Question - 01

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 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.
2. Describe how the development team can prioritize these user stories during a sprint planning meeting Considering value to the customer and technical feasibility.
3. Illustrate how these tasks will be tracked using a scrum board. Use proper terms like "To Do", "In Progress" and "Done".

Answer:

(1)

For the e-commerce application, the product Backlog is created by breaking the user stories into actionable tasks. These tasks are small enough to be worked on during a sprint and will help the development team complete each user story.

User Story: As a user, I want to log in securely so that I can access my account.

Task 1: Design the login page UI (e.g. fields for username, password and submit button)

Task 2: Implement client-side validation for login (e.g. check if fields are empty, validate email format)

Task 3: Set up backend authentication API (e.g. using JWT or session-based authentication)

Task 4: Implement password hassing and secune storage (e.g. using bcbypt or Argon 2)

Task 5: Implement session management.

Task 6: Write unit tests for the login functionality.

Task 7: Perform security testing.

Task 8: Implement logout

User Story 2: As a user, I want to search for products by category to find items easly.

Task 1: Design the search VI with category dropdown on filter panel

Task 2: Create a Database schema for product categories

Task 3: Implement search API to filter products by selected category

Task 4: Integreate frontend with the search API to display products dynamically based on selected categ

Task 5: Write until tests for product search functionality

Task 6: Optimize search query for performance.

Task 7: Test the search functionality for both accuracy and speed.

(2)

During Sprint Planning, the team will prioritize these user stories based on:

- Value to the customer:

User story 1 (Login): High priority secure log in is crucial for user trust and account access, enabling core functionalities like order history, saved items and personalized recommendations.

User story 2 (Category Search): High priority Effective product discovery is essential for successful e-commerce platform. It enhances user experience and increases the likelihood of finding desired products.

Technical feasibility:

User Story 2 (Login): Moderate complexity Requires Carefull implementation of security measures and integration with authentication services.

User Story 2 (Category Search):

Moderate to high complexity depending on the complexit of the product catalog and the desired search features.

(3)

The scrum board will track the progress of tasks during the sprint. The board consists of columns to visualize the workflow such as "To Do", "In Progress", and "Done".

Columns:

1. To Do: Tasks that have not yet started and are in the planning stage
2. In Progress: Tasks that the development team is actively working on.
3. Done: Tasks that are completed and meet the Definition of Done (DoD) ready for review on deployment

To Do	In Progress	Done
User story 1 (login):	User story 1 (login)	User story (login)
Design the logic page UI	Implement frontend validation	Write unit test for login
Implement password hashing and encryption	Implement session and management	Implement password encryption
Conduct security testing	Write unit tests for login	
User story 2 (search):		User story 2 (search):
Design search UI with category dropdown		Write unit tests for search.

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 it's novelty. and there's uncertainly regarding the client's future needs. The client is open to interactive changes, but the team must ensure that the software evalves in a managable cost-effective way.

considering the high risks and the evalving

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?

Answer:

Risk Management and Adaptability in Spiral, Agile and Extreme Methodologies:

* Special Methodology: Focuses on risk management through frequent cycles of planning, risk analysis, and refinement. It's ideal for high-risk projects, because it proactively identifies and mitigates risks.

* Agile methodology: Uses short iterations (sprints) and constant client feedback, allowing for early identification of risks and frequent adjustments based on evolving requirements.

* Extreme Programming: Emphasizes continuous integration and test-driven development, ensuring high-quality code and minimizing defects.

Given that the project in question has high-risk components and there is uncertainity regarding future client needs, Agile would likely be the most suitable methodology. Here's why:

1. Continuous Risk Mitigation:

Agile's frequent feedback loops allow the team to identify and address risks early. Risks are managed through short iterations, with each Sprint serving as an opportunity to adjust the approach based on client feedback and evolving requirements.

2. Flexibility and Adaptability: Agile's iterative proces allow for flexibility, making it easy to adapt to new information or changes in client needs. The

Client is involved throughout the process.

3. Cost - Effective Evolution :

Agile helps in delivering incremental improvements. This iterative approach ensures that the project evolves in a manageable.

Why not Spiral or XP?

