Software Engineering Assignment

Name: Abdus Salam
ID: IT-21016 Session: 2020-2021
3rd Year 2nd Semester Dept: ICT

Question

1.) In a Scrum-based software development project, the Product Owner has defined the following user stories for an e-commerce application:

- As a user, I want to log in securely so that I can access my ~~acc~~ account.
- As a user, I want to search for products by category to find items easily.

1) Create a product backlog for these user stories by breaking them into tasks.

Ans: Product Backlog Breakdown

Story 1: 1. Design the login UI (Frontend)
2. Implement user authentication (Backend)
3. Store user credentials securely (Database setup)
4. Implement session management (cookies, JWT tokens)
5. Implement password recovery functionality (OTP)
6. Perform security testing (SQL Injection)

Story 2:

1. Design the product search UI
2. Implement product category classification (Backend & ~~Frontend~~ Database)
3. Develop search API (Filters)
4. Implement sorting and filtering features.
5. ~~Code~~ Conduct usability testing.

ii) Describe how the development team can prioritize these user stories during a Sprint Planning meeting, considering value to the customer and technical feasibility.

~~Ans:~~ Sprint Planning Prioritization

The development team can prioritize tasks based on:

- Customer Value: Logging in securely is essential before searching for products, so User Story 1 is more critical.
- Technical Feasibility: If the team encounters security implementation challenges, they can parallelly start the product search system.

(ii) Illustrate how these tasks will be tracked using a Scrum board. Use proper terms like "TO DO", "In Progress", and "Done".

Ans: Scrum Board (Task Tracking)

Task	TO DO	In Progress	Done
Design Login UI	OK		
Implement Authentication		OK	
Store User credentials Securely			OK
Develop Search API	OK		
Usability Testing			OK

Question:

2) A software development team is about to start a project for a new innovative product. The project has several high-risk components due to its novelty and there's uncertainty regarding the client's future needs. The client is open to iterative changes, but the team must ensure that the software evolves in a manageable, cost-effective way.

- Considering the high risks and the evolving nature of the client's needs, discuss how the Spiral, Agile and Extreme methodologies address risk management and adaptability. Which methodology would be the most suitable for a project with significant risk and evolving requirements, and why?

Ans: A project with high risk and evolving client needs requires a development approach that manages risk effectively while allowing flexibility for

changes, Here's how different methodologies handle this:

1. Spiral Model
   - Best for projects with high risks (e.g. security)
   - Focuses ~~for~~ on risk analysis at every phase before development.
   - Slower and more expensive due to repeated risk.

2. Agile Methodology
   - Best for projects with evolving requirements and continuous client feedback.
   - Uses short Sprints (iterations) to deliver working software frequently.
   - High flexible - changes can be made anytime.
   - cost- effective because only valuable features.

3. Extreme Programming (XP)
   - Focus on fast iterations (1-2) and continuous testing.
   - Requires constant customer involvement and pair programming.
   - Best for small teams working on projects that need quick changes.

Agile is the best choice for this project because:

→ It allows frequent updates to match changing client needs.

→ It reduces risk with regular testing and feedback.

→ It ensures cost-effective development by focusing on high-value features.

Question-3:

A company is working on two different projects. Project A has well-defined requirements and strict deadline, while Project B has evolving requirements with an uncertain timline and continuous customer feedback. Both projects involve high stakes, and the team must decide which development methodology to use.

- Compare and contrast the Waterfall, Agile, Extreme, and Spiral development models. Based on the characteristics of both projects (Project A & B), which methodology would best suit each?

Ans: Comparison of Software Development Models:

Model	Best For	Key Features	Pros	Cons.
Waterfall	well-defined, fixed requirements	Linear, step-by-step process	Predictable, easy to manage	Rigid, no flexibility for changes.
Agile	Evolving requirements, continuous feedback	Iterative, short sprints	Highly ~~Flexible~~ Flexible, frequent + feedback	Less predictable timeline
Extreme Programming (XP)	Small teams, rapid changes	Continuous testing, pair programming	Quick delivery, frequent updates	Requires constant client involvement
Spiral	High-risk projects	Risk analysis in each phase	Reduces risks, flexible	Expensive and time-consuming

Best Methodology for Each Project

1. Project A (Fixed requirements, strict deadline) → Waterfall
   - Predictability
   - Structured approach
   - Better for documentation & regulatory compliance.
2. Project B (Evolving requirements, continuous feedback) → Agile
   - Flexibility
   - Customer collaboration
   - Faster delivery of working software.

Question - 4

Explain the principles of software engineering ethics, highlighting the issues related to professional responsibility. Discuss how the ACM/IEEE Code of Ethics guides ethical decision-making in software engineering practices.

