Gyanmanjari Innovative University

Syllabus
Gyanmanjari Institute of Technology
Semester-2 (B. Tech.)

**Subject:** Software Engineering: Crafting Scalable and Secure Applications – BET1CE12303

**Type of course:** Engineering Science Courses

**Prerequisite:** Basic Knowledge of Programming and Computer Fundamentals

**Rationale:**

This course provides a holistic understanding of the software development lifecycle by introducing fundamental software engineering concepts, requirements gathering and modeling, architectural and design principles, coding practices and testing methodologies, and essential project management techniques. Students learn how to identify and document requirements using elicitation methods and UML diagrams, design scalable and secure architectures using modern design concepts, follow coding standards with proper reviews and documentation, and apply testing strategies using appropriate tools for different types of applications. The course also builds competency in handling project planning, cost estimation, risk management, scheduling, and quality assurance using recognized standards such as ISO and CMMI, ultimately preparing learners to manage real-world software projects effectively from conception to deployment.

## Teaching and Examination Scheme:

| Teaching Scheme | | | Credits | Examination Marks | | Total Marks |
|---|---|---|---|---|---|---|
| CI | T | P | C | SEE | CCE | |
| 4 | 0 | 2 | 5 | 100 | 50 | 150 |

*Legends: CI-Class Room Instructions; T – Tutorial; P - Practical; C – Credit; SEE - Semester End Evaluation; CCE-Continuous and Comprehensive Evaluation.*

**Course Content:**

| Sr. No. | Course Content | Hrs. | % Weightage |
|---|---|---|---|
| 1 | **Introduction to Software Engineering:**<br>***Theory Topics:***<br>Software Concept, Characteristics and Types, Software Application Domains, Definition of Software Engineering, The Software Process, Software Crisis, Software Myths, Software Quality Attributes, Software Development Life Cycle (SDLC) Overview, Umbrella Activities: Software Quality Assurance (SQA), Documentation, Project Management, Configuration Management, Risk Management, Software Process Models: The Waterfall Model, Incremental Process Model, The Spiral Model, Prototyping Model, Define Agility and the cost of change, Agile Process Model: Adaptive Software Development (ASD), Scrum, Dynamic Systems Development Method (DSDM). Agile Project Management Tools (JIRA, Trello)<br>***Practical:***<br>1. Create a concept map illustrating software characteristics and types, showing relationships between system software, application software, and development tools using Canva or PowerPoint.<br>2. Identifying and categorizing different types of software by listing real examples and classifying them based on their purpose and usage in MS Excel.<br>3. Prepare a short case study document explaining the software crisis and how software engineering principles evolved to address it using MS Word.<br>4. Analyzing a real software failure case to understand contributing factors and identify common software myths that impacted the project in MS Word.<br>5. Creating a visual representation of the SDLC phases to clearly illustrate the sequential stages of software development from requirements to maintenance using tools like, Lucidchart, Draw.io (diagrams.net), Canva, PowerPoint, Miro, etc.<br>6. Creating a paper-based prototype to visually simulate the user interface and workflow of a simple application for early feedback in Canva Whiteboard or paper-based.<br>7. Comparing major software process models in a structured table to highlight their steps, usage scenarios, strengths, and limitations in Excel or Word.<br>8. Simulating an Agile Scrum workflow using a task board to visualize task progress through To-Do, Doing, and Done stages in Jira or Excel. | 18 | 20% |

9. Create a burndown chart for a hypothetical Scrum sprint using Excel to track remaining work over time.
10. Preparing basic SQA and risk management tables to track quality checks and identify potential project risks with their impact and preventive measures in Excel or Jira.