- Spiral is a great risk management approach, but it can be more formal and document-heavy than Agile, which might slow down adaptation, especially in a fast-changing, high-uncertainty project.
- XP is highly effective in ensuring high-quality code, but it may be overkill for a project where the client's evolving needs are the primary concern.

Agile is the most suitable methodology for a project with significant risks and evolving requirements. Its ability to incorporate

frequent client feed back, adjust priorities iteratively and manage risks.

Comparison of Waterfall, Agile, Extreme and Spiral Development models:

1. Waterfall Methodology:

• Overview: Waterfall is a linear and sequential development model. It involves distinct phase: requirements, gathering, design, implementation, testing, deployment and maintenance.

• Characteristics:

Predictability: Highly predictable for well-defined projects because each phase is completed before the next begins.

• Customer collaboration: Minimal customer involvement after the requirements phase.

• Risk management: Limited risk management. Risks are addressed primarily only at the beginning and end.

2. Agile Methodology:

- **Overview:** Agile is an iterative and incremental approach. Development is split into short cycles (sprints) and feedback is continuously incorporated into the process.

- **Characteristics:**

- **Predictibility:** Less predictable in terms of timeline and scope as priorities can change during each sprint.
- **Customer Collaboration:** High customer involvement with regular feedback loops to adjust features based on client needs.
- **Risk Management:** Focuses on managing risks by addressing them early in the process.

3. Extreme Programming (XP):

- **Overview:** XP is an Agile methodology that emphasizes technical excellence, continuous integration collaboration and frequent releases.
- **Characteristics:**

Predictability: Less predictable due to frequent

• Customer collaboration: very high customer involvemen with constant feedback and changes based on user needs.

• Risk management: strong emphasis on reducing technical risks through practices like test driven development (TDD) and pair programming.

4. Spiral Methodology:

• Overview: spiral is a risk driven model that combine iterative development with regular risk assessment It involves

• Characteristics:

• Predictability: offers moderate predictability as it breaks the project into smaller, manageable, cycles, but requires ongoing assessments.

• Customer collaboration: Regular feedback and iteration allow for high collaboration with the customer, ensuring their needs are met.

• Risk Management: Emphasizes proactive risk management by identifying and addressing risks at every phase of the cycle.

Question-3:

A company is working on two different projects. project A has well-defined requirements and a strict deadline, while project B has evolving requirements with an uncertain timeline and continuous customer feedback. Both projects involve stakes.

- Compare and constract the waterfall, Agile, Extreme and spiral development models, which methodology would best suit each? Support your answer with a detailed analysis of how each methodology would address the specific needs of the projects considering factors such as: predictability, customer collaboration, and risk management.

Answer:

Project A (well-defined requirements, strict deadline):

- Best methodology: Waterfall
- Why: Since project A has well-defined requirements and a strict deadline, waterfall is the most suitable methodology. Its predictability

and structured approach make it ideal for projects where the requirements are clear from the start and changes are minimal.

- How it addresses the project:
- Predictability: Waterfall's clear, linear progression ensures that the project stays on schedule with clear milestones.
- Customer collaboration: minimal, as requirements are set at the beginning.
- Risk management: While waterfall does not excel in handling changes once the project has started.

its st.

- Project B (Evolving requirements, uncertain timeline, continuous feedback):

Best methodology: Agile or spiral (with a slight preference for Agile)

- Why methodology: Agile or spiral (with a sl.

Project B requires continuous customer feedback and has evolving requirements, making both Agile and Spiral suitable. However, Agile is slightly better suited because of its flexibility and adaptability.

- How it addresses the project:

: Predictability: Agile is less predictable, but this is acceptable given the uncertain timeline and evolving nature of the project.

- Customer Collaboration: Agile emphasizes close collaboration with the customer, allowing them to provide feedback at the end of each sprint.

- Risk manggement: Risks are addressed continously in Agile by identifying and resolving issues during each sprint.

- Spiral Alternative: While spiral also allows for continuous feedback and risk management, it may be more formal.

- Summary of methodology suitability:

- Project A: waterfall is the most suitable methodology due to its predictability, structure and ability to meet deadlines with well-defined requirements.

- Project B: Agile is the best fit for its flexibility customer collaboration and ability to adapt to evolving requirements.

