Gyanmanjari Innovative University

Syllabus
Gyanmanjari Institute of Technology
Semester-5 (B. Tech.)

Subject: Software Engineering: Principles, Design, and Development – BETICE15329

Type of course: Professional Elective Course

Prerequisite: Basic knowledge of programming, object-oriented concepts, data structures, and database fundamentals.

Rationale:

This course provides a comprehensive understanding of modern software engineering principles, methodologies, and practices required to develop high-quality, scalable, secure, and reliable software systems. It introduces students to software development processes, Agile methodologies, requirements engineering, software architecture, UML modeling, quality engineering, software testing, and project management. The course emphasizes industry-oriented practices such as AI-assisted software development, AI-assisted requirements specification, modern architectural styles, secure software design, test-driven development, automated testing, and Agile project management using contemporary tools. Students gain practical experience in analyzing user requirements, preparing Software Requirements Specifications (SRS), designing software architectures, creating UML models, applying coding standards, performing software testing, and managing software projects with effective risk management techniques. By integrating modern development practices, collaboration tools, and quality assurance techniques, the course prepares students to design, develop, test, and manage software solutions that meet industry standards and evolving technological demands while fostering teamwork, ethical practices, and continuous improvement throughout the software development lifecycle.

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 and Agile Development:** ***Theory Topics:*** Software Concept, Characteristics and Types, Software Application Domains, Definition of Software Engineering, Evolution of Software Engineering, The Software Process, Software Quality Attributes, Software Development Life Cycle (SDLC) Overview, 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), Scrum Framework, Agile Project Management Tools (JIRA, Trello), AI-Assisted Software Development (GitHub Copilot, ChatGPT), Ethics in AI-Assisted Software Development. ***Practical:*** 1. Create a concept map illustrating software characteristics, types, and application domains using Canva or PowerPoint. 2. Prepare a comparative table of software types and application domains with real-world examples using MS Excel. 3. Develop a timeline showing the evolution of Software Engineering and major milestones using PowerPoint or Canva. 4. Create an SDLC workflow diagram showing all phases of software development using Lucidchart, Draw.io (diagrams.net), Canva, or Miro. 5. Compare Software Process Models (Waterfall, Incremental, Spiral, Prototyping, Agile) based on phases, advantages, disadvantages, and suitable applications using MS Excel or MS Word. 6. Prepare a case study report comparing traditional and Agile software development approaches for a real-world project using MS Word. 7. Create a Scrum Board with Product Backlog, Sprint Backlog, To-Do, In Progress, and Done columns for a mini-project using Jira or MS Excel. 8. Prepare a Sprint Burndown Chart for a hypothetical two-week sprint using MS Excel. 9. Design a paper prototype or low-fidelity UI prototype for a simple application using Canva Whiteboard or paper-based design. 10. Generate software project documentation (Project Overview, Features, User Stories, and Sprint Goals) using GitHub Copilot or ChatGPT, review the AI-generated content, and refine it in MS Word.	18	20%

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Software Process Modeling and Agile Planning:** Students will analyze a given software application, prepare the SDLC workflow, compare software process models, create a Scrum Board, and generate a brief AI-assisted project overview. Submit the work as a PDF.	20	
2	**Active Learning Assignment** **Agile Sprint Planning for a Mini Project:** Students must create User Stories, a Product Backlog, and a Sprint Backlog for a mini-project and submit the work as a PDF.		10
	Total	20	10

Examination Style:

Software Process Modeling and Agile Planning (20 Marks)
The faculty will provide a real-world software application (e.g., Library Management System, Hospital Management System, Online Food Ordering System, etc.). Students must analyze the given application, identify its software type and application domain, prepare the SDLC workflow, compare suitable software process models, create a Scrum Board with Product Backlog and Sprint Backlog, and generate a simple AI-assisted project overview using GitHub Copilot or ChatGPT. The complete work should be submitted as a single PDF.

Agile Sprint Planning for a Mini Project (10 Marks)
Students must select a mini-project of their choice and prepare at least 8–10 User Stories, prioritize them into a Product Backlog, create a Sprint Backlog for the first sprint, and write a short reflection (100–150 words) on how AI tools (GitHub Copilot or ChatGPT) can assist during software development. Save the document as a PDF and upload it to the GMIU Portal.