**Evaluation Method:**

| Sr. No. | Evaluation Methods | SEE | CCE |
|---|---|---|---|
| 1 | **Software Process Analysis and Prototype Simulation:** Students will analyze a given software application using tools like MS Word, PowerPoint, Draw.io, or Canva to classify the software, show SDLC phases, compare two process models, and create a simple module prototype/workflow. Optional SQA and risk tables may be included. Submit all work as one PDF. | 20 | |
| 2 | **Active Learning Assignment Process Model Selection for a Personal Mini-Project:** Choose a small application you want to develop and write a short paragraph explaining which software process model suits it best and why. Save your work as a PDF and upload it to the GMIU portal. | | 10 |
| | Total | 20 | 10 |

***Examination Style:***

**Software Process Analysis and Prototype Simulation (20 Marks)**

The faculty will provide a software application problem (such as a library system). Students should first study the given problem to clearly understand its purpose and main functionalities. Using tools like MS Word, PowerPoint, Draw.io, or Canva, students must then:

1. Classify the software based on its type and usage.
2. Visually represent the SDLC phases followed for the application.
3. Compare any two software process models using a table that includes their phases, use cases, advantages, and limitations.
4. Simulate a simple prototype or workflow for one selected module or feature, showing how it works step by step.

Optionally, students may also prepare basic Software Quality Assurance (SQA) details and risk management tables to identify quality measures and potential risks. After completing all sections, students should compile their work into a single PDF file.

| | | | |
|---|---|---|---|
| | **Process Model Selection for a Personal Mini-Project (10 Marks)**<br>Each student must select a small personal project idea (for example, a fitness tracker, reminder app, timetable manager, or budget calculator). Write a short paragraph explaining which Software Process Model (Waterfall, Incremental, Spiral, Prototyping, or Agile) is most suitable for your project and justify your choice. Ensure that each student's project idea is unique.<br>Perform the Task as per below:<br>1. Choose a small application you want to develop.<br>2. Analyze your project requirements and decide which software process model will work best.<br>3. Write a clear paragraph justifying your choice.<br>4. Save your paragraph as a PDF file.<br>5. Upload the PDF to the GMIU portal within the deadline. | | |
| 2 | **Requirements Analysis and Software Requirements Specification:**<br>***Theory Topics:***<br>Requirement Engineering, Requirements Engineering Process, Requirements Gathering (Studying existing documentation, Interview, Questionnaire, Record Review, Task Analysis, Observation), Tools for Documenting Procedures and Decisions (Decision Concepts, Decision Trees, Decision Tables), Requirements Analysis, Software Requirements Specifications (SRS) Document, Format, IEEE Standards for SRS Documents, SRS Validation, Components of SRS, Characteristics of SRS, Types of Requirements: Functional, Non-functional, design and implementation constraints, external interfaces required, Other non-functional requirements, Requirements Validation & Verification, Entity-Relationship Diagram.<br>***Practical:***<br>11. Creating a visual flow diagram of the Requirements Engineering process to represent how requirements are elicited, analyzed, documented, validated, and managed.<br>12. Prepare a requirement gathering report for a Student Management System using the interview method, documenting stakeholder questions and collected requirements.<br>13. Prepare a requirement gathering report for a Student Management System using the questionnaire method, including designed questions and analyzed responses.<br>14. Prepare a requirement gathering report for a Student Management System using the observation method, recording user activities and system interactions.<br>15. Prepare a requirement gathering report for a Student Management System using the record review method, analyzing existing documents, forms, or reports. | 18 | 20% |

| | | | |
|---|---|---|---|
| | 16. Prepare a requirement gathering report for a Student Management System using task analysis, breaking down user tasks and workflows.<br>17. Design a questionnaire with 10–15 questions to collect requirements for a chosen system.<br>18. Conduct a sample interview and prepare an interview summary highlighting key user requirements.<br>19. Collecting real user requirements through a structured online survey to understand needs and expectations for a mini project.<br>20. Create a decision tree for a selected decision-making scenario in your system.<br>21. Prepare a decision table showing conditions and actions for the same scenario used in the decision tree.<br>22. Preparing a software requirements specification (SRS) in IEEE format to formally document system requirements, descriptions, interfaces, and constraints.<br>23. List all components of your SRS document and map them to the corresponding sections of your system.<br>24. Verifying and validating documented requirements using a checklist to ensure they are complete, consistent, unambiguous, and properly reviewed.<br>25. List the functional requirements for a Student Management System to clearly specify what the software must do.<br>26. List the non-functional requirements for a Student Management System to clearly specify the quality attributes and constraints the software must meet.<br>27. Prioritizing software requirements using the MoSCoW method to categorize them as Must-Have, Should-Have, Could-Have, or Won't-Have for better project planning.<br>28. Prepare a table describing design constraints, implementation constraints, and external interface requirements for the system.<br>29. Create a Requirements Traceability Matrix (RTM) linking user requirements, system requirements, use cases, and test cases.<br>30. Draw an Entity–Relationship (ER) Diagram for your selected system using a suitable tool. | | |