Question-4:

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

Answer:

Principles of software Engineering Ethics:

Software Engineering ethics emphasizes the responsibility of engineers to ensure their work prioritizes the safety, well-being and privacy of users as well as the broader social impact.

1. Public welfare: Software engineers must prioritize the public's safety and interests, ensuring their work does not cause harm as unethical consequence

2. Professional competence: Engineers should work within their areas of expertise, continuously improving their skills to deliver high-quality work.

3. Honesty and Integrity: Transparency, honesty and integrity in communication, reporting progress.

4. Confidentiality: Engineers must respect the confidentiality of client.

Issues related to professional responsibility:

- Accountability: Engineers must be accountable for their actions and decisions, ensuring the software is reliable and safe.
- Conflict of Interest: Engineers should avoid situations where personal or financial interests.
- Privacy and security: Protecting user data and ensuring compliance with privacy laws is a critical responsibility.
- Inclusivity: Ensuring software is accessible to all users, regardless of physical or cognitive limitations, is part of professional responsibility.

ACM/IEEE code of Ethics and Ethical Decision Making:

The ACM/IEEE code of Ethics provides a framework for ethical decision making by offering principles that guide software engineers in making responsible choices.

1. Public interest: Engineers are encouraged to act in the public's best interest, ensuring their work is safe.

2. Competence and Quality: Engineers must produce high-quality, well-tested and reliable software.

3. Honesty and Transparency: The code stresses the importance of being truthful, reporting issues early, and

4. Respect for others: It encourages respect for colleagues, clients, users and society.

5. Professionalism: The code supports ethical conduct throughout one's career, promoting lifelong learning and integrity in the face of challenges.

Question-5:

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

Answer:

The airport reservation system (ARS) is designed to manage flight bookings, passenger reservations, and other related services in an airport or airline Below are five functional and five non-functional requirements for such a system:

Functional Requirements:

1. User Registration and Login: Allows users to create and manage accounts, ensuring secure access to bookings and personal details.

2. Flight search and Booking: Enables users to search for flights and books tickets based on various criteria.

3. Payment Processing: secure Financial transactions for bookings.

4. Flight Reservation management: Allows users to view, modify or cancel their bookings, enhancing flexibility and user control.

5. Notification system: Sends updates on bookings, cancellations and flight statues.

Non-functional Requirements:

1. Performance (Response Time): The system should process requests quickly to ensure a smooth user.

2. Scalability: The system must handle increased users and traffic without compromising performance.

3. Availability: Ensures the system is accessible 24/7 with minimal downtime, offering reliability.

4. Security (Data Protection): Protects sensitive user data with encryption.

5. Maintainability: Allows for easy updates and bug fixes, ensuring long-term system reliability.

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.

Answer:

The V-model is a sequential software development lifecycle model that emphasizes the synchronization of development and testing activities.

Key relationships:

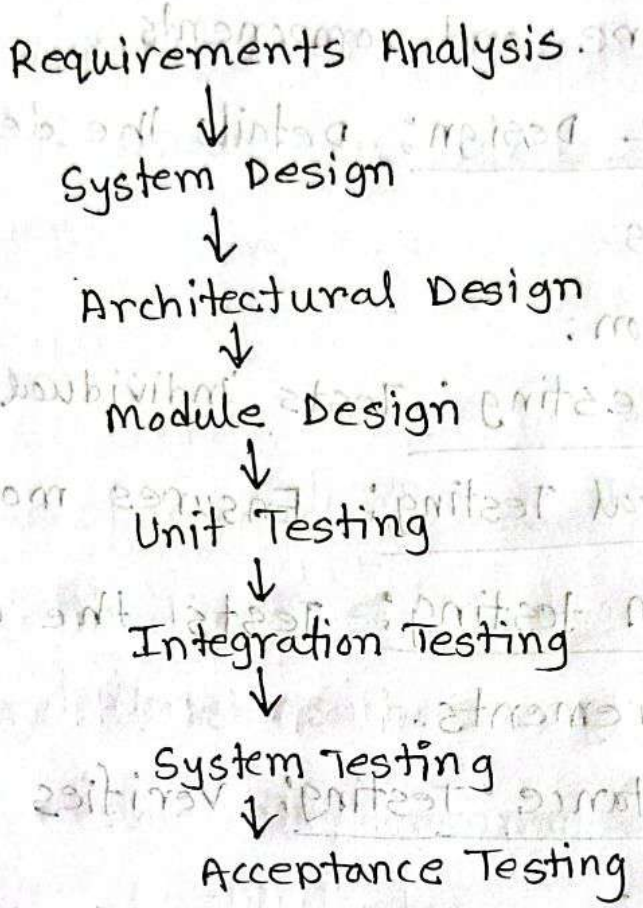
- Verification (Left side):
- Requirements Analysis: Defines user needs and system requirements.
- System Design: Creates high-level system architecture

