBJECTIVE**
Unit 1: Introduction to Software Engineering(06 Periods)					
1	Program vs. Software Product	Differentiate between a single program and a complex, maintainable software product.	Discuss the difference between a simple "Hello World" program and a web browser.	Find and analyze an example of a simple program and a complex software product.	CO1
2	Emergence of Software Engineering.	Understand the historical context and reasons for the formalization of software development.	Class discussion on the "software crisis" and its impact.	Read a short article on the history of software development.	CO1
3	Classical Waterfall Model and its limitations.	Explain the sequential phases of the Waterfall Model and identify its weaknesses.	Draw a simple Waterfall diagram on the board and label its phases.	Research a project type for which the Waterfall model is well-suited.	CO1
4	Iterative Waterfall and Prototyping Models.	Describe how these models address the limitations of the classical Waterfall model.	Role-play a scenario where prototyping is used to refine requirements with a client.	Compare and contrast the Iterative Waterfall and Prototyping models in a short report.	CO1
5	Evolutionary and Spiral Models.	Understand how these models accommodate change and manage risk throughout the development cycle.	Draw the Spiral Model diagram and explain its iterative, risk-driven nature.	Find a real-world example of a project that used the Spiral Model.	CO1

6	Comparison of all models.	Evaluate different models based on project type, size, and risk.	Group activity where each group is given a project and must justify which model they would use.	Final review of Unit 1 topics.	CO1
Unit 2: Software Project Management(10 Periods)					
7	Responsibility of a Project Manager. Project Planning.	List the key roles and responsibilities of a project manager.	Brainstorm a list of tasks a project manager would perform on a daily basis.	Research the "triple constraints" (scope, time, cost) of project management.	CO2
8	Metrics for Project size estimation (LOC and FP).	Calculate project size using Lines of Code (LOC) and Function Point (FP) analysis.	Give a simple software description and have students practice a basic Function Point count.	Complete a worksheet on calculating LOC and FP for a given problem statement.	CO2
9	Project Estimation Techniques.	Explain different methods for estimating project effort and cost.	Discuss the differences between expert judgment, analogy, and decomposition techniques.	Read about the differences between a bottom-up and top-down estimation approach.	CO2
10	COCOMO Models (Basic and Intermediate).	Apply the COCOMO I and II models to estimate a project's effort and schedule.	Work through a COCOMO Basic calculation for a project on the board.	Practice using COCOMO Intermediate with a different set of cost drivers.	CO2
11	COCOMO Model (Complete).	Explain the differences and increased complexity of the complete COCOMO model.	Group discussion on when a complete COCOMO model would be necessary.	Study the various cost drivers used in the COCOMO models.	CO2
12	Scheduling.	Understand the importance of scheduling and techniques like Gantt charts.	Students create a simple Gantt chart for a class project they've been assigned.	Research and explain the critical path method (CPM).	CO2
13	Organization and Team structure. Staffing.	Describe different team structures and the roles within a software team.	Discuss the pros and cons of hierarchical vs. flat team structures.	Write a job description for a "Software Engineer" and a "Software Project Manager."	CO2
14	Risk Management.	Identify, analyze, and mitigate project risks.	Brainstorm a list of potential risks for a project and create a simple risk register.	Read a case study of a project that was derailed by unmanaged risks.	CO2
15	Configuration Management.	Explain the importance of version control and managing changes to a project's artifacts.	Live demonstration of a version control system like Git.	Install Git and set up a simple repository on their local machine.	CO2

16	Unit 2 Review.	Consolidate knowledge on project management concepts.	Q&A session covering all topics from the unit.	Study for an upcoming quiz on Unit 2.	CO2
Unit 3: Requirement Analysis and Specification(06 Periods)					
17	Requirements gathering and analysis.	Understand techniques for eliciting and analyzing software requirements.	Role-play an interview with a "client" to gather requirements for a new app idea.	Write down the requirements gathered from the role-play.	CO3
18	Software Requirements Specification (SRS).	Explain the purpose and importance of an SRS document.	Review a sample SRS document provided by the instructor.	Outline the sections of a basic SRS document.	CO3
19	Contents of SRS and characteristics of a good SRS.	Describe the key sections of an SRS and what makes a requirement effective.	As a class, evaluate some example requirements and determine if they are "good" or "bad."	Rewrite a set of bad requirements to make them "good."	CO3
20	Organization of SRS.	Understand how to structure a large SRS document.	Discuss different ways to organize requirements (e.g., by feature, by user).	Reorganize the outline from a previous homework assignment.	CO3
21	Techniques for representing complex logic.	Use decision tables, decision trees, and state transition diagrams to represent complex system behavior.	Create a decision table for a simplified login process with multiple conditions.	Draw a state transition diagram for a user's lifecycle in a social media app.	CO3
22	Unit 3 Review.	Consolidate knowledge of requirement analysis.	Q&A session and a short quiz on the key concepts.	Prepare for the midterm exam (covering Units 1-3).	CO3