**Evaluation Method:**

| Sr. No. | Evaluation Methods | SEE | CCE |
|---|---|---|---|
| 1 | **SRS Preparation for Real-Life Project:** The faculty will provide a real-life problem statement. Based on the given system, students must prepare a complete Software Requirements Specification (SRS) document by identifying user needs, listing functional and non-functional requirements, drawing necessary diagrams, and formatting the SRS as per IEEE standards. | 20 | |
| 2 | **Active Learning Assignment Mini Project Risk Scenario:** Students must select their mini project and imagine adding a new feature (e.g., notifications, offline mode, biometric login). Write 4–5 sentences describing at least two risks the feature may cause (technical, schedule, or usability) and how you would handle or reduce them. Save the work as pdf and upload it to the GMIU portal. | | 10 |
| | Total | 20 | 10 |

***Examination Style:***

**SRS Preparation for Real-Life Project (20Marks)**

The faculty will provide a real-life problem statement to the students (such as a Library Management System, Online Food Ordering App, Banking Application, Attendance System, etc.). Based on the given problem, students are required to prepare a complete and structured Software Requirements Specification (SRS) document. They must analyze the system, identify user needs, list functional and non-functional requirements, draw appropriate system diagrams, and prepare the SRS according to the IEEE standard format. Students must prepare a detailed Software Requirements Specification (SRS) document based on the real-life problem statement provided by the faculty.

The SRS must include the following components:

1. **Introduction**
   - Purpose of the system
   - Scope
   - Users and stakeholders

| | | | |
|---|---|---|---|
| | 2. **Overall Description**<br>– System features overview<br>– Assumptions and constraints<br>3. **Functional Requirements**<br>– List at least five clear functional requirements<br>4. **Non-Functional Requirements**<br>– Specify at least four NFRs (performance, security, usability, reliability, etc.)<br>5. **Entity-Relationship Diagram**<br>– Draw a simple ER diagram for the selected system to show key entities, attributes, and relationships.<br>6. **Conclusion**<br>– Brief summary describing how the SRS will support system development<br><br>**Mini Project Risk Scenario (10 Marks)**<br>Each student must take their own mini project and imagine adding a new feature to the system. The feature may be something like push notifications, offline mode, biometric login, dark mode, QR-based login, voice search, or any other enhancement that improves the project. Students must then write 4–5 well-explained sentences describing at least two risks that this new feature could introduce. These risks may be:<br>• Technical Risks (e.g., integration issues, data security problems, device compatibility)<br>• Schedule Risk (e.g., feature taking longer to develop than planned, unexpected rework)<br>• Usability Risks (e.g., users getting confused, interface becoming complicated)<br>For each identified risk, students must also write how they would handle, reduce, or mitigate the risk, such as using testing strategies, simplifying design, allocating more time, or choosing alternative implementation methods. Finally, students must upload the PDF to the GMIU portal. | | |
| 3 | **Software Design and Object Modeling using UML:**<br><br>***Theory Topics:***<br><br>System/Software Design Concepts, Architectural Design, Coupling and Cohesion, Approaches to Software Design : Function-oriented Design and Object-oriented Design, Architectural Styles: Layered, MVC, Client-Server, Micro services, Component-level Design, Interface Design Principles, Secure Design Principles, Design for Scalability & Reliability, UML Diagrams: Object, class, Methods, Class relationships, Use Case diagrams, Class diagrams, Activity Diagrams, State Chart Diagrams, Sequence Diagram. Using Draw.io (diagrams.net), Microsoft Visio, https://online.visual-paradigm.com/drive/#proj=0&dashboard. | 18 | 25% |