- Architectural Design: Establishes the system's structure and components.
- Module Design: Details the design of individual modules.
- Validation:
- Unit Testing: Tests individual modules for correctness
- Integral Testing: Ensures modules work together
- System testing: Tests the entire system against requirements.
- Acceptance Testing: Verifies the system meets user needs.

How it works:

1. Development: starts with requirements analysis and progresses through design phases
2. Testing: Corresponding testing phases are planned and executed in parallel with development
3. verification: Ensures the product is built correctly.
4. validation: Ensures the product is built right

Visual Representation:



Benefits:

- Early Defect Detection: Testing starts early, reducing rework costs.
- Improved Quality:
- Better Communication
- Limitations:

* Less Flexibility
* Limited Customer Interaction

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 prototype model, particularly in terms of user feedback risk reduction and iterative development.

Answer:

Prototype development is a software development model where a basic version of the system, known as a prototype is created to gather feedback and refine requirements. It's an iterative process that allows developers.

key stages of prototype Development:

1. Requirements Gathering: Basic, high-level requirements are collected to built the prototype.

2. Prototype Development:

Build the prototype ensuring it demonstrate the primary features and provides a basis for user interaction.

3. User Evolution:

Share the prototype with stakeholders and end-users for feedback.

4. Iterative Refinement:

Based on feedback refine and improve the prototype.

How prototype Helps Refine software Requirement

i. Early validation of Requirements

ii) clear communication

iii) Discovery of Hidden Needs

Benefits of Using the Prototype Model:

1. User feedback.

2. Risk Reduction

3. Iterative Development

4. Improved Usability

5. Cost and Time.

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.

Answer:

The process improvement cycle in software engineering and focuses on refining and optimizing software development processes to improve quality efficiency and reliability. It at follows an iterative approach to identify inefficiencies, implement changes and measure the effectiveness of those changes.

key stages in the process Improvement cycle:

1. Process Assessment

2. Goal setting: Define clear, measurable objectives for process improvement. Align goals with organiz

3. Process Redesign: Propose and plan changes to existing processes,

4. Implementation: Apply the proposed process changes incrementally.

5. Monitoring and measurement: Collect and analyze process metrices

6. Review and optimization: Evaluate the success of implemented changes based on collected data.

7. Continuous Improvement: Treat process improve as an ongoing activity, repeating the cycle periodically to adapt to evolving challenges requirements.

commonly used process metrics in software Engineering:

Process metrics provide quantitative data to monitor and access software processes.

1. Productivity Metrics.

Example: Lines of code per developer per hour function points deliver per team.

2. Defect Metrics: Access the quality of the software by measuring defects.

3. Cycle time: Measures the time taken to complete a specific processs or task such as resolving a bug or delivering a feature.

4. Effort Variance: The difference between planned effort and actual effort expand.

5. Customer satisfaction metrics: measures user satisfaction with the delivered product on process.

6. Process compliance matrics: measure adherence to prescribed processes and standard.

How process metrics help in monitoring and Improving software processes.

1. Data-Driven Decision making
2. Early Problem Detection
3. Performance Benchmarking
4. Encouraging Accountability
5. Continuous feedback loop

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.

Answer:

The capability maturity model (CMM) developed by the software Engineering Institute (SEI) is a structured approach to evaluating and improving software development processes.

By categorizing process maturity into five levels it helps organizations move and ~~incogti~~ inconsitent practices to streamlined efficient and continously improving operations.