Ans: Principles of Software Engineering Ethics

Software engineering ethics ensure responsible and professional conduct in software development. Key principle include:

1. Public Interest: Prioritize user safety, privacy.
2. Client and Employer Responsibility
3. Product Quality
4. Fairness and Honesty
5. Intellectual Property
6. Professional Competence
7. Respect for Colleagues
8. Social Responsibility

Professional Responsibility Issues

- Data Privacy Violations
- Software Failures
- Unethical Coding Practices
- Intellectual Property Theft

ACM/IEEE Code of Ethics

The ACM (Association for computing Machinery) and IEEE (Institute of Electrical and Electronics Engineers) Code of Ethics provide guidelines to ensure ethical decision-making:

1. Act in the Public Interest
2. Be Honest and Trustworthy
3. Maintain Product Quality
4. Respect Confidentiality
5. Professional Growth.

Question - 5:

Given the story of the Airport Reservation System, identify at least five functional and five non-functional requirements for the system. In your answer, explain how each requirements contributes to the overall performance, usability, and security of the system. Consider factors such as performance, user experience, and system maintenance in your discussion.

Ans:

Airport Reservation System Requirements

Functional Requirements (Define what the system should do)

1. User Authentication - Passengers and staff must log in securely to access features. Enhances security by preventing unauthorized access.
2. Flight Booking & Cancellation - Users can search, book, and cancel flights. Improves user experience by providing flexibility in reservations.

3. Payment Processing – Supports secure transactions using credit cards. Ensure safe and efficient financial transactions.

4. Seat Selection :– Users can choose preferred seats based on availability. Enhances usability by offering personalized option.

5. Check-in and Boarding Pass Generation – Online check-in and e-boarding passes.

Non-Functional Requirements (Define system quality attributes)

1. Performance – The system should process booking within 2 seconds.

2. Security – Implements encryption for user data and payment details.

3. Scalability – Supports increasing users during peak travel seasons.

4. Usability – The interface must be user-friendly and intuitive.

5. Reliability – The system should have 99.9% uptime

Question-6

Illustrate and explain the V-model of testing phases in a plan-driven software process, detailing the relationships between development activities and corresponding testing activities.

Ans:

V-Model of Testing in Software Development.

The V-Model (Verification and Validation Model) is a plan-driven software development approach where testing activities run parallel to development phases. The "V" shape represents the relationship between development and testing phases.

Phases of the V-Model

1. Verification Phases (Development Stages on the Left Side)

a. Requirements Analysis → Acceptance Testing
- User needs are gathered.

b. System Design → System Testing
- High-Level design of system components is created

c. Architectual Design → Integration Testing
- Software is divided into modules and there interactions are defined.

d. Module Design → Unit Testing
   • Individual modules are designed.

2. Validation Phase (Testing Stages on the Right Side)
   • Each development phase has a corresponding testing phase to validate correctness before moving forward.

Advantages of V-Model

→ Early defect detection
→ Structured approach
→ Ensure quality

Question-7

Explain the process of prototype development in software engineering. Discuss the key stages involved in creating a prototype and how it helps in refining software requirements. Analyze the benefits of using the prototyping model, particularly in terms of user feedback, risk reduction, and iterative development.

Ans: Prototype Development in Software Engineering

Prototyping is a software development approach where an initial working model (prototype) of the system is built to gather user feedback and refines requirements before full-scale development.

Key Stages in Prototype Development

1. Requirements Gathering (Identify initial user needs and system requirement)
2. Quick Design (Create a basic design)
3. Prototype Development (working model)

4. User Evaluation & Feedback (Users test)

5. Refinement & Iteration (updated)

6. Final Development

Benefits of Prototyping Model

→ Improves Requirements Accuracy

→ Reduces Risks

→ Enhances User Satisfaction

→ Encourages Iterative Development

Question - 8

Explain the process improvement cycle in software engineering and describe its key stages. Name and explain some commonly used process metrics, highlighting how they help in monitoring and improving software processes.

Ans:

Process Improvement Cycle in Software Engineering

The process Improvement Cycle is a structured approach to enhancing software development processes to improve quality, efficiency, and performance. It follows a continuous PDCA (Plan - DO - Check - Act) cycle:

Key Stages of Process Improvement Cycle

1. Plan
   - Identify areas for Improvement in the development process.
2. Do (Implement Changes)
   - Apply process improvements on a small scale.

3. Check (Measure & Analyze)
   - Monitor the effectiveness of changes using metrics.

4. Act (Optimize & Standardize)
   - Refine and scale successfull changes across projects.

Commonly Used Process Metrics

1. Defect Density → Measure the number of defects per kLOC (thousand lines of code).
2. Cycle Time → Measure the time taken from development to deployment.
3. Code Coverage → Percentange of code tested by automated tests.
4. Customer Satisfaction Index → Based on user feedback and defect reports.
5. Productivity Rate → Measures the number of features delived per sprint.

