SCHEME :K

Name :____________________________
Roll No.: ______________Year:20__20__
Exam Seat No. :______________________

LABORATORY MANUAL FOR

SOFTWARE ENGINEERING (315323)

Requirement Analysis
Design
Implementation
Testing
Evolution
SDLC
Software/System Development Life Cycle - SDLC
CODE PROGRAMMING
APPLICATION DEVELOPPMENT
SYSTEM ALGORITHM LANGUAGE
TESTING REQUIREMENTS PROCESS
AGILITY SOFTWARE SCRIPT
QUALITY PLANNING
ANALYSIS ENGINEERING AGILE
SCRIPT FUNCTION
TECHNOLOGY SCRUM TECHNOLOGY DATABASE
COMPUTER IMPLEMENTATION FUNCTION
ARCHITECTURE COMPUTER DEBUGGING
FUNCTION DEBUGGING ANALYSIS

COMPUTER ENGINEERING GROUP

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI
(Autonomous)(ISO21001:2018)(ISO/IEC27001:2013)

Vision

To ensure that the Diploma Level Technical Education constantly matches the latest requirements of technology industry and includes the all-round personal development of students including social concerns and to become globally competitive, technology led organization.

Mission

To provide high quality technical and managerial manpower, information and consultancy services to the industry and community to enable the industry and community to face the changing technological & environmental challenges.

Quality Policy

We, at MSBTE are committed to offer the best in class academic services to the students and institutes to enhance the delight of industry and society. This will be achieved through continual improvement in management practices adopted in the process of curriculum design, development, implementation, evaluation and monitoring system along with adequate faculty development programs.

Core Values

MSBTE believes in the following:

- Education industry produces live products,
- Market requirements do not wait for curriculum changes.
- Question paper is the reflector of academic standards of educational organization.
- Well-designed curriculum needs effective implementation too.
- Competency based curriculum is the backbone of need based program.
- Technical skills do need support of life skills,
- Best teachers are the national assets.
- Effective teaching learning process is impossible without learning resources.

A Laboratory Manual for

SOFTWARE ENGINEERING (315323)

Semester-V

'K' Scheme

(CM/ CO/ CW/ HA/ IH/ SE)

Maharashtra State
Board of Technical Education, Mumbai

(Autonomous) (ISO 21001:2018)(ISO/IEC 27001:2013)

Maharashtra State
Board of Technical Education, Mumbai
(Autonomous) (ISO 21001:2018)(ISO/IEC 27001:2013)
4th Floor, Government Polytechnic Building, 49, Kherwadi,
Bandra (East), Mumbai -400051.
(Printed On July, 2025)

Maharashtra State

Board of Technical Education

Certificate

This is to certify that Mr. / Ms .... Roll No. …………., of Fifth Semester of Diploma in……………………..……… ……………………………………… of Institute ……………………..…...... …………………………. (Institute Code : ………………...) has completed the term work satisfactorily in course **Software Engineering (315323)** for the academic year 20………. to 20.. as prescribed in the curriculum.

Place: ……………	Enrollment No. : ………………
Date: …………….	Exam. Seat No: ……………...

Subject Teacher **Head of the Department** **Principal**

Seal of Institution

Preface

The primary focus of any engineering laboratory/field work in the technical education system is to develop the much needed industry relevant competencies and skills. With this in view, MSBTE embarked on this innovative ‘K’ Scheme curricula for engineering diploma programs with outcome- based education as the focus and accordingly, relatively large amount of time is allotted for the practical work. This displays the great importance of laboratory work making each teacher, instructor and student to realize that every minute of the laboratory time need to be effectively utilized to develop these outcomes, rather than doing other mundane activities. Therefore, for the successful implementation of this outcome-based curriculum, every practical has been designed to serve as a ***‘vehicle’*** to develop this industry identified competency in every student. The practical skills are difficult to develop through ‘Chalk and duster’ activity in the classroom situation. Accordingly, the ‘K’ scheme laboratory manual development team designed the practical to focus on the outcomes, rather than the traditional age old practice of conducting practical to ‘verify the theory’ (which may become a byproduct along the way).

This laboratory manual is designed to help all stakeholders, especially the students, teachers and instructors to develop in the student the pre-determined outcomes. It is expected from each student that at least a day in advance, they have to thoroughly read through the concerned practical procedure that they will do the next day and understand the minimum theoretical background associated with the practical. Every practical in this manual begins by identifying the competency, industry relevant skills, course outcomes and practical outcomes which serve as a key focal point for doing the practical. The students will then become aware about the skills they will achieve through procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This manual also provides guidelines to teachers and instructors to effectively facilitate student- centered lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students.

Learning Software Engineering is essential for students as it equips them with the skills to design, develop, and maintain reliable and efficient software systems. By mastering software engineering principles and practices, students can enhance their ability to solve real-world problems, collaborate in development teams, and deliver high- quality software solutions that meet user needs. These skills not only expand career opportunities in areas such as application development, systems design, and quality assurance but also foster critical thinking, problem-solving, and innovation in the software development lifecycle.

Although the best possible care has been taken to check for errors (if any) in this laboratory manual, perfection may elude us as this is the first edition of this manual. Any errors and suggestions for improvement are solicited and highly welcome.

Lab Manual Development Team

Program Outcomes (POs) to be achieved through the Practical of this course

PO1. Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

PO2. Problem analysis: Identify and analyze well-defined engineering problems using codified standard methods.

PO3. Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4. Engineering Tools, Experimentation, and Testing: Apply modern engineering tools and appropriate techniques to conduct standard tests and measurements.

PO5. Engineering practices for society, sustainability, and environment: Apply appropriate technology in the context of society, sustainability, environment, and ethical practices.

PO6. Project Management: Use engineering management principles individually, as a team member, or as a leader to manage projects and effectively communicate about well-defined engineering activities.

PO7. Life-long learning: Ability to analyze individual needs and engage in updating in the context of technological changes.

Practical- Course Outcome matrix

Course Outcomes (COs)

CO1 - Select suitable software development process model.

CO2 - Prepare software requirement specification.

CO3 - Construct different Software design models.

CO4 - Apply different planning and cost estimation techniques for a software product.

CO5 - Apply project management techniques in software development.

CO6 - Use quality assurance principles in software development.

Sr. No.	Title of the Practical	CO1	CO2	CO3	CO4	CO5	CO6
1	*Write problem statement to define the project title with bounded scope of the project.	√					
2	Select relevant process model to define activities and related tasks set.	√					
3	*Gather application specific requirements for assimilate into RE (Requirement's engineering) Model.		√				
4	*Prepare broad SRS (software requirement Specification) for the project.		√				
5	*Write use-cases and draw use-case diagram.			√			
6	Draw the activity diagram to represent flow from one activity to another for software development.			√			
7	*Create DFDs (data flow diagram), Decision tables and E-R (entity-relationship) diagram.			√			
8	Draw class diagram and Sequence diagram, State Transition Diagram.			√			
9	* Create decision table for a project.			√			
10	*Write test cases to validate requirements from SRS document.			√			
11	Prepare test cases for Black Box Testing.			√			
12	* Identify risks involved in the project and prepare RMMM (RMMM-Risk Management, Mitigation and Monitoring) plan.				√		
13	* Calculate size of the project using Function point metric.				√		
14	*Calculate cost of the project using COCOMO (Constructive Cost Model) / COCOMO II approach.				√		
15	*Create software project scheduling charts using CPM (Critical Path Method) / PERT (Project Evaluation and Review Technique)					√	
16	Track progress of the project using Timeline Charts/ Gantt charts.					√	

17	Prepare SQA plan that facilitates various Attributes of quality of process.						√
18	*Prepare SQA plan that facilitates various attributes of quality of product.						√

Industry / Employer Expected Outcome

The aim of this course is to help the student to attain the following industry identified outcomes through various teaching learning experiences:

1. Apply SDLC models for structured software development.
2. Analyze and document software requirements effectively.
3. Design modular and maintainable software architectures.
4. Develop reliable and scalable software solutions.
5. Create and execute test cases to ensure software quality.
6. Use version control tools for collaborative development.
7. Apply project management techniques in software projects.
8. Follow coding standards and documentation practices.
9. Understand ethical and legal aspects of software engineering.
10. Communicate technical concepts clearly and professionally.

Guidelines to Teachers

1. For incidental writing on the day of each practical session every student should maintain a dated logbook for the whole semester, apart from this laboratory manual, which s/he has to submit for assessment to the teacher in the next practical session.
2. Teachers should give opportunity to students for hands-on after the demonstration.
3. Assess the skill achievement of the students and COs of each unit.
4. Explain prior concepts to the students before starting of each experiment.
5. List of few sample questions for reference are given. Teachers must design more such questions so as to ensure the achievement of identified CO.
6. Teacher should ensure that the practical skill and competencies are developed in the students after the completion of the practical exercise.
7. Teacher may provide additional knowledge and skills to the students even though it's not covered in the manual but are expected from the students by the industries.
8. Teacher may suggest the students to refer additional related literature of the Technical papers/ Reference books/ Seminar proceedings, etc.
9. Teacher shall assess the performance of students continuously as per norms prescribed by MSBTE.
10. During assessment teacher is expected to ask questions to the students to tap their Achievements grading related knowledge and skills. So that, student can prepare while submitting record of the practical focus should be given on development of enlisted skills rather than theoretical knowledge.

Instructions for Students

1. Understand the purpose of practical and its implementation.
2. Student shall develop practical skills as expected by the Industries.
3. Listen carefully to the instructions given by the teacher about importance of relevant program Outcomes, relevant course outcomes, practical significance, competency and practical skills, practical outcome and the theoretical background during the practical session.
4. Write the answers of the questions allotted by the teacher during practical session.
5. Student should develop the habit of group discussion related to the practical, so that exchange of knowledge/skills could take place.
6. Student shall attempt to develop related hands-on-skills to gain confidence.
7. Student shall refer technical magazines, websites related to the scope of the course.
8. Student should develop habit to submit the practical, exercise continuously and progressively on the scheduled dates and should get the assessment done.
9. Student should be well prepared while submitting the write up of the exercise.
10. Student should not hesitate to ask any difficulty faced during conduct of practical.

Contents Page

List of Practical's and Progressive Assessment Sheet

	(Project Evaluation and Review Technique)						
16	Track progress of the project using Timeline Charts/ Gantt charts.	66					
17	Prepare SQA plan that facilitates various attributes of quality of process.	70					
18	*Prepare SQA plan that facilitates various attributes of quality of product.	74					
				Total Marks			

(**Note : Out of above suggestive LLOs -**
- **'*'** Marked Practical (LLOs) Are mandatory.
- Minimum 80% of above list of lab experiment are to be performed.
- Judicial mix of LLOs are to be performed to achieve desired outcomes.

Note: *Marks to be transferred to Performa of (CIAAN-K format).*

Practical No.1: Write problem statement to define the project title with bounded scope of the project.

I Practical Significance

Defining a problem statement is a foundational step in any software development project. It helps students to articulate the exact issue they are solving, set clear goals, and establish boundaries for the system. This practice is essential in software engineering to ensure that the development process is focused, efficient, and meets user expectations. By writing a well-structured problem statement, students gain practical skills in requirement gathering, problem analysis, and scope definition—skills that are highly valued in the industry.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Analyze and document software requirements effectively.

III Course Level Learning Outcome

CO1- Select suitable software development process model.

IV Laboratory Learning Outcome

LLO 1.1-Use software tool to Write problem statement and identify scope of the project.

V Relevant Affective domain related Outcome(s)

1. Develops a positive attitude toward identifying and solving real-world problems.
2. Demonstrates responsibility in clearly defining project goals and limitations.
3. Values the importance of clear documentation and effective communication in project planning.