***Practical:***

31. Illustrating key design concepts—abstraction, modularity, cohesion, and coupling—through simple visual examples to demonstrate good software design practices.
32. Creating diagrams of major software architectural styles (Layered, MVC, Client-Server, Microservices) to visually understand system organization and communication flow.
33. Developing a component-level block diagram to visually represent software modules and their input/output interfaces for clear structural design.
34. Designing simple user interface screens while applying core UI principles such as consistency, simplicity, visibility, feedback, and error prevention.
35. Applying core secure design principles through a structured checklist to ensure safe input handling, restricted access, and secure system behavior.
36. Preparing a basic scalability and reliability plan to ensure the software can handle growth and maintain consistent performance and availability.
37. Design a Class Diagram for a Student Management System to visually represent the system's classes, attributes, methods, and relationships.
38. Design a Sequence Diagram for a Student Management System to show interactions between objects for a specific use case or scenario.
39. Design an Activity Diagram for a Student Management System to model the workflow and sequence of activities for a particular process.
40. Create a Use Case Diagram for a Student Management System to visually represent the system's functionality and interactions with users.
41. Create a Class Diagram for a Student Management System to model the system's structure, including classes, attributes, methods, and relationships.
42. Create an Activity Diagram for a Student Management System to illustrate the workflow and sequence of activities for a specific process.
43. Create a Sequence Diagram for a Student Management System to depict interactions between objects for a particular use case or scenario.

**Evaluation Method:**

| Sr. No. | Evaluation Methods | SEE | CCE |
|---|---|---|---|
| 1 | **Create UML Diagram for Real-Life Project:** The faculty will provide a real-life system. Students must create Use Case, Class, Sequence, and Activity Diagrams using MS Visio. The diagrams should clearly show system actors, components, relationships, interactions, and workflow, with correct UML notations and proper labeling. | 20 | |
| 2 | **Mini Project Modular Design and Interaction Diagram** Students will divide their mini project into clear modules (e.g., user management, data storage, notifications). They must draw a simple diagram showing each module, its responsibilities, and how modules interact, ensuring high cohesion and low coupling. Add short notes explaining each module's purpose, save the work as a PDF, and upload it to the GMIU portal. | | 10 |
| | Total | 20 | 10 |

***Examination Style:***

**Create UML Diagrams for Real-Life Project (20 Marks)**

The faculty will provide each student with a real-life software system as a problem statement (such as an Online Shopping System, Hospital Management System, Banking System, Attendance System, etc.). Based on the given system, students are required to prepare **UML diagrams using MS Visio**.

Students must create **any three** from the following diagrams:

1. **Use Case Diagram** – identify system actors and illustrate their interactions with various system use cases, showing functional boundaries and user–system relationships.
2. **Class Diagram** – show the major classes, their attributes, methods, and relationships such as association, aggregation, composition, and inheritance.
3. **Sequence Diagram** – depict how objects and actors interact for a selected use case, showing the sequence of messages exchanged over time.
4. **Activity Diagram** – represent the workflow or process flow of a key system activity using actions, decision points, branches, merges, and start/end states.