2	**Requirements Engineering and Software Requirements Specification:** ***Theory Topics:*** Requirement Engineering, Requirements Engineering Process, Stakeholder Identification and Analysis, Requirements Gathering (Document Analysis, Interview, Questionnaire, Observation), Requirements Analysis, Software Requirements Specifications (SRS) Document, Format, IEEE Standards for SRS Documents, SRS Validation, Components of SRS, Characteristics of SRS, AI-assisted SRS Generation	18	20%

(GitHub Copilot, ChatGPT). Types of Requirements: Functional, Non-functional, design and implementation constraints, external interfaces required. Other non-functional requirements, Requirements Validation & Verification, Requirements Traceability Matrix, Requirement Elicitation.

Practical:

11. Create a flow diagram of the Requirements Engineering Process using Draw.io, Lucidchart, or Canva.
12. Identify stakeholders and prepare a Stakeholder Analysis Matrix for a selected software system using MS Excel or MS Word.
13. Prepare a Requirement Elicitation Report using the interview technique for a selected software project in MS Word.
14. Design a Questionnaire (10–15 questions) for gathering software requirements using MS Word or Google Forms.
15. Conduct Requirement Gathering through observation or document analysis and prepare a summary report in MS Word.
16. Classify the collected requirements into Functional and Non-Functional Requirements using MS Word or MS Excel.
17. Prepare a Software Requirements Specification (SRS) document in IEEE format for a selected software project using MS Word.
18. Generate an initial SRS draft using GitHub Copilot or ChatGPT and refine it according to IEEE standards in MS Word.
19. Prepare a Requirements Traceability Matrix (RTM) linking requirements with test cases using MS Excel.
20. Validate the prepared SRS document using an SRS validation checklist and document the findings in MS Word.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Software Requirements Specification (SRS) Preparation:** Students must select a mini-project, identify functional and non-functional requirements, prepare a Software Requirements Specification (SRS) in IEEE format, and validate it using an SRS checklist. The completed document should be saved as a PDF and uploaded to the GMIU Portal.	20	
2	**Active Learning Assignment Requirements Traceability Matrix Preparation:** Students must prepare a Requirements Traceability Matrix (RTM) linking		10

		functional requirements with corresponding test cases for a selected mini-project. Save the document as a PDF and upload it to the GMIU Portal.			
		Total	20	10	

Examination Style:

Software Requirements Specification (SRS) Preparation (20 Marks)
Each student must select a mini-project such as a Library Management System, Student Management System, Online Shopping System, Hospital Management System, or any similar application. Students are required to analyse the project requirements, identify stakeholders, and prepare a Software Requirements Specification (SRS) document following the IEEE standard format. The SRS should include the project overview, functional and non-functional requirements, design and implementation constraints, external interface requirements, and other essential components. Students must validate the prepared SRS using an SRS validation checklist to ensure completeness, consistency, correctness, and clarity. The final document should be compiled into a PDF file and submit it.

Requirements Traceability Matrix (RTM) Preparation (10 Marks)
Each student must select a mini-project such as a Library Management System, Student Management System, Online Shopping System, Hospital Management System, or any similar application. Students are required to identify the functional requirements of the selected project and prepare a Requirements Traceability Matrix (RTM) by linking each requirement to its corresponding use case and test case. The RTM should clearly demonstrate how every requirement is tracked throughout the software development and testing process, ensuring complete coverage and traceability. Students should verify that all requirements are uniquely identified and properly mapped. The completed RTM should be prepared using MS Excel or MS Word, saved as a PDF file, and uploaded to the GMIU Portal within the specified deadline.