VI Relevant Theoretical Background

Students should have a basic understanding of the Software Development Life Cycle (SDLC) and the principles of requirement engineering, including various types of requirements. A **problem statement** is a clear, concise description of the issue that a project aims to solve. It defines the context, identifies the gap or challenge, and outlines the intended outcome. Similarly, the **scope of a project** refers to the boundaries within which the project will be executed—it specifies what is included and excluded in terms of features, functions, and deliverables. Understanding these elements is essential for drafting a focused and effective problem statement that guides the development process.

VII Exercises

Follow the procedure given below:

1. Select a meaningful and relevant **project title**.
2. Study the **background** of the problem using online resources or user feedback.
3. Identify the **issue** and explain why it needs a solution.
4. Propose a **software solution** to address the issue.
5. Clearly define the **scope** – what the project will include or exclude.
6. Use a **software tool** (e.g., MS Word, Google Docs, Notion) to create a formatted document.
7. Review and finalize the problem statement.

VIII Procedure with Example:

Project Title: *Online Book Donation and Request Portal*

Problem Statement:

Many students and individuals find it difficult to access affordable educational resources, especially in underprivileged areas. At the same time, numerous people have unused books that could be reused. However, there is no common platform to connect book donors and seekers. The *Online Book Donation and Request Portal* aims to provide a digital solution to bridge this gap by allowing users to donate or request books through an easy-to-use web portal.

Scope of the Project:

- **In scope**: User registration, book listing, search and filter functionality, request and donation tracking.
- **Out of scope**: Payment integration, courier or delivery management, and mobile app development.

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Project Management Tools:	Open source Software such as Jira.		

X Precautions to be followed

1. Select a project title that is realistic and achievable within the given scope.
2. Clearly define the problem and scope to avoid ambiguity.
3. Use proper formatting and save your work regularly in the chosen software tool.

XI Conclusion

…………………………………………………………………………………………...

…………………………………………………………………………………………...

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write problem statement to define online shopping with bounded scope of the project.
2. Write problem statement to define Online Library Management System with bounded scope of the project.
3. What Problem Statement and Scope of Project?

[Space for Answer]

……………………………………………..……………………………………………

…………………………………………………………………………………………...

…………………………………..………………………………………………………

……………………………………………..……………………………………………

…………………………………………………………………………………………...

……………………………………………………………………………………………

……………………………………………..……………………………………………

…………………………………………………………………………………………...

…………………………………..………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.youtube.com/watch?v=gFLictX8WPw&ab_channel=Tryve
2. https://business.adobe.com/blog/basics/project-scope-definition-best-practices-examples-and-more
3. https://www.pmi.org/blog/how-to-write-a-problem-statement
4. https://www.mural.co/blog/problem-statements

XIV Assessment Scheme

Performance indicators		**Weightage**
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.2: Select relevant process model to define activities and related tasks set.

I Practical Significance

Selecting a relevant process model to define activities and related task sets is a key aspect of software engineering practice. It enables students to understand how different models influence the planning, execution, and delivery of software projects. This approach helps in aligning the development process with project goals, user needs, and technical constraints. By applying the appropriate model, students learn how to manage workflows, allocate resources effectively, and adapt to changes during development. This experience builds essential competencies in project structuring, time management, and quality assurance—skills that are directly applicable and highly valued in the software industry.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Analyze and document software requirements effectively.
3. Use version control tools for collaborative development.

III Course Level Learning Outcome

CO1- Select suitable software development process model.

IV Laboratory Learning Outcome

LLO 2.1- Use appropriate process model and activities related to project.

V Relevant Affective domain related Outcome(s)

1. Select the most suitable process model for a given project.
2. Organize and plans project tasks and activities responsibly.
3. Demonstrate clear communication and proper documentation during project execution.

VI Relevant Theoretical Background

A software process model shows the steps and order of activities needed to develop software. Common models like Waterfall, Agile, and Six Sigma give different ways to organize and manage these tasks depending on the project's needs. Understanding these models helps students learn how to pick the best method for their project or industry.

The process model chosen affects how testing is done, including what tests are needed and when to perform them. Learning about these models also introduces students to ideas of continuous improvement and quality control, which are important for making sure software works well. Having a good understanding of these concepts helps students make better decisions during software development and prepares them for real-world work.

Waterfall Model:

A step-by-step process where each phase (requirement, design, coding, testing) is done one after another. It works well when the project requirements are clear from the start.

Incremental Process Model (RAD):

The project is divided into small parts called increments. Each part is developed quickly and added to the system. This helps deliver working software faster.

Prototyping Model:

A basic version of the software is built first to show users. Their feedback helps improve and finalize the system. This model is good when the exact requirements are unclear.

Spiral Model:

This model repeats phases like planning, risk analysis, development, and evaluation in cycles. It is useful for large or risky projects because risks are managed early.

Agile Process Model:

Development happens in short cycles called sprints, with continuous feedback and improvements. It allows changes at any time and involves close teamwork with users.

VII Exercises

Follow the procedure given below:

1. **Learn about Software Process Models**
   Read and take notes on models like Waterfall, Agile, Spiral, etc.
2. **Assess the Needs of Stakeholders**
   Gather expectations from users or clients. (Google Forms, MS Forms, Word, Excel)
3. **Define Activities and Related Tasks**
   List project phases and break them into smaller tasks. (Excel, Google Sheets, Notion)
4. **Set Selection Criteria**
   Write down important factors like cost, time, and team size. (Word, Google Docs, Excel Sheets)
5. **Choose the Suitable Model**
   Compare process models and select the one that matches the project needs. (Excel, Google Sheets, Word, and Google Docs).

VIII Procedure with Example:

Project Title: *Online Book Donation and Request Portal*

Step 1: Use different models to problem statement

Model	**Description**	**Suitable For**	**Application to This Project**
Waterfall	Step-by-step, complete one phase before next	When requirements are fixed	Use if all portal features are clearly known
RAD (Incremental)	Build small parts fast, get feedback, add more	Fast delivery needed	Quickly develop login, donation, request parts
Prototyping	Build basic version, get user feedback, improve	Requirements unclear	Prototype donation/ request forms to understand needs
Spiral	Repeat cycles of planning and risk management	Complex or risky projects	For secure donation tracking and admin controls
Agile	Work in short cycles, get feedback, improve continuously	Flexible and changing requirements	Deliver portal features bit by bit, adapt as needed

Step 2: Assess the Needs of Stakeholders:

Stakeholder	Role	Expectations
Donors	People donating books	Easy donation process, condition details
Requesters	People requesting books	Quick search, request tracking
Admin	Portal manager	Manage donations and requests easily
Volunteers	Help with logistics	Donation schedules, communication
Community Partners	Libraries/NGOs	Data sharing and reports
IT Support Team	Maintenance and support	Easy system maintenance

Step 3: Define Activities and Related Tasks

Project Phase	Activity	Important Tasks
Requirement Analysis	Understand what is needed	Identify user needs, list features, gather feedback
System Design	Plan system structure	Create UI layouts, define database structure
Development	Build functional parts	Code login, donation form, request system, admin panel
Testing	Ensure everything works	Test all functions, fix bugs, verify user experience
Deployment	Make it live for users	Host the system, connect the database, open to users
Maintenance	Keep it running smoothly	Monitor usage, fix issues, add small updates if needed

Step 4. Set Selection Criteria

Criteria	Description	Importance	Reason
Cost	Total budget for development	High	Choose a model that fits budget
Time	Deadline to complete project	High	Must finish within 6 months
Flexibility	Ease of handling changes	High	Requirements may evolve
Scalability	Can support many users	High	Needs to grow with demand
Quality	Reliability and performance	High	Must work fast and without bugs

Step 5: Choose the Suitable Model

Model	Best Used When	Reason for Suitability
Waterfall	Requirements are fixed and well understood	Less suitable — project needs ongoing updates and user feedback
RAD	Project needs quick delivery in parts	Good for fast development of modules like donate, login, request
Prototyping	Requirements are unclear at the start	Helps gather user input and refine main features early

Spiral	Project involves risks or sensitive data	Useful for secure features like admin access and donation tracking
Agile	Needs regular updates and user input	Best choice — supports feedback, flexible improvements, fast delivery

For the Online Book Donation and Request Portal, **Agile** is the most suitable model due to its flexibility and user-focused development.

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Project Management Tools:	Open source Software such as Jira.		

X Precautions to be followed

1. Select a project title that is realistic and achievable within the given scope.
2. Clearly define the problem and scope to avoid ambiguity.
3. Use proper formatting and save your work regularly in the chosen software tool.

XI Conclusion

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Which process model recommend for the Online Library Management System.
2. Write down the waterfall model activities and related tasks set for Hospital Patient Record Management System.

[Space for Answer]

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

………………………………………………………………………………………..

……………………………………..……………………………………………………

…………………………………………..…………………………………………….

………………………………………………………………………………………..

……………………………………..……………………………………………………

…………………………………………..…………………………………………….

…………………………………………..…………………………………………….

………………………………………………………………………………………..

……………………………………..……………………………………………………

…………………………………………..…………………………………………….

………………………………………………………………………………………..

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.youtube.com/watch?v=gFLictX8WPw&ab_channel=Tryve
2. https://business.adobe.com/blog/basics/project-scope-definition-best-practices-examples-and-more
3. https://www.pmi.org/blog/how-to-write-a-problem-statement
4. https://www.mural.co/blog/problem-statements

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.3: Gather application specific requirements for assimilate into RE (Requirement's engineering) Model.

I Practical Significance

Gathering application-specific requirements and assimilating them into a Requirements Engineering (RE) model is a crucial phase in the software development lifecycle. It helps students to interact with stakeholders, extract relevant information, and interpret business needs into clear, actionable software requirements. This exercise develops critical thinking, communication, and analytical skills necessary for defining functional and non-functional requirements. Integrating these requirements into a structured RE model—using techniques such as use case diagrams, data flow diagrams, or requirement traceability matrices—ensures systematic documentation and traceability throughout the project. Practicing this process allows students to apply industry-standard methods used in real software projects, thereby enhancing their readiness for professional roles in software engineering and systems analysis.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Analyze and document software requirements effectively.
3. Use version control tools for collaborative development.

III Course Level Learning Outcome

CO 2- Prepare software requirement specification.

IV Laboratory Learning Outcome

LLO 3.1 Apply the principles of requirement engineering.

V Relevant Affective domain related Outcome(s)

1. Select the most suitable process model for a given project.
2. Organize and plans project tasks and activities responsibly.
3. Demonstrate clear communication and proper documentation during project execution.

VI Relevant Theoretical Background

Students should have a basic understanding of how to gather application-specific requirements and how to incorporate them into a Requirements Engineering (RE) model. Gathering requirements involves identifying the specific needs, constraints, and expectations of users and stakeholders for the software application. This process helps to clearly define what the system must do and under what conditions it should operate. Integrating these requirements into an RE model means organizing and documenting them systematically using tools like use case diagrams, data flow diagrams, or requirement traceability matrices. Understanding this process is essential for creating a structured, clear, and complete representation of the requirements, which guides the entire software development lifecycle.

VII Exercises

Follow the procedure given below:

1. Select a meaningful and relevant software project.

2. Research the project background by consulting users, stakeholders, or online resources.
3. Identify and list the **specific requirements** related to the application.
4. **Classify the requirements** into functional and non-functional categories.
5. Organize and document these requirements using a Requirements Engineering (RE) model (e.g., use case diagrams, data flow diagrams).
6. Use a software tool (e.g., MS Word, Google Docs, Draw.io) to prepare a clear and formatted RE document.
7. Review the requirements and RE model with peers or mentors, and finalize the documentation.