Question – 9

Explain the Software Engineering Institute Capability Maturity Model (SEI CMM) and its five levels of capability and maturity. Analyze how each level contributes to improving the software development process and organizational performance.

Ans:

The Capability Maturity Model (CMM) is a framework developed by the Software Engineering Institute (SEI) to assess and improve software development processes. It consists of five levels, each representing an increasing maturity in software development practices.

Five Levels of SEI CMM and Their Contributions

1. Level 1: Initial (Chaotic)
   - Process: Unpredictable, reactive, and unstructured.
   - Issues: High project failure rate.
   - Improvement: Organizations must stabilize processes to reduce risks.

2. Level 2 : Repeatable

- Process: Basic project management practices are established.
- Issues: Still lacks standardization across teams.
- Improvement: Focus on requirements management, cost, and schedule tracking.

3. level 3: Defined

- Process : Organization-wide standardized processes are defined.
- Issues : Teams need to consistently follow documented.
- Improvement: Better training, documentation.

4. Level 4: Managed

- Process: Quantitative measurements and data-driven
- Issues: Performance metrics need to be effectively used
- Improvement: Predictable quality and process control.

5. Level 5: Optimizing

- Process: Continuous process improvement and innovation.
- Issues: Requires proactive adaptation to new challenges.
- Improvement: Focuses on defect prevention, efficiency, and automation.

Question-10

Describe the core principles of agile software development methods. Analyze how these principles are applied in different software development environments and assess the benefits and challenges of using agile methods in various project types and organizational setting.

Ans:

Agile software development is based on flexibility collaboration, and iterative progress. The core principles from the Agile Manifesto include:

1. Customer Collaboration Over Contract Negotiation.
2. Individuals and Interactions Over Processes and Tools.
3. Working Software Over Comprehensive Documentation
4. Responding to Change Over Following a Plan.

Application in Different Environments

- Startups & Fast-Paced Environments
- Large Enterprises .

- Mission-Critical Systems (e.g. Healthcare, Banking)

Benefits of Agile

→ Faster Delivery

→ Better Customer Satisfaction

→ Higher Flexibility

→ Improved Team Collaboration.

Question-11:

Ans:

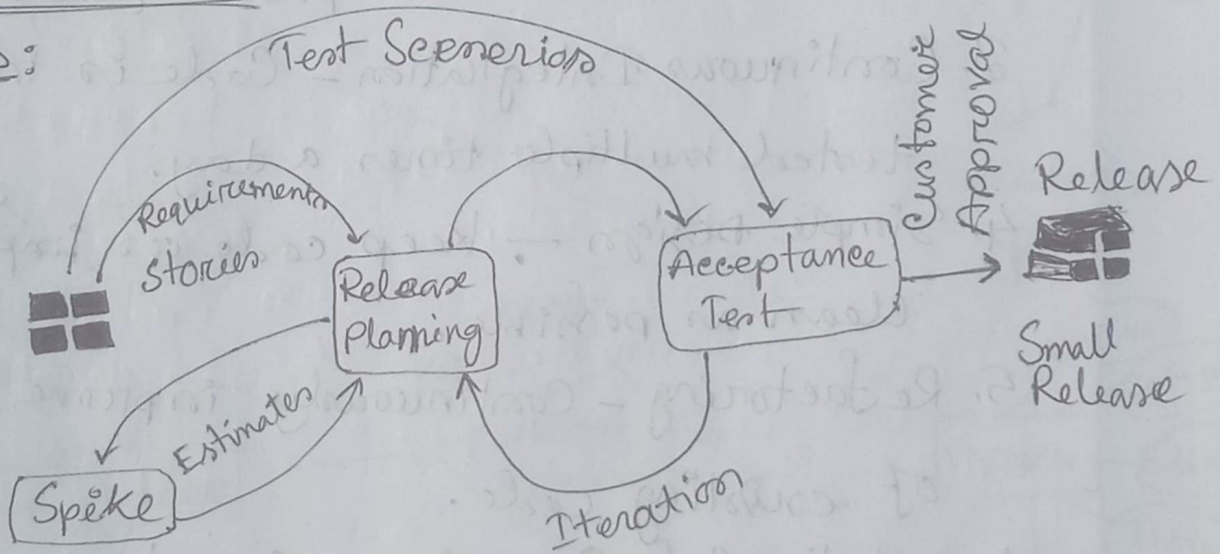


Fig: Release Cycle of Extreme Programming (XP)

Influential Programming Practices in XP

Extreme programming emphasizes high-quality code and teamwork using key engineering practices such as:

1. Test-Driven Development (TDD) – Write tests before writing code
2. Pair Programming – Two developers work on the same code to improve quality.
3. Continuous Integration – Code is integrated and tested multiple times a day.
4. Simple Design – Keep code as simple and clean as possible.
5. Refactoring – Continuously improve the structure of existing code.
6. Collective Code Ownership – Any developer can modify any part of the code.