Each level of the model builds upon the previous one guiding organizations toward predictable outcomes, better quality, and enhanced efficiency.

The Five levels of CMM and their impact:

(i) Level 1: Initial (Unpredictable)

- Processes are ad hoc, dependent on individuals
- outcomes are inconsistent highlighting the need

② Level 2: Repeatable (& Basic control)

- Basic project management practices are established.
- Ensures repeatable success and reduces project risks.

③ Level 3: Defined (standardized)

- Process are documented are standardized across the organization.

④ Level 4: Managed (measured)

- Metrics are used to monitor and control process
- Enables predictable performance through data driven decisions.

⑤ Level 5: Optimizing (Continuous Improvement

- Focus on innovation, defect prevention and adaptability.
- Processes evolve to stay competitive and efficient.

How each level Drives process and performance Improvements:

1. Starting point: At level 1, organizations recognize the need for structured processes without formal practices, success depends on individual skills, making outcomes unreliable

2. Building Foundation: Level 2 introduces process discipline.

3. Scaling Efficiency: At level 3 standardize

4. Achieving Precision Level 4, focuses on measurement.

5. Sustaining Excellence: At level 5, innovation takes center stage.

Why CMM matters for organizations:

i. Predictability: Higher maturity levels make project outcomes more reliable and consistent.

ii) Quality Enhancement : Defect prevention and process control lead to superior software products.

iii) Operational Efficiency: standardaziation and optimized processes.

Question - 10:

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

Answer:

An agile software process is designed to handle the unpredictability inherent in most software projects. It recognizes that require

ments and customer priorities can change rapidly and it is difficulty to predic the necessary design before construction begins.

① customer collaboration over contact Negotiation
Engage customers throughout the devebpment process to ensure the product meets their needs.

② Working Software over comprehensive Document
Prioritize delivering functional software over extensive initial documentation.

③ Responsing to change over following a plan:
Embrace change even date in development to improvement the product.

④ Individuals and Interactions over processes and Tools: Faster collaboration, communication and teamwork as the Foundation of success.

(5) Frequent Delivery of value: Delivery working software in small, frequent increments.

(6) Sustainable Development: maintain a consistent work place to avoid burnout.

(7) Continous feedback and Improvement

(8) Technical Excellence and Good Design

(9) self organizing Teams

(10) simplicity: maximize

Benefits of Agile:

i) Increased customer satistaction

ii) Improved Quality

iii) Increased Flexibility

iv) Improved Team morale

v) Reduced Time-to-Market

Challenge of Agile:

i) Difficulty in estimating project timelines

ii) Resistance to change

iii) Maintaining focus

iv) Finding and retaining skilled personal.

Question-11:

Draw the release cycle of Extreme Programming (XP) and explain the influential programming practices:

Answer:

The extreme Programming (XP) release cycle Focus on frequent small releases ensuring that the software is always in a deployable state and customer feedback is continuously integrated.

Here's how the release cycle typically works:

Extreme programming (XP) release cycle:

i) Planning: Initial and iteration planning define features

ii) Development Iteration: Code is developed in short iterations (1-2 weeks)

iii) Continuous Testing: Automated tests ensure functionality and quality.

iv) Customer Feedback: customer reviews each iteration and provides input.

v) Release: A small working release is delivered after each iteration.

vi) Repeat: The process repeats in cycles refining the software with each iteration.