VIII Procedure with Example:

Project Title: *Online Book Donation and Request Portal*

Problem Statement:

Gathering application-specific requirements is essential to understand the detailed needs of users and stakeholders for the Online Book Donation and Request Portal. Without clear and well-organized requirements, the development process can become unfocused and may fail to meet user expectations. This project focuses on collecting, analyzing, and documenting the specific functional and non-functional requirements from potential users, donors, and administrators. These requirements will then be organized into a Requirements Engineering (RE) model to provide a structured and clear blueprint for the system design and development.

Scope of the Project:

In scope: Identifying user needs such as registration, book donation and request processes, search and filter options, and tracking features; documenting these requirements using RE tools like use case diagrams and data flow diagrams.

Out of scope: Technical implementation details, UI/UX design specifics, and deployment environment considerations.

IX Required Resources

Sr. No.	Name of Resource	Major Specification		Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards		
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Project Management Tools:	Open source Software such as Jira.		

X Precautions to be followed

1. Select a project title that is realistic and achievable within the given scope.
2. Clearly define the problem and scope to avoid ambiguity.
3. Use proper formatting and save your work regularly in the chosen software tool.

XI Conclusion

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write the application-specific requirements for an Online Shopping system and organize them into a Requirements Engineering (RE) model.
2. Identify and document the requirements for an Online Library Management System and represent them using RE techniques.
3. What is the importance of gathering application-specific requirements and how are they integrated into an RE model?

[Space for Answer]

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

……………………………………………………………………………………………..

………………………………………………………………………………………………..

……………………………………..……………………………………………………

………………………………………………………………………………………………..

……………………………………..……………………………………………………

………………………………..…………………………………………………………

…………………………………………………………………………………………………

……………………………………………………………………………………………..

………………………………..…………………………………………………………

…………………………………………………………………………………………………

……………………………………..……………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.tutorialspoint.com/software_engineering/software_requirements_specification.htm
2. https://www.geeksforgeeks.org/requirements-engineering/
3. https://www.atlassian.com/software/jira
4. https://www.lucidchart.com/pages/tutorial

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			Dated Sign of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.4: Prepare broad SRS (software requirement Specification) for the project.

I Practical Significance

Preparing a broad Software Requirements Specification (SRS) is a vital step in the software development lifecycle that consolidates all gathered requirements into a comprehensive and structured document. It enables students to articulate software functionality, performance criteria, design constraints, and system interfaces in a clear and formal manner. By organizing requirements according to IEEE standards or similar frameworks, students learn to communicate technical details effectively with developers, testers, and stakeholders. The SRS serves as a foundation for design, development, and validation processes, ensuring that all parties share a common understanding of project scope and objectives. This process helps students gain familiarity with documentation practices used in real-world software projects, improving their ability to plan, negotiate, and manage software systems in professional environments.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Analyze and document software requirements effectively.
3. Communicate technical concepts clearly and professionally.

III Course Level Learning Outcome

CO2- Prepare software requirement specification.

IV Laboratory Learning Outcome

LLO 4.1- Create SRS document for the project.

V Relevant Affective domain related Outcome(s)

1. Select the most suitable process model for a given project.
2. Organize and plans project tasks and activities responsibly.
3. Demonstrate clear communication and proper documentation during project execution.

VI Relevant Theoretical Background

Students should have a basic understanding of how to prepare a broad Software Requirements Specification (SRS) and its role in guiding software development. Preparing an SRS involves compiling all functional and non-functional requirements, system interfaces, constraints, and assumptions into a well-structured and formal document. This specification acts as a single source of truth for developers, testers, and stakeholders, ensuring that everyone has a shared understanding of what the system should accomplish. A broad SRS is typically organized using industry standards such as IEEE 830, enabling consistency and clarity throughout the project. Understanding how to create and use an SRS is essential for maintaining traceability, minimizing misunderstandings, and supporting successful project planning, design, and validation efforts.

VII Exercises

Follow the procedure given below:

1. Select a meaningful and relevant software project.

2. Gather requirements through user/stakeholder interaction or research.
3. Identify and classify functional and non-functional requirements.
4. Follow a standard structure (e.g., IEEE 830) for organizing the SRS.
5. Include system scope, interfaces, constraints, and performance needs.
6. Use appropriate tools to create a well-formatted document.
7. Review and refine the SRS based on feedback for accuracy and clarity.

VIII Procedure with Example:

Project Title: *Online Book Donation and Request Portal*

Problem Statement:

Preparing a broad Software Requirements Specification (SRS) is crucial to formally document and communicate the comprehensive needs of users, donors, and administrators involved in the Online Book Donation and Request Portal. Without a well-defined SRS, project development risks misinterpretation, scope creep, and inconsistency in implementation. This project focuses on compiling the gathered functional and non-functional requirements into a clear, standardized, and detailed SRS document. The SRS will act as a definitive guide for developers and stakeholders, ensuring that the system design and development align closely with the agreed-upon objectives and constraints.

Interview potential users like donors, recipients, and administrators to understand their needs and expectations. Research similar platforms to gather functional features. Identify and document the system's functional and non-functional requirements. Organize the SRS into sections like Introduction, Overall Description, Specific Requirements, System Features, External Interface Requirements, etc. Use a documentation tool to prepare a clearly formatted and professional-looking SRS document. Use MS Word or Google Docs to draft the SRS with headings, tables, and diagrams for clarity. Review the SRS with peers, mentors, or stakeholders and incorporate feedback to finalize the specification

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards		
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Project Management Tools	Open source Software such as Jira.		

X Precautions to be followed

1. Select a project title that is realistic and achievable within the given scope.
2. Clearly define the problem and scope to avoid ambiguity.
3. Use proper formatting and save your work regularly in the chosen software tool.

XI Conclusion

………………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Prepare a broad Software Requirements Specification (SRS) for an Online Shopping system covering functional and non-functional requirements.
2. Develop a structured SRS document for an Online Library Management System following IEEE 830 or similar standards.
3. Explain the significance of a Software Requirements Specification (SRS) and how it supports the software development lifecycle.

[Space for Answer]

…………………………………………………………………………………………...

…………………………………………………………………………………………….

…………………………………………………………………………………………….

…………………………………………………………………………………………...

…………………………………………………………………………………………….

…………………………………………………………………………………………….

…………………………………………………………………………………………...

…………………………………………………………………………………………….

…………………………………………………………………………………………….

…………………………………………………………………………………………...

…………………………………………………………………………………………….

…………………………………………………………………………………………….

…………………………………………………………………………………………….

…………………………………………………………………………………………….

…………………………………………………………………………………………...

…………………………………………………………………………………………….

………………………………………………………………………………………..

………………………………………..………………………………………………

………………………………….…………………………………………………….

…………………………………………..…………………………………………….

………………………………….…………………………………………………….

……………………………………………………………………………………….

………………………………………………………………………………………..

………………………………….…………………………………………………….

……………………………………………………………………………………….

…………………………………………..…………………………………………….

………………………………………………………………………………………..

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.tutorialspoint.com/software_engineering/software_requirements_specification.htm
2. https://www.geeksforgeeks.org/requirements-engineering/
3. https://www.atlassian.com/software/jira
4. https://www.lucidchart.com/pages/tutorial

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			Dated Sign of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.5: Write use-cases and draw use-case diagram.

I Practical Significance

Use-cases and use-case diagrams are key tools in the software development lifecycle for capturing system functionality from the perspective of those who interact with the system. This process helps students identify the different participants and the ways they engage with the system, outlining scenarios that describe how the system responds under various conditions. Developing detailed use-case descriptions along with use-case diagrams strengthens students' ability to analyze and model requirements clearly and effectively. This approach encourages critical thinking and improves communication with stakeholders by providing a visual, user-focused representation of system behavior. Mastering use-case modeling equips students with industry-relevant skills that support collaboration among analysts, designers, and developers throughout the project.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Design modular and maintainable software architectures.
3. Develop reliable and scalable software solutions.

III Course Level Learning Outcome

CO3 - Construct different Software design models.

IV Laboratory Learning Outcome

LLO 5.1- Construct use case diagram for software models.

V Relevant Affective domain related Outcome(s)

1. Select the most suitable process model for a given project.
2. Demonstrates attention to detail and clarity when developing use-cases and diagrams.
3. Demonstrate clear communication and proper documentation during project execution.

VI Relevant Theoretical Background

Students should have a basic understanding of how to develop use-cases and represent them using use-case diagrams, and how these tools contribute to effective system analysis and design. Creating use-cases involves identifying key interactions between participants and the system to describe how specific goals are achieved through system behavior. Use-case diagrams provide a visual representation of these interactions, helping to clarify functionality and system boundaries. These modeling techniques are part of standard industry practices, commonly applied using UML (Unified Modeling Language), and serve as important communication tools among stakeholders, analysts, and developers. Understanding how to construct and use use-cases and diagrams is essential for capturing requirements accurately, improving collaboration, and guiding system implementation.

VII Exercises

Follow the procedure given below:

1. Select a meaningful and relevant software project.
2. Identify system interactions and main functions.

3. Define key use-cases with clear steps.
4. Draw the use-case diagram using UML notation.
5. Use appropriate tools for documentation.
6. Review and refine for clarity and completeness.

VIII Procedure with Example:

Project Title: *Online Book Donation and Request Portal*

Problem Statement:

Defining and modeling system interactions through use-cases and use-case diagrams is essential for understanding the functional flow of the Online Book Donation and Request Portal. Without clear and well-structured use-cases, there may be confusion regarding user roles, system boundaries, and expected system behavior. This project aims to identify and document key system scenarios and represent them visually using use-case diagrams. These artifacts will help bridge communication between stakeholders and developers by clearly outlining how the system is expected to operate under different conditions, supporting better analysis, design, and implementation.

Interview different types of participants such as donors, recipients, and administrators to gather scenarios of how they interact with the system. Analyze their goals and define use-cases for major features like donating books, requesting books, managing listings, and approving requests. Write detailed use-case descriptions that include steps, conditions, and outcomes. Create a use-case diagram using UML notation to visually represent these interactions. Tools like *Draw.io* or *StarUML* can be used for diagram creation. Present the use-cases and diagrams to peers or mentors for feedback, and revise them for accuracy and completeness.

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Design Tools	Free Use Case Diagram Creator		

X Precautions to be followed

1. Follow standard UML conventions in diagrams.
2. Use proper tools to draft and save diagrams.
3. Review diagrams with peers or mentors for accuracy.

XI Conclusion

…………………………………………………………………………………………...

…………………………………………………………………………………………...

……………………………………………………………………………………………...

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Identify and document use-cases for an Online Shopping system, and create the corresponding use-case diagram.
2. Develop use-case descriptions and draw a use-case diagram for an Online Library Management System.
3. Explain the importance of use-cases and use-case diagrams in understanding system requirements and facilitating communication among stakeholders.

[Space for Answer]

……………………………………………………………………………………………...

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………...

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………...

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………...

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………...

……………………………………………………………………………………………

……………………………………………………………………………………………...

………………………………………..……………………………………………

……………………………………..………………………………………………

………………………………………..……………………………………………

……………………………………..………………………………………………

………………………………………..……………………………………………

……………………………………………………………………………………...

……………………………………..………………………………………………

………………………………………..……………………………………………

………………………………………..……………………………………………

……………………………………………………………………………………...

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm
2. https://www.geeksforgeeks.org/use-case-diagram-in-uml/
3. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/
4. https://www.lucidchart.com/pages/tutorial/uml-use-case-diagrams

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.6: Draw the activity diagram to represent flow from one activity to another for software development.