Question-12

Ans:

Entity-Relationship Diagram (ERD) for Library Management System

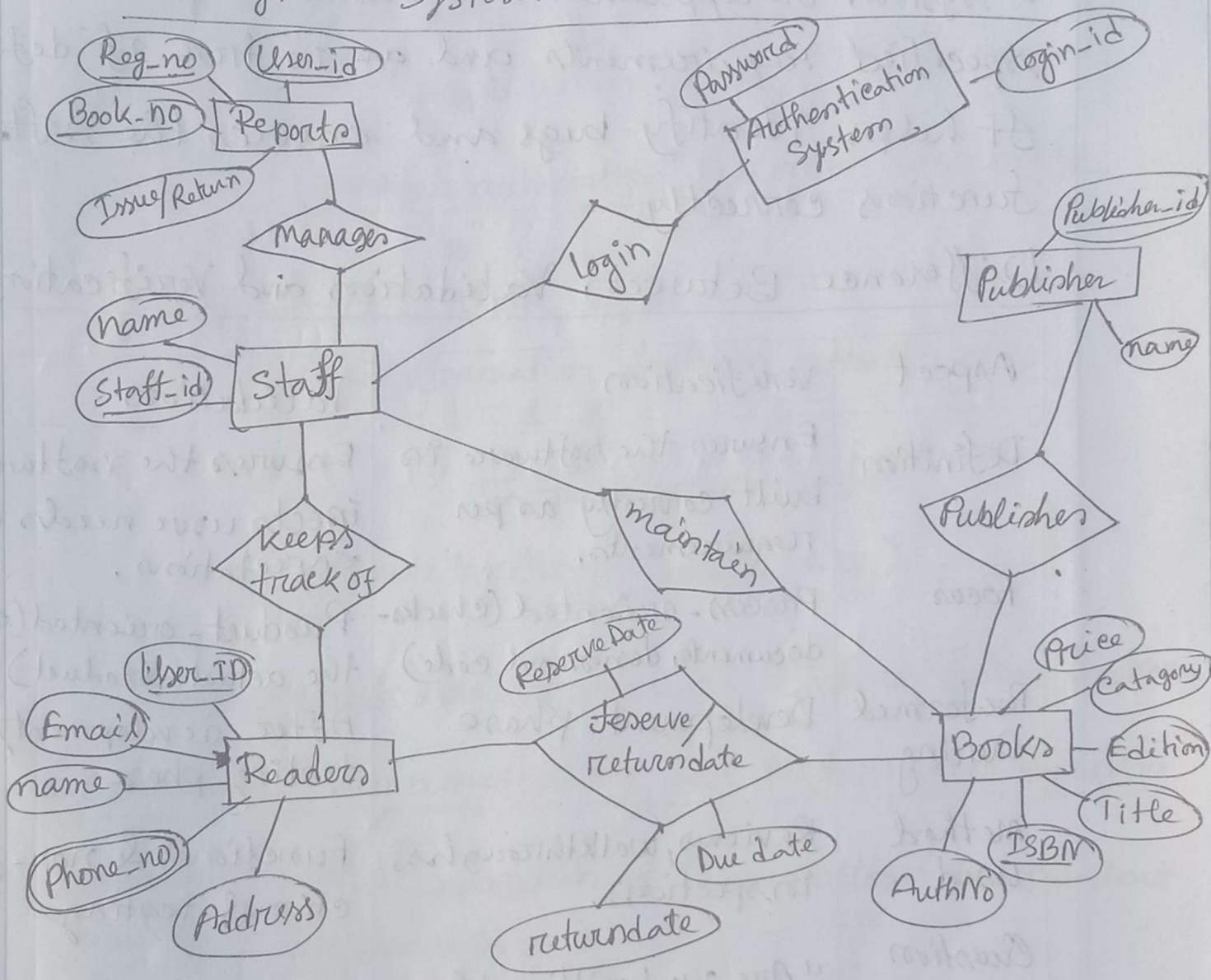


Fig: ERD (Entity-Relationship Diagram) for Library Management System.

Question-13

Ans:

Software Testing is the process of evaluating a system or application to ensure it meets the specified requirements and as is free of defects. It helps identify bugs and ensures the software functions correctly.

Difference Between Validation and Verification.