3	**Software Design, Architecture, and UML Modeling:** ***Theory Topics:*** System Design Concepts, Architectural Design, Coupling and Cohesion, Secure Software Design, Approaches to Software Design : Function-oriented Design and Object-oriented Design, Architectural Styles: Layered, MVC, Client-Server, Micro services, Cloud-Native Architecture, 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, Package Diagram, Deployment Diagram. Using	18	25%

	Draw.io (diagrams.net), Microsoft Visio, https://online.visual-paradigm.com, Lucid Chart, AI-assisted UML Diagram Generation. ***Practical:*** 21. Illustrate System Design Concepts such as abstraction, modularity, coupling, and cohesion for a Library Management System using MS Word or PowerPoint. 22. Compare Function-Oriented Design and Object-Oriented Design for a Student Management System using MS Word or MS Excel. 23. Create architectural diagrams (Layered, MVC, Client-Server, Microservices, and Cloud-Native) for an Online Shopping System using Draw.io, Lucidchart, or Visual Paradigm. 24. Design a Component Diagram for a Hospital Management System using Draw.io or Microsoft Visio. 25. Design user interface screens for an Online Food Ordering System by applying Interface Design Principles using Canva, Figma, or PowerPoint. 26. Prepare a Secure Software Design Checklist for an Internet Banking System using MS Word. 27. Create a Use Case Diagram for a Library Management System using Draw.io, Visual Paradigm, or Lucidchart. 28. Design a Class Diagram for a Student Management System showing classes, attributes, methods, and relationships using Visual Paradigm or Draw.io. 29. Create an Activity Diagram for the Online Course Registration System representing the student registration workflow using Draw.io or Lucidchart. 30. Design a Sequence Diagram for the Online Shopping System to illustrate the order placement process using Visual Paradigm or Draw.io. 31. Create a Package Diagram for a Hospital Management System to organize software modules using Visual Paradigm or Draw.io. 32. Design a Deployment Diagram for an E-Commerce System showing the client, web server, application server, and database server using Draw.io or Microsoft Visio. 33. Generate a Use Case Diagram for an Online Examination System using GitHub Copilot or ChatGPT, refine it using Visual Paradigm or Draw.io, and document the improvements in MS Word.		

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	**Active Learning Assignment Software Architecture Selection:** Students must select a mini-project, choose an appropriate software architecture, justify their selection, and prepare a high-level architecture diagram. 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. **Software Architecture Selection (10 Marks)** Each student must select a mini-project such as a Library Management System, Student Management System, Hospital Management System, Online Shopping System, or any similar application. Students are required to analyze the functional and non-functional requirements of the selected project and identify the most suitable software architecture, such as Layered, MVC, Client-Server, Microservices, or Cloud-Native Architecture. They must justify their architecture selection by explaining how it satisfies the project's scalability, maintainability, security, and performance requirements. Students should then prepare a high-level software architecture diagram showing the major components, their interactions, and data flow using Draw.io, Visual Paradigm, Lucidchart, or Microsoft Visio. The completed work should be compiled into a PDF file and uploaded to the GMIU Portal within the specified deadline.		
4	**Software Quality Engineering and Software Testing:** ***Theory Topics:*** An object-oriented analysis and OOAD Methodology, Applications of the analysis and design process, Coding Standard and coding Guidelines, Code Review, Software Documentation, Test Driven Development (TDD), Behavior-Driven Development (BDD), Testing, Software-Testing Strategies, Unit Testing, Black-box Testing, White-box testing, Integration Testing, System Testing, Selenium Testing, Performance Testing, Acceptance Testing, Test Cases, Test Suites Design, Testing Conventional Application, Testing Object Oriented Applications, Testing Web and Mobile Applications, Test Automation. ***Practical:*** 34. Apply coding standards and coding guidelines to a Student Management System program and document the improvements using MS Word. 35. Perform a code review for a Library Management System module and prepare a review report highlighting coding issues and recommendations using MS Word. 36. Prepare software documentation (Program Description, Inputs, Outputs, and User Guide) for a Hospital Management System using MS Word. 37. Develop Unit Test Cases for the Login Module of an Online Shopping System using JUnit/PyTest or MS Word. 38. Design Black-box Test Cases using Equivalence Partitioning and Boundary Value Analysis for the Student Registration Form of a Student Management System.	18	20%