I Practical Significance

Activity diagrams are useful tools in software development for showing the flow of actions and decisions within a system. They help students visualize how processes move from one step to another, including choices and parallel paths. This improves understanding of system logic and workflow. Creating these diagrams develops analytical and planning skills. It also enhances communication with stakeholders by clearly presenting process flows.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Design modular and maintainable software architectures.
3. Develop reliable and scalable software solutions.

III Course Level Learning Outcome

CO3 - Construct different Software design models.

IV Laboratory Learning Outcome

LLO 6.1- Design activity diagram for the project.

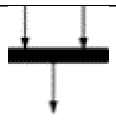
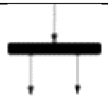
V Relevant Affective domain related Outcome(s)

1. Select the most suitable process model for a given project.
2. Demonstrates clarity and precision when creating activity diagrams.
3. Ensure workflow diagrams are accurate and follow standards.

VI Relevant Theoretical Background

Students should understand how to create activity diagrams to show the step-by-step flow of actions in a system. These diagrams include actions, decisions, and parallel processes to explain how the system works. Activity diagrams use UML (Unified Modeling Language) symbols and are common in real-world software design. They help in analyzing logic, improving process clarity, and planning development.

Notations used in activity diagram:

Shape	Description
●	**Start Symbol**: Represents the beginning of a process or workflow in an activity diagram.
▭	**Activity Symbol:** Indicates the activities that make up a modeled process.
→	**Connector Symbol (Arrow):** Shows the direction of flow in the activity.
	**Join Symbol:** A thick vertical or horizontal line that combines concurrent activities.
	**Fork Symbol:** Splits a single activity flow into two concurrent activities.

◇	**Decision Symbol:** Diamond shape representing branching or merging of flows.
🗋	**Note Symbol:** Adds messages or comments that don't fit within the diagram.
◉	**Final Node –** Marks the end of the workflow

VII Exercises

Draw an activity diagram for an Online Food Delivery System showing activities like browsing menus, placing orders, making payments, order preparation, and delivery. Include decision points for order confirmation and payment success.

VIII Procedure:

Steps to Draw an Activity Diagram for Software Development

1. **List Activities** – Identify key phases (e.g., Requirements, Design, Coding, Testing).
2. **Start Node** – Use a solid circle to show the starting point.
3. **Add Actions** – Draw each phase as a rounded rectangle.
4. **Connect with Arrows** – Use arrows to show the flow between actions.
5. **Use Decision Nodes** – Add diamonds for choices (e.g., test pass/fail).
6. **Fork/Join if needed** – Show parallel activities using bars.
7. **End Node** – Use a circle with a dot to mark the end.
8. **Label Clearly** – Name each activity and decision.
9. **Add Swim lanes** – Optional: divide by roles (e.g., Developer, Tester).
10. **Review** – Check for accuracy and logical flow.

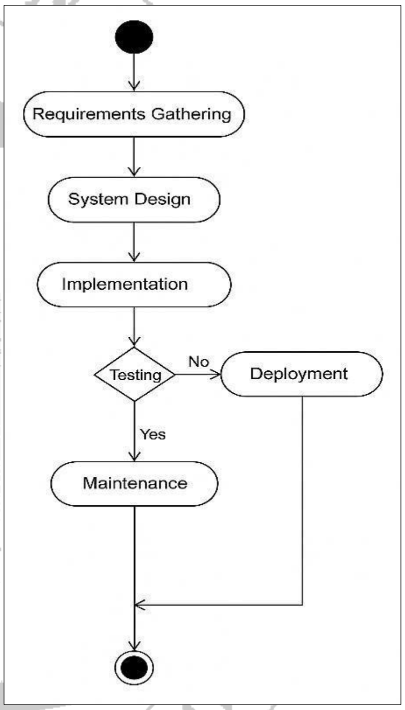


Fig 7.1 Sample Activity diagram

IX Required Resources

Sr. No.	Name of Resource	Major Specification		Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards		
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Design Tools	Draw.io, Decision Table Maker, Tiny tools		

X Precautions to be followed

1. Follow standard UML conventions in diagrams.
2. Use proper tools to draft and save diagrams.
3. Review diagrams with peers or mentors for accuracy.

XI Conclusion

……………………………………………………………………………………………...

……………………………………………………………………………………………...

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Develop an activity diagram illustrating the steps involved in user registration and login for an online portal.
2. What is an activity diagram used for in software engineering?
3. Name the key components of an activity diagram.
4. How does an activity diagram differ from a flowchart?

[Space for Answer]

……………………………………………………………………………………………...

……………………………………………………………………………………………….

……………………………………………………………………………………………….

……………………………………………………………………………………………...

……………………………………………………………………………………………….

……………………………………………………………………………………………….

……………………………………………………………………………………………...

……………………………………………………………………………………………….

……………………………………………………………………………………………….

……………………………………………………………………………………………...

……………………………………………………………………………………………….

……………………………………………………………………………………………….

……………………………………………………………………………………………….

……………………………………..……………………………………………

………………………………………………………………………………..

……………………………………..……………………………………………

………………………………………………………………………………..

……………………………………..……………………………………………

………………………………………………………………………………………

……………………………………..……………………………………………

…………………………………..…………………………………………………

……………………………………..……………………………………………

……………………………………..……………………………………………

………………………………………………………………………………..

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/
2. https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
3. https://www.lucidchart.com/pages/uml-activity-diagram
4. https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.7: Create DFDs (data flow diagram), Decision tables and E-R (entity-relationship) diagram.

I Practical Significance

DFDs, Decision Tables, and E-R Diagrams help visualize data flow, decision logic, and relationships in a system. These tools improve students' ability to analyze and design software clearly. Using them supports better planning and problem-solving skills. Clear visual representations simplify communication of complex system details to stakeholders.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Design modular and maintainable software architectures.
3. Develop reliable and scalable software solutions.

III Course Level Learning Outcome

CO3 - Construct different Software design models.

IV Laboratory Learning Outcome

LLO 7.1- Draw data flow diagram for the project.

LLO 7.2- Create Decision tables and E-R diagram.

V Relevant Affective domain related Outcome(s)

1. Select the most suitable process model for a given project.
2. Demonstrates clarity and precision when creating DFD, E-R diagrams.
3. Ensure workflow diagrams are accurate and follow standards.

VI Relevant Theoretical Background

Students should understand how to create DFDs, Decision Tables, and E-R diagrams to represent data flow, decision rules, and entity relationships in a system. These tools use standard symbols and notations and are widely used in software analysis and design. A Data Flow Diagram (DFD) is a graphical representation that depicts the flow of data within a system. It illustrates how data moves from external entities into the system, how it is processed, stored, and outputted. DFDs are a fundamental tool used in systems analysis and design to model the system's functional processes and data movement clearly and systematically.

- ***Processes:*** Represent the transformation or manipulation of data. Processes take inputs, process them, and produce outputs. In DFDs, processes are usually represented by circles or rounded rectangles.
- ***Data Flows***: Indicate the movement of data between entities, processes, and data stores. Data flows are shown by arrows pointing in the direction of data movement.
- ***Data Stores:*** Represent places where data is stored within the system, such as databases or files. They hold data for later retrieval or processing. Data stores are depicted as open-ended rectangles or parallel lines.
- ***External Entities:*** Are outside the system but interact with it by providing inputs or receiving outputs. These can be users, organizations, or other systems. They are shown as squares or rectangles.

VII Exercises

Design an Online Book Donation System where users can search for books available for donation, donate books by providing details, request donated books, and receive confirmation of donation or request. The system manages donor and recipient information, checks book availability, and confirms transactions. Create Data Flow Diagrams (DFDs) to represent the flow of data in the system.

VIII Procedure:

1. Choose a project like an Online Book Donation System.
2. Identify main data flows such as book donation, request, and approval.
3. List decision points, for example, eligibility to donate or approve requests.
4. Determine key entities like donors, books, and administrators.
5. Create a context-level DFD showing overall data movement.
6. Develop detailed DFDs for donation and request processes.
7. Construct decision tables for approval criteria.
8. Draw an E-R diagram illustrating relationships between donors, books, and requests.
9. Use tools like Draw.io and review diagrams for clarity and correctness.

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Design Tools	Draw.io, Decision Table Maker, Tiny tools		

X Precautions to be followed

1. Follow standard UML conventions in diagrams.
2. Use proper tools to draft and save diagrams.
3. Review diagrams with peers or mentors for accuracy.

XI Conclusion

………………………………………………………………………………………...

………………………………………………………………………………………...

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is a Data Flow Diagram (DFD)?
2. Describe the main components of a Data Flow Diagram with examples.

3. Create detailed DFDs, decision tables, and E-R diagrams for an Online Shopping System to show data flow, decision points, and entity relationships.
4. Develop DFDs and decision tables illustrating the order processing and payment approval steps, and draw an E-R diagram for customers, products, and orders.

[Space for Answer]

……………………………………..…………………………………………………

……………………………………………..…………………………………………

………………………………………………………………………………………...

……………………………………..…………………………………………………

……………………………………..…………………………………………………

……………………………………………..…………………………………………

………………………………………………………………………………………...

……………………………………..…………………………………………………

……………………………………………..…………………………………………

………………………………………………………………………………………...

……………………………………………..…………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.geeksforgeeks.org/difference-between-dfd-and-erd/
2. https://www.youtube.com/watch?v=KN-inGJG540
3. https://www.youtube.com/watch?v=qtTWBs49BCA
4. https://www.lucidchart.com/blog/data-flow-diagram-tutorial
5. https://archive.nptel.ac.in/content/storage2/courses/106108103/pdf/PPTs/mod5.pdf

XIV Assessment Scheme

Performance indicators		**Weightage**
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.8: Draw class diagram and Sequence diagram, State Transition Diagram.

I Practical Significance

Class, sequence, and state transition diagrams visualize system structure and behavior, improving design clarity and planning. These diagrams enhance communication with stakeholders.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Design modular and maintainable software architectures.
3. Develop reliable and scalable software solutions.

III Course Level Learning Outcome

CO3 - Construct different Software design models.

IV Laboratory Learning Outcome

LLO 8.1- Represent software project by class diagrams.

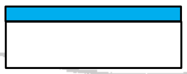
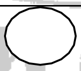
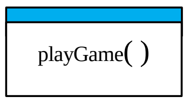
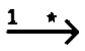
V Relevant Affective domain related Outcome(s)

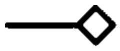
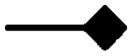
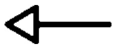
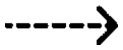
1. Select the most suitable process model for a given project.
2. Apply clarity and accuracy when creating class, sequence, and state diagrams.
3. Ensure workflow diagrams are accurate and follow standards.

VI Relevant Theoretical Background

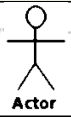
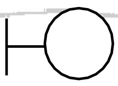
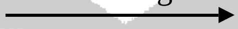
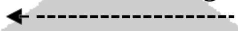
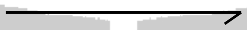
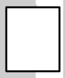
Students should understand how to create class diagrams, sequence diagrams, and state transition diagrams to represent system structure, interactions, and state changes. These diagrams use standard UML notations and are essential in object-oriented analysis and design.

UML Class Diagram Notations

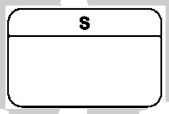
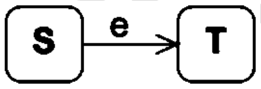
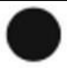
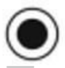
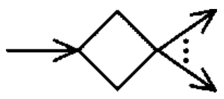
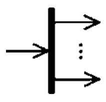
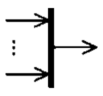
Element	Symbol /Notation	Description
Class		A blueprint with name, attributes, and operations.
Interface		Declares method signatures without implementation.
Attribute	+ for public - for private # for protected / for derived ~ for package	Variable/property in a class.
Method/Operation		Function or behavior of a class.
Association		Regular relationship between two classes.
Multiplicity		Indicates how many instances are involved.