Aspect	Verification	Validation
Definition	Ensures the software is built correctly as per requirements.	Ensures the software meets user needs and expectations.
Focus	Process-oriented (checks documents, design and code)	Product-oriented (checks the actual product)
Performed During	Development phase	After development/ testing phase
Method Used	Reviews, walkthroughs, inspections.	Functional & non-functional testing
Question Answered	"Are we building the product right?"	"Are we building the right product?"

Question - 14

Answer:

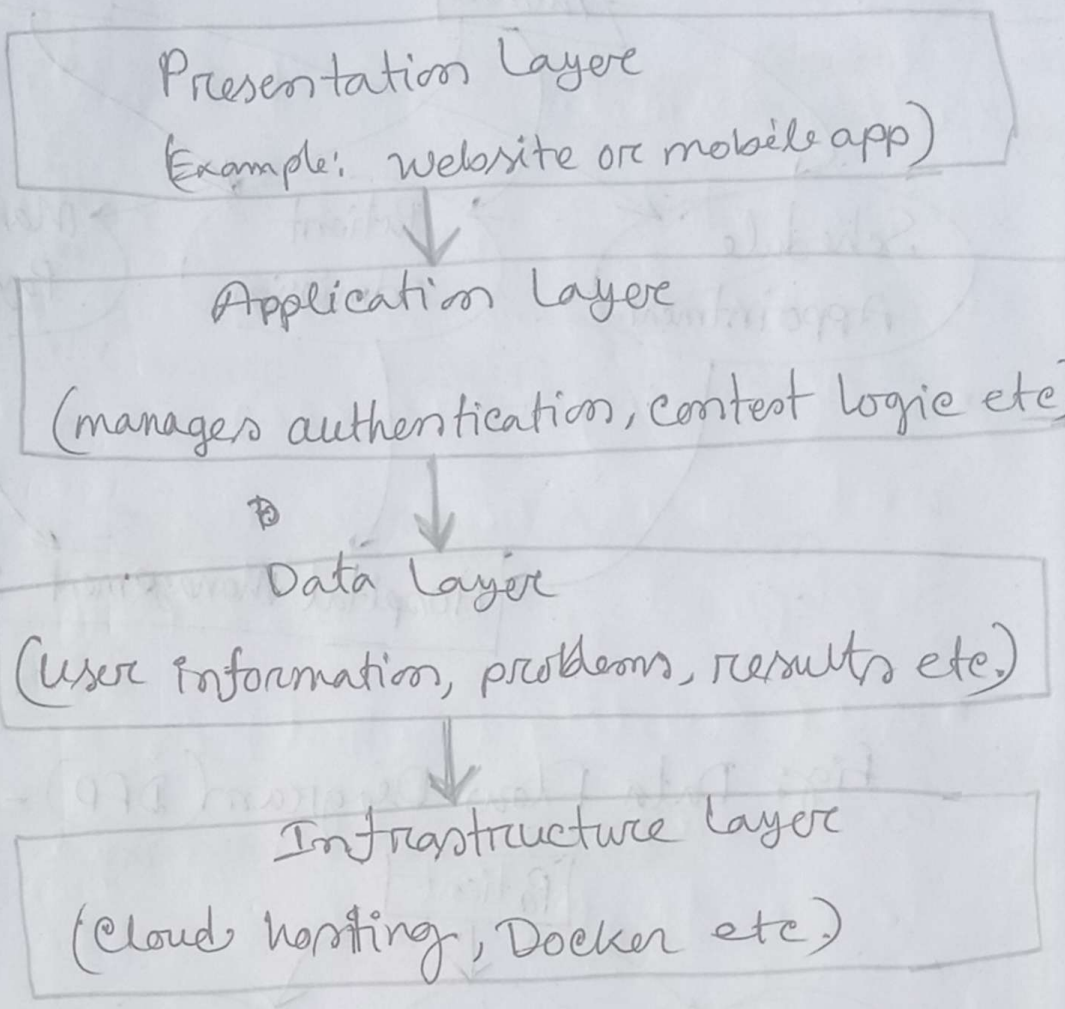


Fig: Layered Architecture of Online Judge System.

Scalability, Maintainability, and Performance Benefits

Scalability: The separation of layers allows independent scaling of components.

Maintainability: Each layer is modular, making it easier to update without affecting other parts of the system.

Performance: The business logic layer optimizes resource-intensive tasks, while caching and indexing.