| | | | |
|---|---|---|---|
| | To complete the task, students must analyze the system requirements, identify key entities, actors, processes, and interactions, and convert them into correct UML notations. All diagrams must be **clear, accurate, well-structured, and properly labeled** to reflect the functionality and workflow of the given system effectively.<br><br>**Mini Project Modular Design and Interaction Diagram (10 Marks)**<br>In this activity, each student will use their own mini project and break it down into meaningful and manageable modules. Students must first identify the major functional parts of their system—such as User Management, Data Storage, Authentication, Notifications, Reporting, Payment Processing, or any other components relevant to their project.<br>Once the modules are identified, students must create a modular design diagram that visually shows:<br>• Each module as a separate block<br>• The main responsibility or function of each module<br>• How the modules interact or communicate with one another<br>• Proper application of high cohesion (each module focuses on a single purpose)<br>• Proper application of low coupling (modules depend minimally on each other)<br>Students must also write short explanatory notes describing the purpose of each module, its responsibilities, and how it contributes to the overall system. After completing the diagram and notes, the work should be compiled into a PDF file and uploaded to the GMIU portal as instructed. | | |
| 4 | **Software Development and Software Testing:**<br>***Theory Topics:***<br>An object-oriented analysis and OOAD Methodology, Applications of the analysis and design process, OOD Goodness Criteria, Coding Standard and coding Guidelines, Code Review, Software Documentation, Testing, Testing activities, Software-Testing Strategies, Unit Testing, Black-box Testing, White-box testing, Debugging, Integration Testing, System Testing, Test Cases, Test Suites Design, Testing Conventional Application, Testing Object Oriented Applications, Testing Web and Mobile Applications, Testing Tools (Win runner, Load runner), Bug Report.<br>***Practical:***<br>44. Applying coding standards and formatting guidelines to write clean, readable, and well-structured code in a programming environment.<br>45. Reviewing a given code sample to identify issues and provide improvement comments, reinforcing best practices in readability and correctness. | 18 | 20% |

46. Preparing clear software documentation that explains program purpose, inputs/outputs, logic flow, and execution results.
47. Designing structured test cases and organizing them into a test suite to systematically validate program functionality and expected behavior.
48. Using Selenium IDE to record, replay, and validate automated user interactions on a live web application.
49. Perform unit testing on a function that calculates the total bill amount including tax and discount.
50. Design test cases for black-box testing using Equivalence Partitioning for a User Age Validation field.
51. Simulating mobile testing using an online emulator to evaluate responsiveness, layout, and usability of a website on various mobile devices.
52. Perform statement coverage testing using white-box testing for a program that checks whether a number is even or odd.
53. Perform Integration Testing on Login and Dashboard modules.
54. Perform System Testing for an Online Railway Reservation System.
55. Record and playback a test script using WinRunner for a Login Page.
56. Perform load testing for a Web Application using LoadRunner.
57. Documenting identified defects in a structured bug report with details like steps, expected vs actual results, severity, and status for proper tracking and resolution.
58. Prepare a bug report for a Payment Failure Issue in an E-commerce application.

**Evaluation Method:**

| Sr. No. | Evaluation Methods | SEE | CCE |
|---|---|---|---|
| 1 | **Positive & Negative Input Testing for a Form Field:** Students must test an assigned form field using positive, negative, and boundary values, document results in a test case table, and submit the final PDF. | 20 | |
| 2 | **Active Learning Assignment Test Case Design:** Students must select a simple mini-project module, create minimum 5 test cases using standard testing techniques, organize them in a table, save as PNG, and upload to the GMIU portal. | | 10 |
| | Total | 20 | 10 |