Aggregation		Whole-part relationship; part can exist independently.
Composition		Strong whole-part; part can't exist without whole.
Generalization		Inheritance (is-a relationship).
Dependency		One class uses another temporarily.

UML Sequence Diagram Notations

Element	Symbol /Notation	Description
Actor/Object		Participant (user, system, or object) in interaction.
Lifeline	 Boundary	Shows object's presence during the interaction.
Message (Call)	Message 	A call from one object to another.
Return Message	Return message 	A return value or response from a method.
Asynchronous Message	Asynchronous Message 	Message sent without waiting for response.
Execution Occurrence		A thin rectangle showing when an object performs an action.
Destroy Message		Indicates termination of an object.

UML State Diagram Notations

Element	Symbol /Notation	Description
State		Description of a specific time span in an object's lifecycle.
Transition		State transition from a source to a target state.
Initial state		Start of a state machine diagram.
Final state		End of a state machine diagram.
Terminate node		Termination of an object's state machine diagram.
Decision node		Node from which multiple transitions can proceed.
Parallelization node		Splitting of a transition into multiple parallel transitions.
Synchronization node		Merging of multiple parallel transitions into one.

Shallow and deep history state	Ⓗ / Ⓗ*	Return address to a sub state or nested sub state.

VII Exercises

Design a Railway Reservation System where users can search for trains, book tickets, and receive booking confirmation. The system checks seat availability, processes user information, and confirms bookings after successful payment.

VIII Procedure:

1. Identify the main entities/objects involved (e.g., Passenger, Ticket, Train).
2. Determine the attributes, states, or interactions relevant to each entity.
3. Define relationships, messages, or state transitions based on system behavior.
4. Choose the diagram type and represent the elements accordingly (classes, lifelines, states).
5. Add details like methods, messages, or events to clarify functionality.
6. Review the diagram to ensure it accurately models the Railway Reservation System.

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Design Tools	Draw.io, Decision Table Maker, Tiny tools		

X Precautions to be followed

1. Follow standard UML conventions in diagrams.
2. Use proper tools to draft and save diagrams.
3. Review diagrams with peers or mentors for accuracy.

XI Conclusion

……………………………………………………………………………………………...

……………………………………………………………………………………………...

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Create detailed DFDs, decision tables, and E-R diagrams for an Online Shopping System to show data flow, decision points, and entity relationships.
2. Develop DFDs and decision tables illustrating the order processing and payment approval steps, and draw an E-R diagram for customers, products, and orders.
3. Differentiate between attributes and operations in a class diagram.
4. What is the purpose of visibility symbols (+, −, #) in a class diagram?

[Space for Answer]

……………………………………..……….…………………………………………

……………………………………..…………………………………………………

………………………………………………..……………………………………

…………………………………………………………………………………...

……………………………………..…………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………...

……………………………………………………..………………………………

……………………………………………………………………………………...

……………………………………..…………………………………………………

……………………………………………..……………………………………

……………………………………………………………………………………...

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.geeksforgeeks.org/difference-between-dfd-and-erd/
2. https://www.lucidchart.com/blog/data-flow-diagram-tutorial
3. https://www.youtube.com/watch?v=HuL9EMx8NQo
4. https://www.youtube.com/watch?v=VnRQ5CNC4Fs

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			Dated Sign of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 9: Create decision table for a project.

I Practical Significance

Decision tables are powerful tools used to represent complex business logic or conditions in a structured and visual format. They help in analyzing various combinations of inputs (conditions) and their corresponding system actions (outputs). This practical introduces students to modeling decision rules logically and systematically, improving decision-making, system design, and validation. By developing a decision table for their selected project, students gain insight into how software handles multiple conditions and are better equipped to analyze and implement conditional logic accurately.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Design modular and maintainable software architectures.
3. Develop reliable and scalable software solutions.

III Course Level Learning Outcome

CO3 - Construct different Software design models.

IV Laboratory Learning Outcome

LLO 9.1- Prepare decision table for the project.

V Relevant Affective domain related Outcome(s)

1. Select the most suitable process model for a given project.
2. Demonstrate clarity and precision when creating decision table for a project.
3. Ensure workflow diagrams are accurate and follow standards.

VI Relevant Theoretical Background

Students Students should understand the basics of decision tables—how they are structured and why they are used. A decision table consists of: ***Conditions:*** Logical statements or inputs. ***Actions:*** Operations to be executed based on the conditions.***Rules:*** Columns representing combinations of condition outcomes (Yes/No or True/False) mapped to specific actions.
Decision tables are widely used in requirement analysis, business rule validation, and test case generation. They ensure that all possible scenarios are considered, reduce ambiguity, and help design robust systems.

VII Exercises

Follow the procedure given below:

1. Select a meaningful and relevant software project.
2. Identify the key decision points or conditional logic within the system.
3. List all possible conditions and actions.
4. Create a decision table to represent condition combinations and their corresponding actions.
5. Use a documentation or table tool to create a well-formatted decision table.
6. Review and refine the table to ensure completeness and correctness

VIII Procedure with Example:

Project Title: Railway Reservation System.

Problem Statement:

In the Railway Reservation System, several conditions must be validated before confirming a ticket reservation—such as seat availability, passenger details accuracy, and payment status. Without a systematic approach, these checks may be applied inconsistently, potentially resulting in booking errors, double reservations, or user dissatisfaction. A decision table helps standardize this logic by clearly mapping each combination of conditions to a defined system action. This structured methodology improves clarity, ensures reliable and fair processing, and supports consistent development, testing, and maintenance.

Conditions	**Rule 1**	**Rule 2**	**Rule 3**	**Rule 4**
Seat Available?	Yes	Yes	No	Yes
Passenger Details Valid?	Yes	No	Yes	Yes
Payment Successful?	Yes	Yes	Yes	No
Actions				
Confirm Reservation	✓	✗	✗	✗
Show Error (Invalid Details)	✗	✓	✗	✗
Show Error (No Seats Available)	✗	✗	✓	✗
Show Error (Payment Failed)	✗	✗	✗	✓

IX Required Resources

Sr. No.	**Name of Resource**	**Major Specification**	**Qty.**	**Remarks**
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Design Tools	Draw.io, Decision Table Maker, Tiny tools		

X Precautions to be followed

1. Follow standard UML conventions in diagrams.
2. Use proper tools to draft and save diagrams.
3. Review diagrams with peers or mentors for accuracy.

XI Conclusion

……………………………………………………………………………………………...

……………………………………………………………………………………………...

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What are the main components of a decision table?
2. What are the advantages of decision table?
3. Create a decision table for an Online Shopping System.

[Space for Answer]

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.geeksforgeeks.org/software-engineering-decision-table
2. https://docs.oracle.com/en/cloud/paas/integration-cloud/user-processes/create-decision-tables.html
3. https://youtu.be/Znf96SUuP4o

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.10: Write test cases to validate requirements from SRS document.

I Practical Significance

Writing test cases based on the Software Requirements Specification (SRS) is a critical activity in software development that ensures the system meets user expectations. This practice helps students understand how to interpret and validate requirements through concrete test scenarios. It improves attention to detail, logical thinking, and a structured approach to identifying input conditions, expected outputs, and edge cases. Developing test cases also strengthens the connection between documentation and implementation, enhancing skills in quality assurance and ensuring reliable software performance.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Create and execute test cases to ensure software quality
3. Understand ethical and legal aspects of software engineering..

III Course Level Learning Outcome

CO3 - Construct different Software design models.

IV Laboratory Learning Outcome

LLO 10.1 Design test cases by referring SRS document.

V Relevant Affective domain related Outcome(s)

1. Select the most suitable process model for a given project.
2. Values the importance of requirement validation in delivering quality software.
3. Demonstrates responsibility and precision in designing test cases that reflect real-world expectations.

VI Relevant Theoretical Background

Students should have a foundational understanding of Software Requirements Specification (SRS) documents and their role in defining system behavior, constraints, and functionalities. Test case development is a critical step in the software testing life cycle, aimed at ensuring the system meets the stated requirements. Writing test cases involves identifying input conditions, expected outputs, and execution steps based on the requirements in the SRS. It draws upon concepts such as black-box testing, requirement traceability, and test coverage. This process helps in detecting discrepancies early, reducing defects, and improving software quality. Understanding how to derive test cases from the SRS equips students with skills essential for quality assurance, verification & validation, and is a standard practice in industry-grade software development.

VII Exercises

Write test cases for key functionalities of an Online Library Management System, such as user login, book search, borrowing and returning books, and viewing borrowed history. Each test case should include the scenario, input, expected output, and the related requirement ID.

VIII Procedure:

1. Select a system like Online Library Management System.
2. Identify key features: login, search, borrow, return, view history.
3. Assign a unique requirement ID to each feature.
4. Write test cases for each feature.
5. Include scenario, input, expected output, and requirement ID.
6. Use a table to organize all test cases clearly.
7. Review and improve test cases for accuracy.

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Project Management Tools	Open source Software such as Jira.		

X Precautions to be followed

1. Define clear, complete, and testable requirements for each functionality.
2. Use a consistent test case format and validate with stakeholders.

XI Conclusion

……………………………………………………………………………………………..

……………………………………………………………………………………………..

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Identify functional and non-functional requirements from a given SRS of an Online Library Management System.
2. Write detailed test cases to validate each requirement from the SRS, including scenario, input, and expected output.
3. Explain the importance of validating SRS requirements through test cases in ensuring software quality.

[Space for Answer]

……………………………………………………………………………………………..

……………………………………………………………………………………………….

……………………………………………………………………………………………….

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.youtube.com/watch?v=83-S5Qu6VP8
2. https://www.tutorialspoint.com/software_engineering/software_requirements_specification.htm
3. https://www.geeksforgeeks.org/requirements-engineering/
4. https://www.atlassian.com/software/jira
5. https://www.lucidchart.com/pages/tutorial

XIV Assessment Scheme

	Performance indicators	**Weightage**
	Process related: 15 Marks	**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
	Product related: 10 Marks	**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.11: Prepare test cases for Black Box Testing

I Practical Significance

Black Box Testing is a critical software testing technique used to validate the functionality of an application without looking into its internal structures or workings. Preparing test cases for Black Box Testing helps identify errors in functionality, user interface issues, input validation problems, and incorrect or missing features. This practical enables learners to design effective test cases that simulate real-world user inputs and evaluate system behavior, ensuring software quality and robustness. It also helps in developing analytical skills needed to anticipate how end-users might interact with the system.

II Industry/ Employer Expected Outcome

1. Design modular and maintainable software architectures.
2. Develop reliable and scalable software solutions.

III Course Level Learning Outcome

CO 3- Construct different Software design models.

IV Laboratory Learning Outcome

LLO 11.1- Write test cases for Black box testing.

V Relevant Affective domain related Outcome(s)

1. Follow safetypractices.
2. Maintain tools and equipment.
3. Follow ethical practices.

VI Relevant Theoretical Background

To prepare test cases for Black Box Testing, you need to understand some basic concepts of software testing. In ***Black Box Testing***, we do not look at the code inside the software. Instead, we check whether the software works correctly by giving different inputs and checking the outputs. A ***test case*** is a set of steps we follow to see if the software behaves as expected. Two useful methods for creating test cases are Equivalence Partitioning and Boundary Value Analysis. These help we choose test inputs in a smart way to test more with fewer cases. Also, we must know the functional requirements of the software, because Black Box Testing is mainly used to check if the software does what it is supposed to do.