Question - 15

Answer:

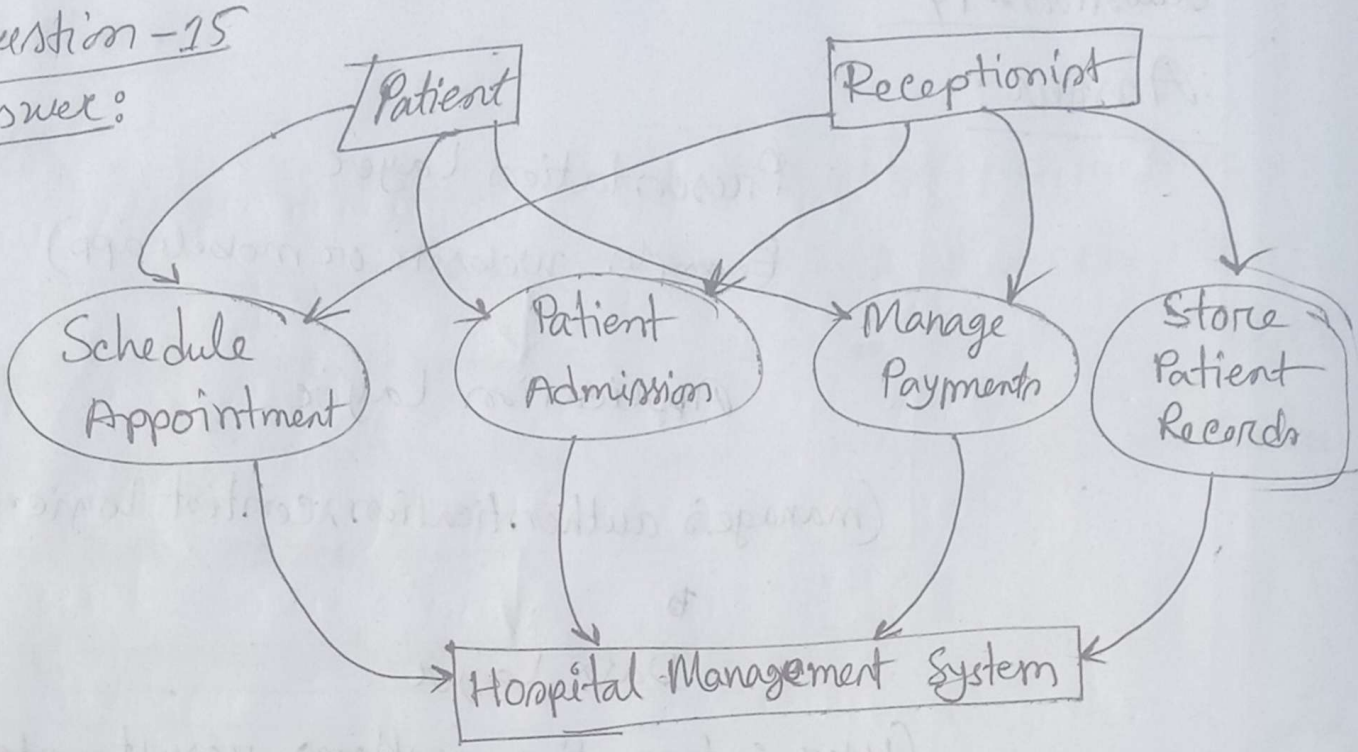


Fig: Data Flow Diagram (DFD) - Level 0.