| | | | |
|---|---|---|---|
| | ***Examination Style:***<br><br>**Positive & Negative Input Testing for a Form Field (20 Marks)**<br>Each student will test a specific form input field assigned by the faculty—such as Email, Password, Phone Number, Age, Username, or Address—by performing Positive Testing with valid inputs, Negative Testing with invalid or unexpected inputs, and Boundary Value Testing using edge-case values. Students must check all valid, invalid, and limit-based inputs and record their observations in a structured test case table that includes the test case ID, input value, test type, expected result, actual result, and pass/fail status. The work should be documented using MS Word or Google Docs, and tables may be created using MS Excel or Google Sheets for proper formatting. After completing all tests and documenting the results, students must export the final work as a PDF file and submit it as instructed.<br><br>**Test Case Design (10 Marks)**<br>Students must select a simple function or module from their mini project—such as a login form, calculator feature, or shopping cart operation—and design at least five test cases using a variety of testing techniques, including Boundary Value Analysis, Equivalence Partitioning, Positive Testing, and Negative Testing. Each test case should clearly specify the input data, the expected output, and the actual result if the test was executed. All test cases must be neatly organized in a structured table or diagram to ensure clarity and correctness. Once completed, students should save their test case design as a PNG file and upload it to the GMIU portal as instructed. | | |
| 5 | **Project Management and Risk Analysis:**<br><br>***Theory Topics:***<br><br>Definition, need of project management, characteristics of software projects, project life cycle, The $W^5HH$ Principle, roles and responsibilities of project manager. Work Breakdown Structure (WBS), scheduling, cost estimation techniques (Line of Code, Function Point), Gantt charts, PERT/CPM, identifying project risks, Risk Projection, Risk Refinement, risk categories, risk analysis (probability & impact), risk prioritization, risk mitigation, monitoring and management, The RMMM Plan, verification & validation, quality standards (ISO, CMMI).<br><br>***Practical:***<br><br>59. Creating a hierarchical Work Breakdown Structure (WBS) to visually decompose a project into modules, submodules, and tasks for better planning and management.<br>60. Developing a project timeline by scheduling tasks and visually representing them using a Gantt chart to track progress and responsibilities. | 18 | 15% |

61. Estimating software effort and cost by calculating development size using LOC and Function Point techniques in a structured spreadsheet format.
62. Draw a Gantt chart for a software project with the following activities:
    Requirements Analysis (5 days), Design (7 days), Coding (10 days), Testing (6 days), Deployment (2 days).
63. Prepare a Gantt chart for developing a Mobile Application with at least 6 project activities and realistic durations.
64. Given a software project schedule, construct a Gantt chart and identify: Start & end dates and Overlapping activities.
65. Draw a Gantt chart showing parallel execution of activities in a Web Development Project.
66. Using a Gantt chart, explain how progress tracking is done for a software development project.
67. Construct a CPM network for the following activities and determine: Critical Path and Project Duration

| Activity | Predecessor | Duration (days) |
|---|---|---|
| A | – | 3 |
| B | A | 5 |
| C | A | 4 |
| D | B | 6 |
| E | C | 2 |
| F | D, E | 4 |

- Using CPM, calculate EST, EFT, LST, LFT, and Total Float for each activity and identify the critical activities.
- A project consists of the following activities with PERT time estimates.
- Calculate the expected time and variance for each activity and find the critical path.

| Activity | Optimistic (O) | Most Likely (M) | Pessimistic (P) |
|---|---|---|---|
| A | 2 | 4 | 8 |
| B | 1 | 2 | 3 |
| C | 3 | 5 | 9 |
| D | 4 | 6 | 10 |

- Draw a PERT network diagram and determine the expected project completion time.
- Given a PERT network, calculate the probability of completing the project within a specified deadline.

68. Constructing a PERT/CPM network to estimate task durations, identify dependencies, and determine the critical path for accurate project scheduling.
69. Identifying project risks and using a probability–impact matrix to quantify and visually categorize them as high, medium, or low severity.

70. A university is developing a Student Management System. Identify any five project risks involved in the development of this system and classify them into appropriate risk categories.
71. Given three risks in a Student Management System—Requirement changes, Data security breach, Staff turnover. Explain how these risks can be prioritized using probability–impact analysis.
72. Designing a structured QA checklist to systematically verify that each project development step meets quality standards and is properly documented.
73. Prepare a brief RMMM (Risk Mitigation, Monitoring, and Management) Plan for the risk *"data loss due to system failure"* in a Student Management System.
74. Explain the role of Verification and Validation (V&V) in ensuring the quality of a Student Management System. Provide suitable examples.
75. Conducting a review-based Verification & Validation process to ensure software documents are correct, complete, consistent, and aligned with user requirements.