VII Exercises

Follow the procedure given below to write the test cases of login form of "*https://practicetestautomation.com/practice-test-login/":*

1. **Read Requirements** – Understand the system's functional specifications.
2. **Identify Inputs/Outputs** – List valid inputs, expected outputs, and error conditions.
3. **Select Technique** – Choose a Black Box Testing method (e.g., **Equivalence Partitioning**, **Boundary Value Analysis**).
4. **Write Test Cases –** Create test cases with input, expected output, and pass/fail status.

5. **Execute Test Cases –** Run the software using test case inputs.
6. **Record Results** – Compare actual vs. expected output and document the result.

VIII Example

Suppose you are testing a **login system** that accepts a *username* and *password*. The valid username is **"admin"** and the valid password is **"1234".**

Using *Equivalence Partitioning*:

Test Case ID	**Description**	**Input (Username, Password)**	**Expected Output**
TC01	Valid input	admin, 1234	Login Successful
TC02	Invalid username	user1, 1234	Invalid Username or Password
TC03	Invalid password	admin, 0000	Invalid Username or Password
TC04	Both username and password invalid	user1, pass	Invalid Username or Password

IX Required Resources

Sr. No.	**Name of Resource**	**Major Specification**	**Qty.**	**Remarks**
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Project Management Tools	Open source Software such as Jira.		

X Precautions to be followed

1. Understand functional requirements clearly – Always refer to the specification to avoid missing or incorrect test cases.
2. Avoid assumptions about internal logic – Base test cases only on inputs and expected outputs, not the program’s code.
3. Record expected and actual outputs accurately – Clearly document test results to ensure reliable pass/fail.

XI Conclusion

…………………………………………………………………………………………...

…………………………………………………………………………………………...

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is a Black box Testing?
2. What is Test Case?
3. What is Equivalence Partitioning? Give an example.
4. How would you test a login page using Black Box Testing?
5. What will you do if the actual result differs from the expected result?

[Space for Answer]

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.guru99.com/software-testing.html
2. https://youtu.be/89VOHd8F8Ao
3. https://youtu.be/JHj3Xx74Pfc

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			Dated Sign of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.12: Identify risks involved in the project and prepare RMMM (RMMM-Risk Management, Mitigation and Monitoring) plan.

I Practical Significance

Risk management is a vital aspect of successful project execution, especially in software and engineering domains. Projects often face uncertainties such as technical challenges, unrealistic timelines, changing requirements, or resource limitations. Identifying these risks in advance and preparing a comprehensive RMMM (Risk Management, Mitigation, and Monitoring) plan helps project teams minimize potential disruptions. This practical is significant as it trains students to think proactively, evaluate possible threats, and build strategies to tackle them efficiently. It enables learners to systematically list risks, assess their potential impact and probability, and outline steps to avoid or control them. By developing an RMMM plan, students gain real-world insight into effective project planning and execution, improving the likelihood of completing projects successfully—on time, within scope, and budget.

II Industry/ Employer Expected Outcome

1. To apply project management techniques in software projects.
2. Understand ethical and legal aspects of software engineering.
3. Communicate technical concepts clearly and professionally.

III Course Level Learning Outcome

CO 4- Apply different planning and cost estimation techniques for a software product.

IV Laboratory Learning Outcome

LLO 12.1- Identify risk involved in the project

LLO 12.2- Prepare RMMM Plan.

V Relevant Affective domain related Outcome(s)

1. Demonstrates a responsible attitude towards identifying and managing project risks.
2. Shows willingness to work collaboratively in analyzing and solving potential project issues.

VI Relevant Theoretical Background

Before preparing an RMMM plan, it is important to understand some basic terms. A **risk** is anything that might go wrong in a project and affect its success. There are different **types of risks**, such as **project risks** (like delays), **technical risks** (such as software or hardware issues), **operational risks** (like lack of skilled people), and **business risks** (such as market changes). To deal with risks, we need to do **risk assessment**, which means checking how likely the risk is to happen (**probability**) and how bad the result would be (**impact**). We can calculate the **risk exposure** by multiplying probability and impact. An **RMMM plan** includes four steps: **risk identification** (finding risks), **risk analysis** (understanding the risks), **risk mitigation** (reducing or avoiding risks), and **risk monitoring** (keeping track of risks during the project). Knowing these simple concepts helps in making a good plan to handle risks in any project.

VII Exercises

For a project titled "***Student Attendance Management System***", identify at least five potential risks and prepare a Risk Management, Mitigation, and Monitoring (RMMM) plan using a tabular format.

VIII Procedure with Example for scenario to Identified Risks and RMMM Plan for *Online Food Delivery Application Development*:

1. **Select a Project**: Choose a sample project (e.g., online food delivery app, library management system, or e-commerce website).
2. **Understand Project Scope**: Clearly define the objectives, timeline, and key components of the selected project.
3. **Identify Risks**: List all possible risks that could affect the project. These may include delays, technical failures, resource shortages, or changes in client requirements.
4. **Classify Risks**: Categorize each risk (e.g., technical, project, business, operational).
5. **Estimate Probability and Impact**: For each identified risk, estimate the **probability** (likelihood of occurring) and **impact** (severity of the effect on the project).
6. **Calculate Risk Exposure**: Multiply probability by impact to get the **risk exposure**. This helps prioritize risks.
7. **Plan Mitigation Strategies**: For each high or medium-exposure risk, plan how to reduce or avoid it.
8. **Prepare Monitoring Plan**: Define how and how often each risk will be tracked during the project (e.g., weekly meetings, status reports).
9. **Document the RMMM Plan**: Create a table or report listing all risks along with their mitigation and monitoring strategies.
10. **Review and Submit**: Review the plan with the instructor or team, then submit the completed exercise.

Risk ID	**Risk Description**	**Category**	**Probability (P)**	**Impact (I)**	**Risk Exposure (P × I)**	**Mitigation Strategy**	**Monitoring Plan**
R1	Delay in backend development	Project Risk	High	High	High	Assign additional developers; use agile sprints to monitor progress	Weekly progress reviews with the team
R2	API integration issues with payment gateway	Technical Risk	Medium	High	Medium-High	Test payment APIs in early stages; consult documentation thoroughly	Log errors; maintain test results and bug reports
R3	App crashes on certain Android versions	Technical Risk	Medium	Medium	Medium	Perform extensive testing on multiple devices and OS versions	Regular QA testing and crash monitoring using Firebase
R4	Key team member leaves the project	Operation-al Risk	Medium	High	Medium-High	Document all work; maintain backup resources	Track team status; regular communication with HR
R5	Changes in client requirements during development	Project/ Business Risk	High	Medium	Medium-High	Freeze requirements after approval; use change request forms	Conduct review meetings with client regularly

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Design Tools:	Project risk Manager		

X Precautions to be followed

1. Identify only realistic and relevant risks based on the selected project.
2. Use a consistent scale for evaluating risk probability and impact.
3. Prepare an original RMMM plan without copying from other sources.

XI Conclusion

……

……

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What does RMMM stand for and why is it important?
2. What is the difference between risk mitigation and risk monitoring?
3. What is the purpose of risk monitoring?
4. How can you reduce the impact of a risk?

[Space for Answer]

……

……

……

……

……

……

……

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.project-risk-manager.com/blog/
2. https://youtu.be/UIg4pwY5k6A
3. https://medium.com/@sivajikm/risk-management-in-software-engineering-a-practical-guide-2166994febcd
4. https://www.geeksforgeeks.org/risk-mitigation-monitoring-and-management-rmmm-plan/

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.13: Calculate size of the project using Function point metric.

I Practical Significance

This practical is important because it helps us estimate the size of a software project using a method called the Function Point (FP) metric. By knowing the size, we can better plan how much time, cost, and effort will be needed to complete the project. It also helps in deciding the number of people required and setting realistic deadlines. This technique is very useful for project managers to manage software development smoothly and avoid delays or budget issues. It gives a clear idea of what the system will do and how complex it is.

II Industry/ Employer Expected Outcome

1. Apply project management techniques in software projects.
2. Follow coding standards and documentation practices.
3. Understand ethical and legal aspects of software engineering.

III Course Level Learning Outcome

CO 4 - Apply different planning and cost estimation techniques for a software product.

IV Laboratory Learning Outcome

LLO 13.1 Estimate size of project using function point matrix

V Relevant Affective domain related Outcome(s)

1. Show commitment to completing software projects on time.
2. Accept feedback and improve project work.
3. Value accuracy in cost and size estimation.

VI Relevant Theoretical Background

To calculate the size of a software project using the **Function Point (FP) metric**, we need to understand a few basic concepts. A **function point** is a way to measure what the software does for the user. It counts the different types of features the system provides. These include **External Inputs (EI)** like data entry forms, **External Outputs (EO)** like reports, **External Inquiries (EQ)** like search options, **Internal Logical Files (ILF)** which are internal databases, and **External Interface Files (EIF)** which are files used from outside systems. Each of these has a **complexity level** (low, average, or high) and a **weight** or score. We also consider some general system features like security, performance, or ease of use, which give us a **Value Adjustment Factor (VAF)**. Using these values, we calculate the final **Function Point** value, which helps estimate project size, effort, and cost. Each function is rated as Low, Average, or High, with predefined weight values.
The total function points are calculated using:

$$FP = \text{Total Unadjusted Function Points (i.e. UFP)} \times (0.65 + 0.01 \times \sum Fi)$$

Where $\sum Fi$ is the sum of 14 general system characteristics (each rated from 0 to 5).

VII Exercises

A student record system has the following components – 5 EIs, 4 EOs, 3 EQs, 3 ILFs, and 2 EIFs. All are of average complexity. If the total value (i.e. ∑Fi) of general system characteristics is 28, calculate the final function point value. (**Note**: Multiply the number of each component by its standard weight as per complexity)

VIII Procedure with Example:

Step 1: Identify functional components

Suppose a *library management system* has:

- 4 External Inputs (EI) – e.g., Add/Update/Delete Book Records
- 3 External Outputs (EO) – e.g., Reports
- 2 External Inquiries (EQ) – e.g., Search Book
- 2 Internal Logical Files (ILF) – e.g., Books, Members
- 1 External Interface File (EIF) – e.g., External Database Interface

Step 2: Assign complexity and corresponding weights

(Assuming all components are of **Average** complexity)

Function Type	Count	Weight (Avg)	Total
EI	4	4	16
EO	3	5	15
EQ	2	4	8
ILF	2	10	20
EIF	1	7	7
Total UFP			**66**

Step 3: Calculate Value Adjustment Factor (VAF)

Assume ∑Fi = 30 (*sum of 14 general system characteristics*)

VAF = 0.65 + (0.01 × 30) = **0.95**

Step 4: Calculate Final Function Points

FP = 66 × 0.95 = **62.7 ≈ 63 FP**

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Design Tools:	Draw.io, Decision Table Maker, Tiny tools		

X Precautions to be Followed

1. Clearly identify all system components (EIs, EOs, EQs, ILFs, EIFs) to avoid incorrect function counting.
2. Assign the correct complexity level (Low, Average, High) based on actual system details, not assumptions.
3. Ensure accurate input for general system characteristics to calculate the correct Value Adjustment Factor (VAF).

XI Conclusion

…………………………………………………………………………………………

…………………………………………………………………………………………

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is the purpose of using Function Point Analysis in software project planning?
2. List and explain the five basic components used in calculating Unadjusted Function Points (UFP).
3. Differentiate between Internal Logical Files (ILF) and External Interface Files (EIF) with examples.
4. Define Value Adjustment Factor (VAF). How is it used in calculating the final Function Points?
5. Why is it important to estimate software size before starting a project?