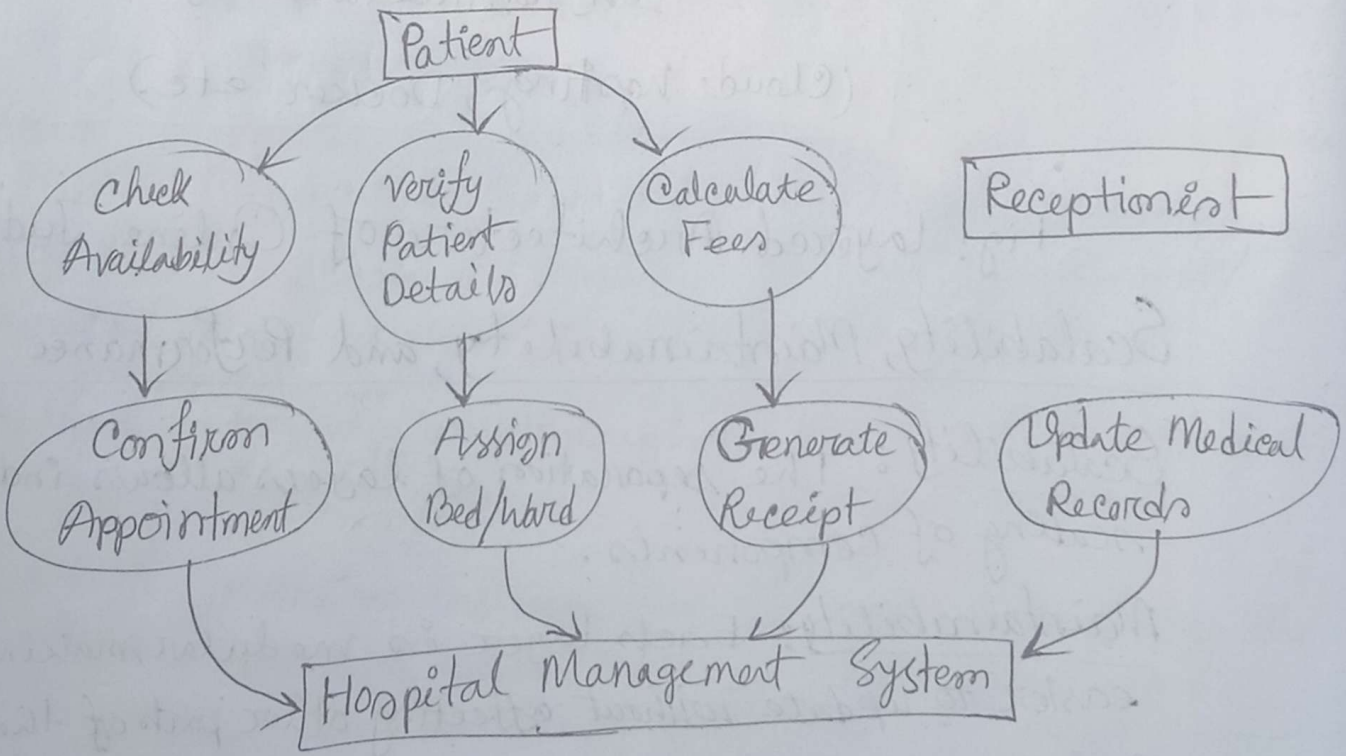


Fig: Data Flow Diagram (DFD) - Level 1.