**Evaluation Method:**

| Sr. No. | Evaluation Methods | SEE | CCE |
|---|---|---|---|
| 1 | **Work Breakdown Structure (WBS):** Students will take the project topic given by the faculty, break it into major modules, and further divide each module into subtasks like UI Design, Coding, Testing, and Documentation. They must create a clear hierarchical WBS diagram using Draw.io, PowerPoint, or Google Slides, showing the project, modules, and subtasks. The final WBS must be saved as a PDF. | 20 | |
| 2 | **Comparative Study of Project Management Techniques** Students must compare two project management techniques (e.g., Waterfall vs Agile or Gantt Chart vs PERT/CPM), prepare a 1–2 page report with overview, pros/cons, comparison, and recommendation, save it as a PDF, and upload to the GMIU portal. | | 10 |
| | Total | 20 | 10 |

| | | | |
|---|---|---|---|
| | *Examination Style:*<br><br>**Work Breakdown Structure (WBS) (20 Marks)**<br>Students will receive a project topic from the faculty and must analyze it to identify the major modules of the system. Each module should then be broken down into smaller subtasks such as UI Design, Coding, Testing, and Documentation. Using tools like Draw.io, MS PowerPoint, or Google Slides, students must create a well-organized hierarchical WBS diagram, with the overall project at the top level, the main modules at the second level, and the subtasks at the third level. The structure should be logical, clearly labeled, and reflect proper project planning, with task assignments added if required. After completing the WBS diagram, students must export or save it as a PDF and submit it before the exam time ends.<br><br>**Comparative Study of Project Management Techniques (10 Marks)**<br>Students must research any two software project management techniques—for example, Waterfall vs Agile/Scrum or Gantt Chart vs PERT/CPM—using credible sources such as textbooks, research papers, and reliable online articles. They are required to prepare a 1–2-page report that includes:<br>1. an overview of each technique and how it works,<br>2. advantages and disadvantages of both techniques,<br>3. a comparison based on project scheduling, cost estimation, risk management, and quality assurance, and<br>4. a recommendation explaining which technique is more suitable for small, medium, or large software projects. The final report must be compiled into a PDF file and uploaded to the GMIU portal. | | |

**Suggested Specification table with Marks (Theory): 100**

| Distribution of Marks (Revised Bloom's Taxonomy) | | | | | | |
|---|---|---|---|---|---|---|
| Level | Remembrance (R) | Understanding (U) | Application (A) | Analyze (N) | Evaluate (E) | Create (C) |
| Weightage % | 15% | 15% | 20% | 15% | 15% | 20% |

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from the above table.

## Course Outcome:

| After learning the course, the students should be able to: | |
|---|---|
| CO1 | Understand the software engineering concepts, processes, quality, and project management to develop and manage software effectively. |
| CO2 | Elicit, analyze, document, validate, and manage software requirements using standard techniques, tools, and models like SRS and ER diagrams. |
| CO3 | Design software systems using design principles, UML diagrams, and modeling tools. |
| CO4 | Perform object-oriented analysis and design, follow coding standards, and apply various software testing strategies and tools to ensure software quality. |
| CO5 | Plan, estimate, schedule, monitor, and manage software projects effectively, including risk management, quality assurance, and adherence to standards. |

## Instructional Method:

The course delivery method will depend upon the requirement of content and needs of students. The teacher, in addition to conventional teaching methods by black board, may also use any tools such as demonstration, role play, Quiz, brainstorming, MOOCs etc.

From the content 10% topics are suggested for flipped mode instruction.

Students will use supplementary resources such as online videos, NPTEL/SWAYAM videos, e-courses, Virtual Laboratory.

The internal evaluation will be done on the basis of the Active Learning Assignment.

Practical/Viva examination will be conducted at the end of semester for evaluation of performance of students in the laboratory.

## Reference Books:

[1] Software Engineering: A Practitioner's Approach by Roger S. Pressman, McGraw-Hill.

[2] Fundamentals of Software Engineering by Rajib Mall, Prentice Hall of India.

[3] Software Engineering 10e- Ian Sommerville, Pearson education Asia.

[4] Software Engineering & Testing – B. B. Agarwal, S. P. Tayal, M. Gupta, Jones and Bartlett Publishers, Firewal Media, An Imprint of Laxmi Publications Pvt. Ltd.

[5] Software Engineering: An Engineering Approach by James S. Peters and Witold Pedrycz, Wiley India.