[Space for Answer]

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.youtube.com/watch?v=7xBcVtjmGwM&ab_channel=GateSmashers
2. https://www.youtube.com/watch?v=bKwvzXQKBRo&ab_channel=GateSmashers
3. https://www.fingent.com/blog/function-point-analysis-introduction-and-fundamentals/

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			Dated Sign of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.14: Calculate cost of the project using COCOMO (Constructive Cost Model) / COCOMO II approach.

I Practical Significance

Estimating the cost, time, and effort needed for a software project is important for good planning and management. COCOMO and COCOMO II are popular models that help software engineers and project managers decide how many resources are needed and how to manage the budget. These models make it easier to check if a project is possible, set proper deadlines, and avoid spending too much. This practical helps students learn how to use these models in real situations and manage software projects more effectively.

II Industry/ Employer Expected Outcome

1. Apply project management techniques in software projects.
2. Follow coding standards and documentation practices.
3. Understand ethical and legal aspects of software engineering.

III Course Level Learning Outcome

CO 4- Apply different planning and cost estimation techniques for a software product

IV Laboratory Learning Outcome

LLO 14.1 Estimate size of project using COCOMO approach.

V Relevant Affective domain related Outcome(s)

1. Show commitment to completing software projects on time.
2. Accept feedback and improve project work.
3. Value accuracy in cost and size estimation.

VI Relevant Theoretical Background

The basic idea behind cost estimation in software development is to predict the amount of effort, time, and financial investment needed to complete a project. The original COCOMO model, developed by *Barry Boehm* in 1981, estimates the effort required in person-months based on the size of the software, which is typically measured in Kilo Lines of Code (KLOC). The basic formula is

$$Effort = a \times (KLOC)^b$$

Where **'a'** and **'b'** depend on the type of project. COCOMO II is an updated version that introduces additional factors such as reuse, platform complexity, and personnel capability, and allows for cost estimation during different phases of the development lifecycle. The general formula in COCOMO II is

$$Effort = A \times (Size)^E \times \prod_{i=1}^{n} EM_i$$

where **'A'** is a constant, **'Size'** refers to the software size, **'E'** is an exponent derived from scale factors, and ***EMᵢ*** are effort multipliers based on various cost drivers. Understanding these models is crucial for applying structured and data-driven approaches to software project estimation.

VII Exercises

A software project has an estimated size of 50 KLOC. The project uses the COCOMO II post-architecture model with the parameters: *A = 2.94*, Scale Factor *(E) = 1.05*, the product of all Effort Multipliers ($\prod \boldsymbol{EM_i}$)= 1.2
Calculate the total effort required in person-months.

VIII Procedure with Example: to calculate the estimated effort for a software project using the COCOMO II (Post-Architecture) model.

Given:
Size of the project = 50 KLOC, A (constant) = 2.94, E (exponent) = 1.05, ∏EMi (product of effort multipliers) = 1.2

Step-by-Step Procedure:

1. Write the COCOMO II Effort Formula:

$$Effort = A \times (Size)^{E} \times \prod_{i=1}^{n} EM_i$$

2. Substitute the given values into the formula:

$$Effort = 2.94 \times (50)^{1.05} \times 1.2$$

3. Calculate the power term:

$$(50)^{1.05} \approx \mathbf{58.436}$$

4. Multiply the values:

$$Effort = 2.94 \times 58.436 \times 1.2$$

Effort ≈ 2.94 × 70.123 Effort ≈ 206.16 person-months

Final Answer: Estimated Effort = **206.16 person-months**

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Design Tools	Draw.io, Decision Table Maker, Tiny tools		

X Precautions to be followed

1. Use accurate KLOC values for correct cost estimation.
2. Select the appropriate COCOMO/COCOMO II model and parameters.
3. Enter realistic values for effort multipliers and scale factors.

XI Conclusion

……………………………………………………………………………………………

……………………………………………………………………………………………

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is the purpose of using the COCOMO model in software project estimation?
2. Differentiate between Basic COCOMO and COCOMO II.
3. What are the key factors affecting cost estimation in COCOMO II?
4. Define the term KLOC and explain its importance in cost estimation.
5. Explain the significance of the Effort Multipliers in COCOMO II.

[Space for Answer]

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://youtu.be/lr0vA_p6T7Q?list=PLsVQtZefq4BrzgRhyOKweZiADInojNduz
2. https://youtu.be/ARSXNaZv_YY
3. https://www.geeksforgeeks.org/software-engineering-cocomo-model/
4. https://www.geeksforgeeks.org/software-engineering-cocomo-ii-model/

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.15: Create software project scheduling charts using CPM (Critical Path Method) / PERT (Project Evaluation and Review Technique).

I Practical Significance

Creating software project scheduling charts using CPM (Critical Path Method) and PERT (Project Evaluation and Review Technique) is very useful in managing software development projects. These methods help in planning tasks, showing task dependencies, and finding the critical path—which is the longest chain of important tasks that decides the total project time. With these charts, we can estimate the project duration, especially when some tasks have uncertain timings (using PERT). They also help in managing resources, avoiding delays, and tracking progress during the project. By using CPM/PERT charts, we can complete projects on time and work more efficiently.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Analyze and document software requirements effectively
3. Apply project management techniques in software projects.

III Course Level Learning Outcome

CO 5- Apply project management techniques in software development.

IV Laboratory Learning Outcome

LLO 15.1 Prepare project schedule using CPM/PERT technique

V Relevant Affective domain related Outcome(s)

1. Demonstrate responsibility and discipline in planning and scheduling project tasks.
2. Show commitment to teamwork while analyzing task dependencies and timelines.
3. Exhibit proactive attitude in identifying critical activities and potential delays.

VI Relevant Theoretical Background

To create **CPM** or **PERT** charts, we need to understand a few basic concepts. An **Activity** is a task that takes time and effort. An **Event (or Node)** is a point where an activity starts or ends. A **Network Diagram** is a drawing that shows the flow and order of activities. The **Critical Path** is the longest path through these activities and determines the **minimum time** required to complete the project. **Slack (or Float)** is the extra time that an activity can be delayed without affecting the project completion time. In **PERT**, we use a formula to estimate the time of each task:

$$TE= \frac{O+4M+P}{6}$$

Where **O** is the **Optimistic time**, **M** is the **most likely time**, and **P** is the **Pessimistic time**. These concepts are enough to start making scheduling charts using **CPM** and **PERT**.

VII Exercises

A software project has the following activities with their estimated durations (in Days) and dependencies:

Activity	Duration (days)	Depends On
A	4	-
B	3	A
C	5	A
D	6	B
E	2	B, C
F	4	D, E

1. Draw the **network diagram** for the project using CPM.
2. Determine the **critical path** and the **minimum project duration**.
3. Calculate the **slack** time for each activity.

VIII Procedure with Example:

Step 1: List Activities and Their Dependencies

Write down all the project activities, their predecessors (which activities come before them), and the estimated time for each activity. For **PERT**, time estimates include Optimistic (O), Most Likely (M), and Pessimistic (P) durations.

Step 2: Calculate Expected Time (TE) for Each Activity

Use the **PERT formula** to calculate the expected time:

$$TE= \frac{O+4M+P}{6}$$

Step 3: Draw the Network Diagram

Create a flowchart showing all activities as nodes or arrows connected based on their dependencies.

Step 4: Determine Earliest Start (ES) and Finish (EF) Times

Calculate when each activity can start and finish at the earliest, moving from the start to the end of the project.

Step 5: Determine Latest Start (LS) and Finish (LF) Times

Calculate the latest times each activity can start and finish without delaying the project, moving backward from the end.

Step 6: Calculate Slack (Float) for Each Activity

Slack = LS – ES (or LF – EF). Activities with zero slack lie on the **critical path**.

Step 7: Identify the Critical Path

The critical path is the longest path through the network with zero slack activities. This path determines the minimum project duration.

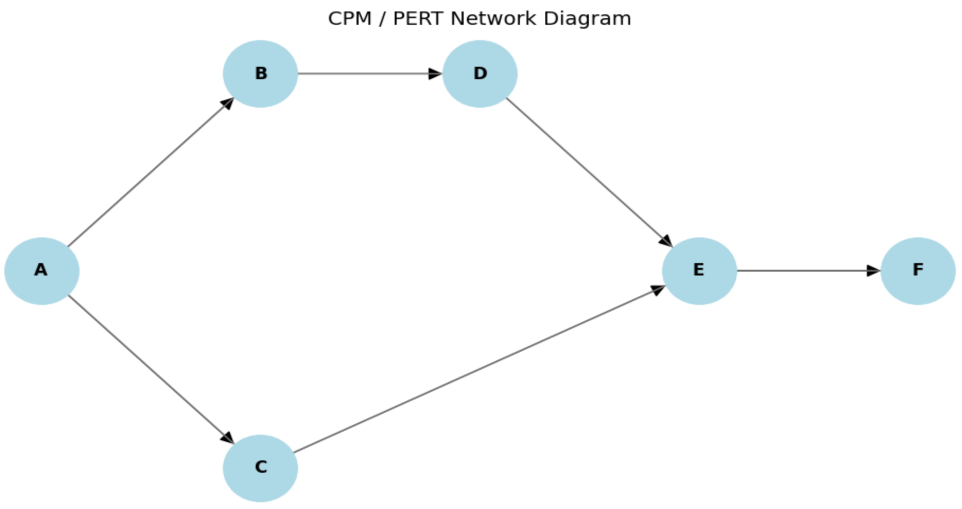
Step 8: Prepare CPM and PERT Charts

Summarize the results in scheduling charts showing activity durations, critical path, and slack times.

Example:

Activity	Predecessor	O	M	P	TE Calculation	TE
A	–	2	4	6	(2 + 4×4 + 6) / 6 = (2+16+6)/6	4.0
B	A	3	5	7	(3 + 4×5 + 7) / 6 = (3+20+7)/6	5.0
C	A	2	3	4	(2 + 4×3 + 4) / 6 = (2+12+4)/6	3.0
D	B	2	4	6	(2 + 4×4 + 6) / 6 = (2+16+6)/6	4.0
E	C, D	3	5	8	(3 + 4×5 + 8) / 6 = (3+20+8)/6	5.2
F	E	2	3	5	(2 + 4×3 + 5) / 6 = (2+12+5)/6	3.2

- **Draw the network with nodes A to F and arrows showing dependencies.**



- **Calculate ES, EF, LS, and LF for each activity.**

Activity	TE	ES	EF	LS	LF	Slack
A	4.0	0.0	4.0	0.0	4.0	0.0
B	5.0	4.0	9.0	4.0	9.0	0.0
C	3.0	4.0	7.0	10.0	13.0	6.0
D	4.0	9.0	13.0	9.0	13.0	0.0
E	5.2	13.0	18.2	13.0	18.2	0.0
F	3.2	18.2	21.4	18.2	21.4	0.0

- **Find slack and identify critical path**

  The critical path includes all activities with zero slack:

  Critical Path: A → B → D → E → F

- **The total project duration is the sum of TE values along the critical path**

  The total duration is the **EF of the last activity (F)**:

  Total Duration = 21.4 units

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.		Open Project, Gantt project 3.3		

X Precautions to be followed

1. Use accurate time estimates to ensure correct project duration.
2. Verify all activity dependencies before drawing the network diagram.
3. Double-check calculations for ES, EF, LS, LF, and slack to avoid errors.

XI Conclusion

……………………………………………………………………………………………

……………………………………………………………………………………………

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is the difference between CPM and PERT techniques?
2. How is the expected time (TE) calculated in PERT?
3. What is a critical path and why is it important in project scheduling?
4. How do you determine slack time for an activity in a network diagram?

[Space for Answer]

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://youtu.be/3dsYrydfMJw
2. https://youtu.be/htms1aOv9v0
3. https://www.youtube.com/watch?v=Us5YtgvfomQ
4. https://www.openproject.org/docs/getting-started/gantt-chart-introduction/

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.16: Track progress of the project using Timeline Charts/ Gantt charts.