39. Perform White-box Testing using Statement Coverage for the Password Validation Module of an Online Banking System.
40. Perform Integration Testing for the Login and Dashboard Modules of a Library Management System and document the results.
41. Perform System Testing for an Online Food Ordering System by preparing and executing a system test report.
42. Automate the Login Process of an E-Commerce Website using Selenium IDE/WebDriver.
43. Prepare Acceptance Test Cases for the Course Registration Module of an Online Course Registration System.
44. Design a Test Suite for the Online Railway Reservation System covering functional and non-functional requirements.
45. Perform Performance Testing for a Hospital Management System using an appropriate testing tool and record the observations.
46. Generate Test Cases for the Library Management System using GitHub Copilot or ChatGPT, verify their correctness, refine them manually, and document the improvements in MS Word.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Test Suite Design and Execution:** Students must design a complete test suite for a given software application, execute the selected test cases, document the expected and actual results, prepare a summary report, and submit the work as a PDF.	20	
2	**Code Review and Test Case Design:** Students must select a mini-project, perform a code review for one module, prepare functional test cases, document the observations and test results, and submit the work as a PDF to the GMIU Portal.		10
	Total	20	10

Examination Style:

Test Suite Design and Execution (20 Marks)

The faculty will provide a software application or one functional module such as a Library Management System, Student Management System, Online Shopping System, or Hospital Management System. Students are required to analyze the given module and prepare a comprehensive test suite by designing appropriate test cases using suitable software testing techniques, such as Black-box Testing, White-box Testing, Boundary Value Analysis, or Equivalence Partitioning. Students must execute the selected test cases, record the expected and actual results, and determine

	the pass/fail status for each test case. Any defects identified during testing should be documented in a structured bug report with appropriate severity and priority. Finally, students must prepare a summary report describing the testing strategy, test execution results, identified defects, and overall software quality. The completed work should be compiled into a single PDF file and submit it. **Code Review and Test Case Design (10 Marks)** Each student must select a mini-project such as a Library Management System, Student Management System, Online Shopping System, or Hospital Management System. Students are required to select one functional module (e.g., Login, Registration, Payment, or Course Enrollment) and perform a code review by checking coding standards, readability, naming conventions, and documentation. Based on the reviewed module, students must design at least five functional test cases using appropriate software testing techniques and document the expected and actual results. Any identified defects or improvement suggestions should also be recorded in a structured report. The completed report should be compiled into a PDF file and uploaded to the GMIU Portal within the specified deadline.		
5	**Software Project Management and Risk Management:** ***Theory Topics:*** Definition, need of project management, characteristics of software projects, project life cycle, roles and responsibilities of project manager. Work Breakdown Structure (WBS), scheduling, Gantt charts, PERT/CPM, identifying project risks, Risk Projection, Risk Refinement, risk categories, risk analysis, risk prioritization, risk mitigation, monitoring and management, Risk Register, The RMMM Plan, Agile Project Management, Scrum Project Planning. ***Practical:*** 47. Prepare a Project Charter for a Library Management System including project objectives, scope, stakeholders, and deliverables using MS Word. 48. Create a Work Breakdown Structure (WBS) for an Online Shopping System using Draw.io, Lucidchart, or PowerPoint. 49. Develop a Gantt Chart for a Hospital Management System project showing project activities, durations, and dependencies using MS Excel or Microsoft Project. 50. 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).	18	15%

51. Construct a PERT/CPM Network Diagram for a Student Management System and determine the critical path using MS Excel or Draw.io.
52. 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.

53. Identify and classify project risks for an Online Food Ordering System into technical, schedule, cost, and operational risks using MS Word.
54. Prepare a Risk Register for an E-Commerce System including risk description, probability, impact, owner, and mitigation strategy using MS Excel.
55. Perform Risk Analysis and Prioritization for a Hospital Management System using a Probability–Impact Matrix in MS Excel.
56. Prepare an RMMM (Risk Mitigation, Monitoring, and Management) Plan for a Library Management System using MS Word.
57. Compare Traditional Project Management and Agile Project Management for an Online Shopping System by preparing a comparative report in MS Word.
58. Prepare a Project Schedule and Milestone Chart for a Hospital Management System using MS Excel or PowerPoint.
59. Prepare a Project Status Report for an Online Course Registration System summarizing completed tasks, pending activities, project risks, and next sprint goals using MS Word.

60. Prepare a Sprint Backlog and Sprint Plan for the first sprint of a Student Management System using Jira or MS Excel.

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:

Work Breakdown Structure (WBS) (20 Marks)
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.