## Suggested Assessment Guidelines:

### Module-1: Introduction to Software Engineering

| Software Process Analysis and Prototype Simulation (20 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Software Categorization & SDLC Representation | Correctly identifies the software type and purpose; creates a clear, accurate visual of SDLC phases with proper labeling and completeness. | 5 |
| Comparison of Software Process Models | Well-structured comparison table showing steps, usage scenarios, strengths, and limitations of two process models accurately and clearly. | 5 |
| Prototype / Workflow Simulation | Simple, logical prototype or workflow of one module; clear steps, correct flow, proper diagramming, and neat presentation. | 5 |
| SQA & Risk Tables (Optional) + Overall Document Quality | Includes basic SQA and risk tables (if added); PDF is well-organized, clearly formatted, properly labeled, and professionally presented. | 5 |
| | Total | 20 |

### Module-2: Requirements Analysis and Software Requirements Specification

| SRS Preparation for Real-Life Project (20Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| SRS Structure & Introduction | Assesses the clarity and completeness of the Introduction section, including Purpose, Scope, and identification of Users/Stakeholders, following IEEE format. | 5 |
| Overall Description & Requirements | Evaluates the accuracy of System Overview, Assumptions, and Constraints, along with the correctness and completeness of Functional and Non-Functional Requirements. | 5 |
| System Diagrams (ER Diagram) | Checks the correctness, clarity, and labeling of the ER Diagram, showing key entities, attributes, and relationships aligned with the system requirements. | 5 |
| Documentation Quality & Conclusion | Reviews formatting, organization, grammar, IEEE-compliant structure, and quality of the Conclusion explaining how the SRS aids system development. | 5 |
| | Total | 20 |

## Module-3: Software Design and Object Modeling using UML

| Create UML Diagram for Real-Life Project (20 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Use Case Diagram Quality | Evaluates correct identification of actors and use cases, clarity of interactions, proper system boundary, and accurate UML notation. | 5 |
| Class Diagram Accuracy | Assesses correctness of classes, attributes, methods, and relationships (association, aggregation, inheritance, composition) and proper labeling. | 5 |
| Sequence Diagram Clarity | Checks accuracy of message flow, lifelines, actor/object interactions, and correct sequence representation for the chosen use case. | 5 |
| Activity Diagram & Overall Presentation | Reviews workflow correctness, decision points, flow clarity, start/end states, diagram neatness, labeling, and overall documentation quality. | 5 |
| | Total | 20 |

## Module-4: Software Development and Software Testing

| Positive & Negative Input Testing for a Form Field (20 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Positive, Negative & Boundary Test Coverage | Assesses whether students tested all required valid, invalid, and edge-case inputs for the assigned form field. | 5 |
| Accuracy of Test Case Table | Evaluates correctness of inputs, expected results, actual results, and pass/fail status recorded in a structured and clear table format. | 5 |
| Quality of Analysis & Observations | Checks whether the student's testing observations are meaningful, accurate, and logically interpreted based on the results. | 5 |
| Documentation & Formatting Quality | Reviews clarity, neatness, formatting, proper labeling, use of Word/Docs or Excel/Sheets, and correctness of the final PDF submission. | 5 |
| | Total | 20 |

## Module-5: Project Management and Risk Analysis

| Work Breakdown Structure (WBS) (20 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Identification of Modules | Evaluates how accurately and appropriately the student identifies major modules from the given project topic. | 5 |
| Breakdown into Subtasks | Checks the correctness, clarity, and completeness of dividing each module into meaningful subtasks such as UI, Coding, Testing, Documentation, etc. | 5 |
| WBS Diagram Structure & Hierarchy | Assesses the quality of the WBS diagram, including proper top–middle–lower hierarchy, logical task flow, clear labeling, and neat organization. | 5 |
| Presentation & PDF Submission Quality | Reviews neatness, tool usage (Draw.io/PowerPoint/Slides), formatting, clarity, and correctness of the final PDF submission. | 5 |
| | Total | 20 |