I Practical Significance

Tracking the progress of a project is important to make sure everything is going according to the plan. Gantt charts and timeline charts help in showing the progress of different tasks in a visual way. These charts make it easy to see which tasks are completed, which are in progress, and which are yet to start. This helps team members and project managers stay organized and meet deadlines. Using these charts, we can also easily identify delays and take quick action to get the project back on track. Overall, it helps in better planning, time management, and coordination among team members.

II Industry/ Employer Expected Outcome

1. Apply SDLC models for structured software development.
2. Analyze and document software requirements effectively
3. Apply project management techniques in software projects.

III Course Level Learning Outcome

CO 5- Apply project management techniques in software development

IV Laboratory Learning Outcome

LLO 16.1 Monitor the progress of project using timeline/Gantt chart.

V Relevant Affective domain related Outcome(s)

1. Develops a sense of responsibility in tracking and meeting project deadlines.
2. Demonstrates commitment to organized project planning and teamwork.
3. Shows willingness to use tools that improve time management and task tracking.

VI Relevant Theoretical Background

A **Gantt chart** is a popular project management tool that visually represents a project schedule. It shows tasks or activities as horizontal bars along a timeline. The position and length of each bar indicate the start date, duration, and end date of a task. Gantt charts help in planning, coordinating, and tracking specific tasks in a project. They make it easy to see which tasks overlap, the order of activities, and deadlines. A **timeline chart** is similar but focuses more on showing when events or milestones happen over time. These charts are widely used in industries to improve project transparency and communication. Basic knowledge of project phases, task dependencies, and how to use software like Microsoft Excel, Google Sheets, or project management tools (e.g., Microsoft Project, Trello) is enough to create and interpret these charts

VII Exercises

Create a Gantt chart to plan and track the progress of a mini software project using any charting tool (e.g., Excel, Google Sheets, or project management software like Trello or Microsoft Project).

VIII Procedure with Example to Track Project Progress Using Gantt chart:

Steps:

1. **Select a Project:**

   Example: *Online Library Management System*

2. **List Tasks:**

   Example tasks: Requirements, Design, Development, Testing, And Deployment.
   Assign start and end dates for each task, based on how long each will take.

Task	Start Date	End Date
Requirements Gathering	01-June	03-June
System Design	04-June	06-June
Database Design	07-June	08-June
Frontend Development	09-June	13-June
Backend Development	14-June	18-June
Integration & Testing	19-June	21-June
Documentation	22-June	23-June
Deployment	24-June	25-June

3. **Assign Dates:**
   Set start and end dates for each task.
   E.g., Requirements: June 1–3, Design: June 4–6, etc.
4. **Create Gantt chart:**
   Use Excel or Google Sheets to draw bars showing task durations along a timeline.
5. **Update Progress:**
   Mark tasks as Not Started, In Progress, or Completed using colors.
6. **Review & Summarize:**
   Check if tasks are on schedule and write a brief status update.

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Design Tools	Open Project, Gantt project 3.3		

X Precautions to be followed

1. Ensure all task durations and deadlines are realistic and clearly defined.
2. Regularly update the chart to reflect actual progress and changes.
3. Avoid overlapping tasks without checking for resource availability or dependencies.

XI Conclusion

……………………………………………………………………………………

……………………………………………………………………………………

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is a Gantt chart and how is it useful in project management?
2. How do you identify task dependencies in a Gantt chart?
3. Which software tools can be used to create a Gantt chart?
4. How does a timeline chart differ from a Gantt chart?

[Space for Answer]

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://youtu.be/oc6zX6vEWrY
2. https://youtu.be/1y40xTIEKbs
3. https://www.projectmanager.com/guides/gantt-chart
4. https://youtu.be/veAHWAJ10Es

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.17: Prepare SQA plan that facilitates various attributes of quality of process.

I Practical Significance

Preparing a Software Quality Assurance (SQA) plan is very important in any software development project. This plan helps ensure that the final software product is reliable, works correctly, and meets user expectations. By preparing an SQA plan, we can define how to check the quality of the software at each step of the process, from planning to delivery. It helps identify possible problems early, saves time and cost, and improves the overall quality of the product. This practice also builds customer trust, reduces chances of failure, and improves teamwork by setting clear quality goals.

II Industry/ Employer Expected Outcome

1. Create and execute test cases to ensure software quality.
2. Apply project management techniques in software projects.
3. Follow coding standards and documentation practices.

III Course Level Learning Outcome

CO 6- Use quality assurance principles in software development.

IV Laboratory Learning Outcome

LLO 17.1 Prepare SQA plan to ensure various quality processes.

V Relevant Affective domain related Outcome(s)

1. Take responsibility for planning software quality.
2. Respect the need for standards and best practices.
3. Stay committed to improving software processes.

VI Relevant Theoretical Background

Software Quality Assurance (SQA) is a process used to make sure that software products meet certain standards of quality. It involves planned activities like reviews, testing, audits, and process improvement. An SQA plan is a document that outlines these activities. It includes details like the standards to follow, who is responsible for quality tasks, what tools will be used, and how problems will be handled. Important quality attributes include reliability, efficiency, usability, maintainability, and portability. Understanding these basic concepts helps in creating a good SQA plan that supports a quality-driven software development process.

VII Exercises

Prepare an SQA plan for an Online Examination System, highlighting quality attributes such as security, reliability, and usability.

VIII Procedure with Example

Prepare a basic Software Quality Assurance (SQA) plan for a Library Management System (LMS) that highlights quality attributes like reliability, usability, and maintainability.

Steps

1 State the objective of the SQA plan.
2 Identify and list quality attributes (e.g., reliability, usability).
3 Mention applicable standards and guidelines (e.g., coding/documentation).
4 Define SQA activities like testing, audits, and reviews.
5 Assign roles and responsibilities for quality-related tasks.
6 List tools and techniques used (e.g., Git, JUnit).
7 Outline the defect management process for tracking and resolving issues.
8 Include sign-off/approval section for plan validation.

Sample SQA Plan – Library Management System (LMS)

Section	Details
Objective	Ensure quality standards are followed in LMS development.
Quality Attributes	Reliability, Usability, Maintainability
Standards Used	Java coding standards, inline documentation practices
SQA Activities	Unit testing, peer reviews, weekly QA meetings
Roles	*Developer:* Write and fix code. *QA Team:* Testing and bug tracking
Tools	Git (Version Control), JUnit (Testing), Google Docs (Reports)
Defect Process	Bug logged → Assigned → Fixed → Verified → Closed
Approval	Reviewed and signed by QA Lead and Project Manager

IX Required Resources

Sr. No.	Name of Resource	Major Specification	Qty.	Remarks
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Project Management Tools	Open source Software such as Jira.		

X Precautions to be followed

1. Ensure all roles and responsibilities are clearly defined.
2. Follow standard coding and documentation guidelines.
3. Regularly review and update the plan as the project progresses.

XI Conclusion

………………………………………………………………………………………………

………………………………………………………………………………………………

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. List and describe the quality control activities in the SQA plan of a Library Management System?
2. What are the key differences between **preventive** and **corrective** quality assurance activities? Give examples from your SQA plan.
3. List any three quality attributes that should be considered while preparing an SQA plan.
4. What is the purpose of an SQA plan in software development?

[Space for Answer]

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://www.tutorialspoint.com/software_quality_management/software_quality_management_sqa_components.htm
2. https://youtu.be/LNSG-yssisA
3. https://youtu.be/aXD7sh6_YHo

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.18: Prepare SQA plan that facilitates various attributes of quality of product.

I Practical Significance

Creating a Software Quality Assurance (SQA) plan is very important in software development. It helps ensure that the final product is reliable, easy to use, and meets user expectations. The SQA plan lists the steps and checks needed to maintain the quality of the software throughout the project. This includes testing, code reviews, standards to follow, and ways to fix issues early. By preparing a good SQA plan, we can reduce bugs, save time and cost, and deliver a better product that performs well in the real world..

II Industry/ Employer Expected Outcome

1. Create and execute test cases to ensure software quality.
2. Apply project management techniques in software projects.
3. Follow coding standards and documentation practices.

III Course Level Learning Outcome

CO 6- Use quality assurance principles in software development.

IV Laboratory Learning Outcome

LLO 18.1 Prepare SQA plan to ensure quality product.

V Relevant Affective domain related Outcome(s)

1. Shows commitment to quality in software development.
2. Follows and promotes quality standards in a team.
3. Values early defect prevention and improvement.

VI Relevant Theoretical Background

Software Quality Assurance (SQA) is a set of activities that make sure software meets quality standards. It is part of software engineering and focuses on improving the development process to avoid problems. Key concepts include quality attributes like reliability, usability, maintainability, and performance. The SQA plan outlines roles, responsibilities, tools, methods, and processes used to check quality at each stage of development. Understanding basic software life cycle models, testing types, and quality standards like ISO or CMMI helps in preparing an effective SQA plan.

VII Exercises

Prepare an SQA plan for a simple software project that ensures quality attributes like reliability, usability, and maintainability.

VIII Procedure:

1. **Understand the project:**
   Choose a simple software project, such as a "Student Attendance Management System."
2. **Identify quality attributes:**
   List key quality attributes you want to ensure, e.g., reliability (system works

correctly), usability (easy to use), maintainability (easy to update).

3. **Define SQA activities:**
   Decide on activities like code reviews, testing types (unit, integration, system), documentation, and standards to follow.
4. **Assign roles and responsibilities:**
   Specify who will perform each SQA activity (tester, developer, reviewer).
5. **Schedule and tools:**
   Plan when each activity will happen and mention tools to use (e.g., test management software).
6. **Write the plan:**
   Prepare a simple document outlining all above points.

E.g. Simple SQA Plan Handout for Project: Student Attendance Management System

Category	**Details**
Project	Student Attendance Management System
Quality Attributes	Reliability, Usability, Maintainability
SQA Activities	- Code reviews before merging
	- Unit testing for each module
	- Usability testing with sample users
	- Document coding standards
Roles	- Developer: Write code and tests
	- Tester: Execute tests and report bugs
	- Lead: Review code and approve releases
Schedule	- Code reviews weekly
	- Testing at end of each sprint
Tools	- Git for version control
	- JIRA for bug tracking

IX Required Resources

Sr. No.	**Name of Resource**	**Major Specification**	**Qty.**	**Remarks**
1.	Computer System	Any desktop or laptop computer with CPU> i3 and RAM 4GB onwards	One computer system for each student	
2.	Software Package	MS Word, Google Docs, Notion		
3.	Software Project Management Tools	Open source Software such as Jira.		

X Precautions to be followed

1. Always keep backups of all project files and documents.
2. Follow coding standards strictly to avoid errors.
3. Test code thoroughly before merging into the main branch..

XI Conclusion

……………………………………………………………………………………………

……………………………………………………………………………………………

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What are the key quality attributes considered in your SQA plan and why?
2. How does your SQA plan ensure the reliability of the software?
3. Explain the role of testing in achieving software quality.
4. Why is documentation important in SQA planning?

[Space for Answer]

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

XIII References / Suggestions for further Reading Software/Learning Websites

1. https://youtu.be/p8SYmtuJv10
2. https://www.youtube.com/watch?v=D2Y-2RO2fsQ
3. https://www.geeksforgeeks.org/software-engineering-software-product/

XIV Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1.	Requirement analysis and documentation skills	20%
2.	Code/Design and modularity of the software system	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		**40%**
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
	Total : 25 Marks	**100%**

Marks obtained			**Dated Sign of Teacher**
Process Related (15)	**Product Related (10)**	**Total (25)**	