Unit 4: Software Design(10 Periods)					
23	What is a good S/W design. Cohesion and Coupling.	Define the concepts of cohesion and coupling and explain their relationship to good design.	Provide code snippets and have students identify high vs. low cohesion and tight vs. loose coupling.	Read a chapter on design principles from a recommended textbook.	CO4
24	Neat arrangement. S/W Design approaches.	Understand the importance of clear, organized design and different design strategies.	Discuss the differences between object-oriented and structured design.	Draw a simple class diagram for a software system.	CO4
25	Structured analysis. Data Flow Diagrams (DFDs).	Introduce the concepts of structured analysis and the role of DFDs.	Draw the symbols used in DFDs and their meanings.	Draw a simple context diagram (Level 0 DFD) for a library system.	CO4
26	Designing DFDs.	Create leveled DFDs (Level 1, Level 2, etc.) to show a system's functionality.	Work through a complete DFD example for an online food ordering system.	Create a set of leveled DFDs for a university's student registration system.	CO4
27	Shortcomings of DFDs. Structured design.	Identify the limitations of DFDs and introduce the concept of structured design.	Discuss what DFDs don't show (e.g., control flow, data structure).	Read about the transition from structured analysis to structured design.	CO4
28	Principles of transformation of DFD to Structure Chart.	Understand the systematic process of converting a DFD into a Structure Chart.	Walk through a transformation from a DFD to a Structure Chart on the board.	Given a DFD, draw the corresponding Structure Chart.	CO4
29	Transform analysis.	Apply Transform Analysis to a DFD to derive a Structure Chart.	Practice identifying the "central transform" in a DFD.	Work on a Transform Analysis problem.	CO4
30	Transaction analysis.	Apply Transaction Analysis to a DFD to derive a Structure Chart.	Practice identifying the "transaction center" in a DFD.	Work on a Transaction Analysis problem.	CO4
31	Design Review.	Explain the purpose and process of a design review.	Conduct a mock design review for a simple system design created by the class.	Prepare a short presentation on a design review checklist.	CO4

32	Unit 4 Review.	Consolidate knowledge on software design principles and techniques.	Q&A session and a review of key diagrams.	Study for an upcoming quiz on Unit 4.	CO4
Unit 5: User Interface Design(08 Periods)					
33	Characteristics of a Good Interface.	Identify the key qualities of an effective user interface, such as usability, learnability, and consistency.	Evaluate the user interface of two popular mobile apps and compare their strengths and weaknesses.	Read about Nielsen's 10 Usability Heuristics.	CO5
34	Basic concepts of UID.	Understand fundamental principles of user interface design.	Discuss the difference between a good user experience (UX) and a good user interface (UI).	Find examples of good and bad UI design online and provide a brief critique.	CO5
35	Types of User Interfaces.	Differentiate between command-line, graphical, touch, and other types of user interfaces.	Discuss the advantages and disadvantages of each interface type for different applications.	Design a simple interface for a smart home device using a pen and paper.	CO5
36	Components-based GUI development.	Explain how reusable components are used to build user interfaces.	Discuss common UI components like buttons, text fields, and dropdowns.	Research a popular front-end framework (e.g., React, Angular) and its component model.	CO5
37	Components-based GUI development (continued).	Gain practical understanding of using a GUI toolkit.	Live demonstration of building a simple UI with a drag-and-drop tool or a front-end framework.	Use a GUI toolkit to create a simple form with various components.	CO5
38	Design for accessibility.	Understand the importance of creating user interfaces that are accessible to all users, including those with disabilities.	Watch a video on screen readers and other assistive technologies.	Research the Web Content Accessibility Guidelines (WCAG).	CO5
39	Design review and feedback.	Learn how to give and receive constructive feedback on UI designs.	Peer review of the UI mock-ups created in a previous homework assignment.	Refine their UI design based on the feedback received.	CO5
40	Unit 5 Review.	Consolidate knowledge of UID.	Q&A and a short quiz.	Prepare for an upcoming quiz on Unit 5.	CO5
Unit 6: Software Coding & Testing(12 Periods)					
41	Coding and Code Review.	Understand the role of coding standards and the value of code reviews.	Provide a poorly written code snippet and have students refactor it according to good coding practices.	Write a short document on best practices for a specific programming language.	CO6