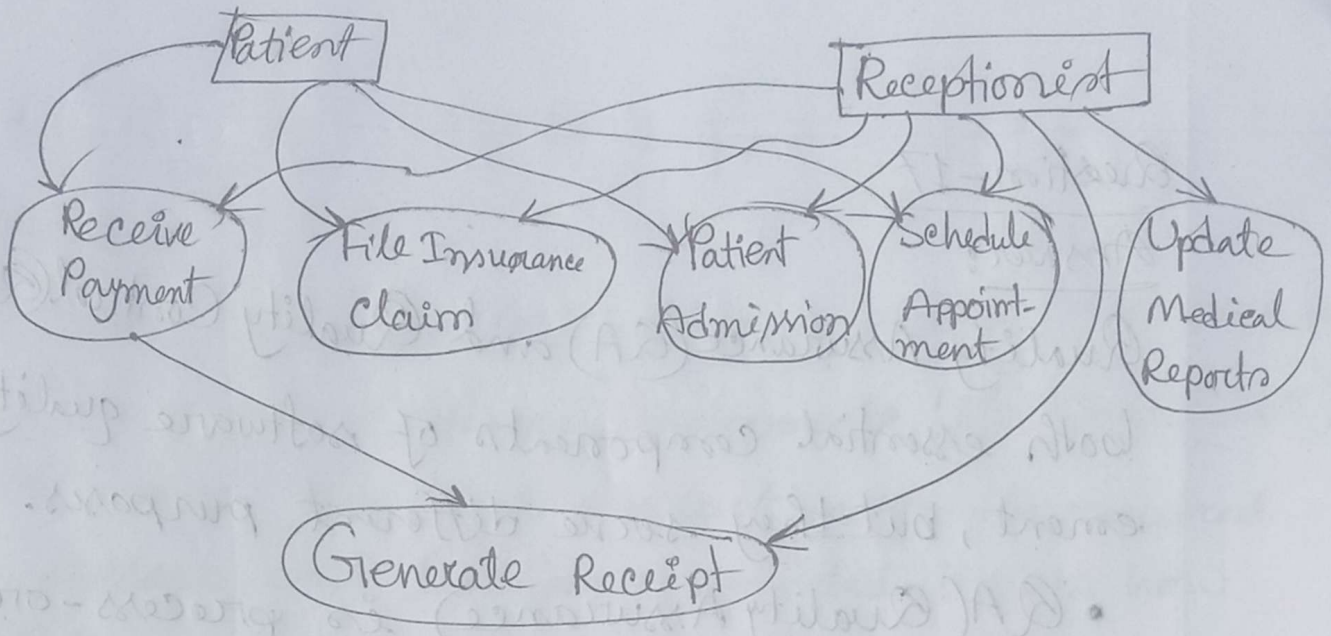
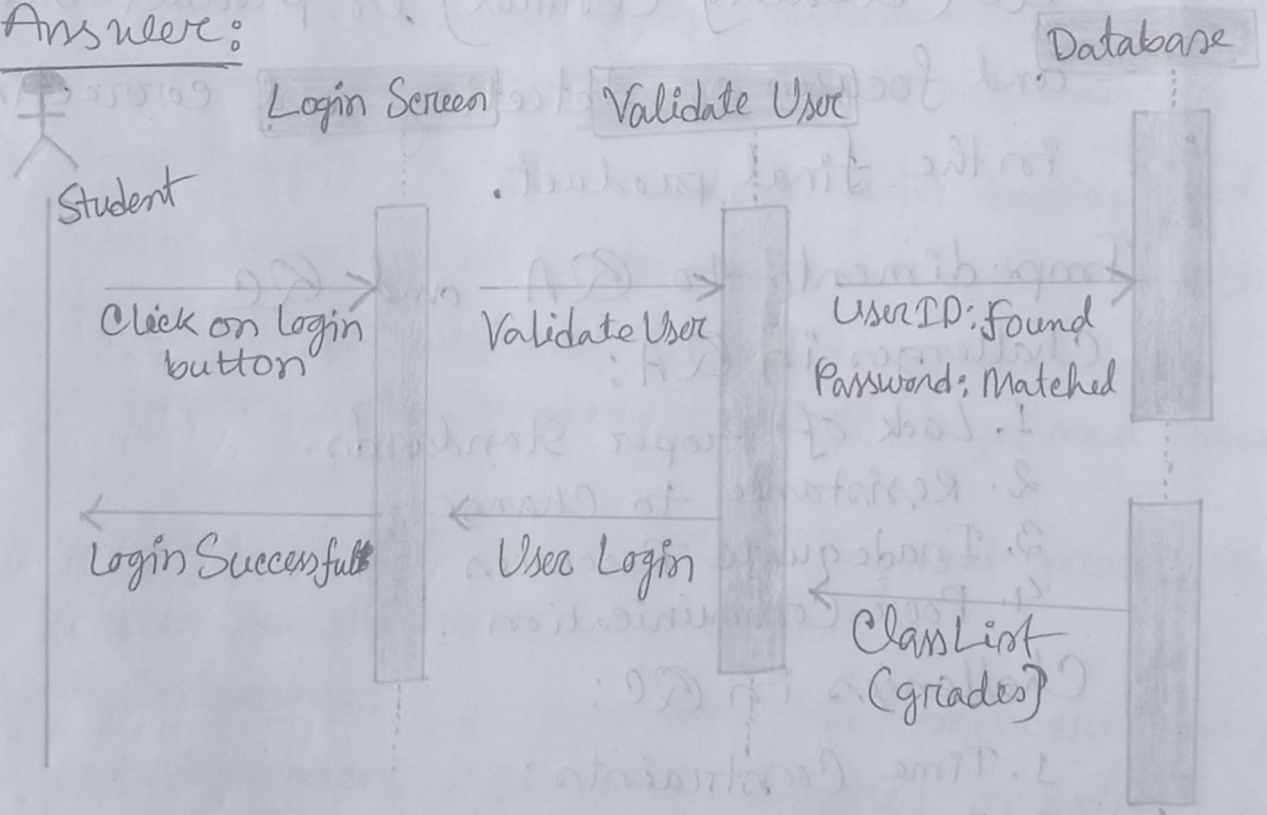


Fig: UML Use Case Diagram

Question :- 16

Answer:



Which is a UML Class diagram for the scenario depicted by the Sequence Diagram.

Question-17

Answer:

Quality Assurance (QA) and Quality Control (QC) are both essential components of software quality management, but they serve different purposes.

- QA (Quality Assurance) is process-oriented and focuses on preventing defects by improving development processes and methodologies.
- QC (Quality Control) is product-oriented and focuses on detecting and correcting defects in the final product.

Impediments to QA and QC

Challenges in QA:

1. Lack of Proper Standards.
2. Resistance to Change.
3. Inadequate Resources.
4. Poor Communication.

Challenges in QC:

1. Time Constraints.
2. Incomplete Test Coverage.
3. Cost of Fixing Bugs Late.
4. Human Errors.

Question – 18

Answer:

Quality Assurance (QA) plays a crucial role in each phase of the Software Development Life Cycle (SDLC) to ensure the final product meets the required standards and functions as expected. Here's how QA contributes at each phase:

1. Requirement Analysis: QA reviews requirements to ensure they are clear, complete, and testable.
2. Design: QA assesses design documents for potential issues and ensures they align with requirements.
3. Implementation: QA conducts unit testing and works closely with developers to catch issues early.
4. Testing: QA performs various tests (functional, integration, system, acceptance) to identify bugs and ensure the product.
5. Deployment: QA verifies the deployment process to ensure smooth rollout and stability.
6. Maintenance: QA monitors the software for any post-deployment issues and ensures ongoing quality during updates.