Comparative Study of Project Management Techniques (10 Marks)
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: 1. an overview of each technique and how it works, 2. advantages and disadvantages of both techniques, 3. a comparison based on project scheduling, cost estimation, risk management, and quality assurance, and 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	Apply software engineering principles, SDLC models, Agile practices, and AI-assisted development tools in software development.
CO2	Analyze, document, validate, and manage software requirements using requirement engineering techniques, SRS, and requirements traceability.
CO3	Design software architecture and UML models by applying software design principles and object-oriented design concepts.
CO4	Evaluate software quality using testing strategies, code review, and test automation techniques.
CO5	Plan and manage software projects using project management and risk management methodologies.

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 and Agile Development

Software Process Modeling and Agile Planning (20 Marks)		
Criteria	**Description**	**Marks**
Software Analysis & SDLC Workflow	Correctly analyzes the given software application and prepares a clear, complete SDLC workflow with appropriate phases and logical flow.	5
Software Process Model Comparison	Accurately compares two software process models (e.g., Waterfall, Spiral, Incremental, Prototype, Agile) based on phases, advantages, limitations, and suitable applications.	5
Scrum Board Preparation	Creates a well-organized Scrum Board with Product Backlog, Sprint Backlog, and task status (To Do, In Progress, Done) appropriate to the selected application.	5
AI-Assisted Project Overview & Report Quality	Generates a concise AI-assisted project overview using GitHub Copilot or ChatGPT, reviews the generated content, and submits a well-organized, properly formatted PDF report.	5
	Total	20

Module-2: Requirements Engineering and Software Requirements Specification

Software Requirements Specification (SRS) Preparation (20Marks)		
Criteria	**Description**	**Marks**
Requirement Identification and Analysis	Correctly identifies stakeholders and analyzes functional and non-functional requirements for the selected mini-project.	5
SRS Preparation (IEEE Format)	Prepares a well-structured Software Requirements Specification (SRS) following the IEEE standard with all essential sections.	5
Requirements Validation and Traceability	Validates the SRS using an SRS checklist and prepares a Requirements Traceability Matrix (RTM) linking requirements appropriately.	5
Document Organization and Presentation	Presents a complete, well-formatted, and properly documented SRS report with clear organization, consistency, and professional presentation.	5
	Total	20

Module-3: Software Design, Architecture, and UML Modeling

Create UML Diagram for Real-Life Project (20 Marks)		
Criteria	**Description**	**Marks**
System Analysis and Use Case Diagram	Correctly analyzes the given system, identifies actors and use cases, and prepares a complete Use Case Diagram using appropriate UML notations.	5
Class Diagram Design	Develops a well-structured Class Diagram showing classes, attributes, methods, relationships, and appropriate object-oriented design concepts.	5
Sequence and Activity Diagrams	Creates accurate Sequence and Activity Diagrams representing system interactions and workflow with proper UML notations and logical flow.	5
Diagram Quality and Documentation	Diagrams are complete, properly labeled, well-organized, use standard UML notations, and are professionally presented in a single PDF.	5
	Total	20

Module-4: Software Quality Engineering and Software Testing

Test Suite Design and Execution (20 Marks)		
Criteria	**Description**	**Marks**
Test Suite Design	Designs a comprehensive test suite with appropriate test cases covering functional requirements using suitable testing techniques.	5
Test Execution and Results	Executes the selected test cases accurately and records the expected results, actual results, and pass/fail status.	5
Defect Identification and Reporting	Identifies software defects, prepares a structured bug report with severity and priority, and suggests appropriate improvements.	5
Summary Report and Documentation	Prepares a well-organized testing summary report with complete documentation, proper formatting, and professional presentation in a single PDF.	5
	Total	20

Module-5: Software Project Management and Risk Management

Work Breakdown Structure (WBS) (20 Marks)		
Criteria	**Description**	**Marks**
Project Analysis and Module Identification	Correctly analyzes the given project and identifies major modules with appropriate project scope and deliverables.	5
Work Breakdown Structure (WBS) Design	Develops a clear hierarchical WBS showing the project, modules, tasks, and subtasks with proper decomposition.	5
Task Organization and Completeness	Logically organizes activities such as UI Design, Coding, Testing, Documentation, and other relevant subtasks with appropriate sequencing.	5
Diagram Quality and Documentation	WBS diagram is neat, properly labeled, professionally formatted, and submitted as a well-organized PDF using Draw.io, PowerPoint, or Google Slides.	5
	Total	20