42	Code walk-throughs and inspections.	Describe the structured process of a code walk-through and a code inspection.	Conduct a mock code walk-through with a simple function.	Read a short article on the differences between code walk-throughs and inspections.	CO6
43	Introduction to Software Testing. Unit Testing.	Explain the purpose of testing and how to perform unit tests.	Write a simple function and then write a unit test for it using a testing framework.	Complete a worksheet on writing unit tests for various functions.	CO6
44	Black Box Testing.	Create test cases based on external specifications without knowing the internal code.	Given a specification for a calculator, have students create black-box test cases.	Create black-box test cases for a simple website form.	CO6
45	Equivalence Class Partitioning and Boundary Value Analysis.	Apply these techniques to generate effective black-box test cases.	Practice applying Equivalence Class Partitioning and Boundary Value Analysis to a problem on the board.	Complete a problem set applying these techniques to a new scenario.	CO6
46	White Box Testing.	Create test cases based on the internal structure and logic of the code.	Provide a code snippet with conditional statements and have students draw a control flow graph.	Given a simple program, write white-box test cases to achieve full statement coverage.	CO6
47	White Box Methodologies: Statement, Branch, and Condition coverage.	Define and apply these different coverage metrics.	Practice creating test cases to achieve each type of coverage for a given code block.	Write test cases to achieve 100% branch and condition coverage for a provided function.	CO6
48	White Box Methodologies: Path coverage, Cyclomatic Complexity.	Explain path coverage and calculate the Cyclomatic Complexity of a function.	Walk through the calculation of Cyclomatic Complexity for a function with multiple loops and if-statements.	Calculate the Cyclomatic Complexity for a different function.	CO6
49	Debugging approaches and guidelines.	Understand systematic methods for finding and fixing bugs.	"Debug this!" challenge. Give students a broken program and have them use debugging tools to find the error.	Write a blog post on their favorite debugging technique.	CO6
50	Integration Testing. Phased and incremental integration testing.	Understand how to test the integration of different modules.	Discuss the differences between top-down and bottom-up integration strategies.	Research and explain the "big-bang" integration approach.	CO6
51	System testing (Alpha, Beta, and Acceptance testing).	Explain the purpose and audience for each type of system test.	Discuss a recent software release and whether it went through Alpha or Beta testing.	Write a brief acceptance test plan for a new feature on a well-known website.	CO6

52	Performance Testing, Error seeding, General issues.	Understand the importance of performance, security, and other non-functional testing types.	Discuss a scenario where performance testing would be critical (e.g., a high-traffic e-commerce site).	Review and summarize the general issues associated with testing from the syllabus.	CO6
Unit 7: Software Reliability(08 Periods)					
53	Software Reliability.	Define software reliability and its distinction from software quality.	Discuss how reliability impacts a user's trust in a system.	Research the difference between reliability and availability.	CO7
54	Different reliability metrics.	Explain and calculate key reliability metrics like Mean Time Between Failures (MTBF).	Work through a problem set on calculating MTBF and other metrics.	Find a software reliability model and explain its components.	CO7
55	Reliability growth modeling.	Understand how to use models to predict and track software reliability over time.	Discuss the concept of a reliability growth curve and its implications.	Read a short paper on a specific reliability growth model (e.g., Jelinski-Moranda model).	CO7
56	Software quality.	Define software quality and its various attributes.	Brainstorm a list of what "quality" means for a software application.	Research the ISO 9126 software quality model.	CO7
57	Software Quality Management System.	Explain the components and purpose of a quality management system.	Discuss the role of quality assurance (QA) in a project.	Read a case study on a company's implementation of a quality management system.	CO7
58	General issues associated with testing.	Consolidate understanding of common challenges and best practices in testing.	Class discussion on the challenges of testing complex systems.	Prepare questions for a final review session.	CO7
59	Final review.	A comprehensive review of all course topics.	Open Q&A session.	Prepare for the final exam.	CO7
60	Final review (continued).	Further consolidation of knowledge.	Practice exam or a final case study.	Last-minute final exam preparation.	CO7

Deepak kumar Bardh 11.07-2025	Deepak kumar Bardh 11.07.2025
Signature of Teacher	**Signature of HOD**