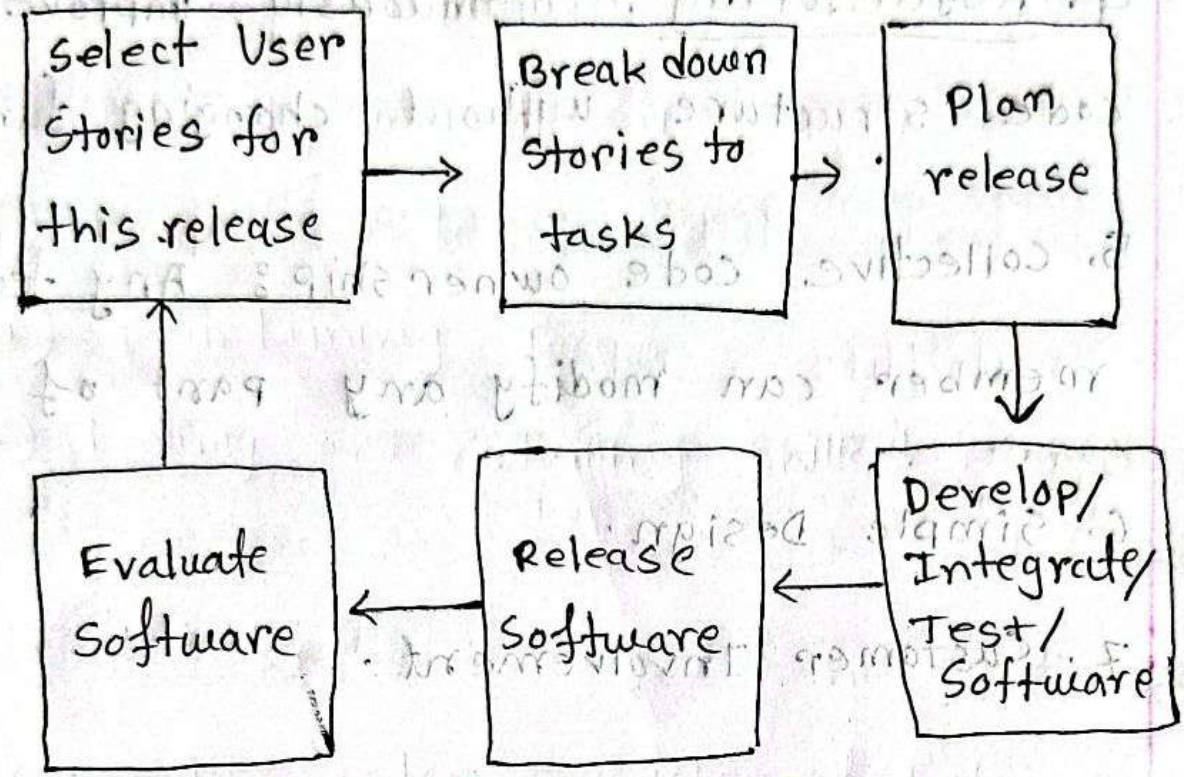


Diagram of release cycle of XP

Influential XP Programming Practices:

1. Pair programming : Two developers collaboration on the same code to improve quality
2. Test-Driven Development (TDD) : Write tests before code to ensure correctness.
3. Continuous Integration (CI) : Integrate code frequently with automated testing.

4. Refactoring: continuously improve the code structure without changing functionality.

5. Collective code ownership: Any team member can modify any part of the code.

6. Simple Design.

7. Customer Involvement.

Question -12:

A Local library wants to create a digital system to mangge its operations. The system will track books, members and borrowing activities. Each book has attributes like title authors, ISBN and genre. members have attributes such as name, membership ID and contact details. When a member borrows a book, the system records the

borrowing date, return due date and return status. The liabrary also want to maintain a catalogue of overdue books and their respective fines.

Using the scenario of a digital library management system, design an Entity-Relationship Diagram (ERD) to represent the entities (e.g. b and their relationships early explain the attributes of each entity and how they are interconnected.

Answer:

To design an Entity-Relationship's Diagram (ERD) for the digital library management system, we'll break down the entities their attributes on the requirements provided. Here's how it can be structured:

1. Entities and Attributes:

i) Book: Track books with attributes like Book-ID, Title, Author, ISBN, Genre and Availability-status

ii) Member: Records members with member-ID, Name, Contact-Details and Membership status.

iii) Borrowing Activity: Manages borrowing details, including Borrow-ID (PK) Borrowing Date Return Due Date Return-Status, and fine Account

iv) Overdue Book: Logs overdue books with overdue_ID (PK) Book-ID (FK), member-ID (FK) Fine amount, overdue days.

2. Relationships:

i) Books ⟷ Borrowing Activity: one book

can have many borrowing activities (1 to many)

ii) Members ↔ Borrowing Activity : One member can borrow multiple books (1-to-many)

iii) Borrowing Activity ↔ overdue Book: A borrowing activity may lead to one overdue record (1-to-1, optional)

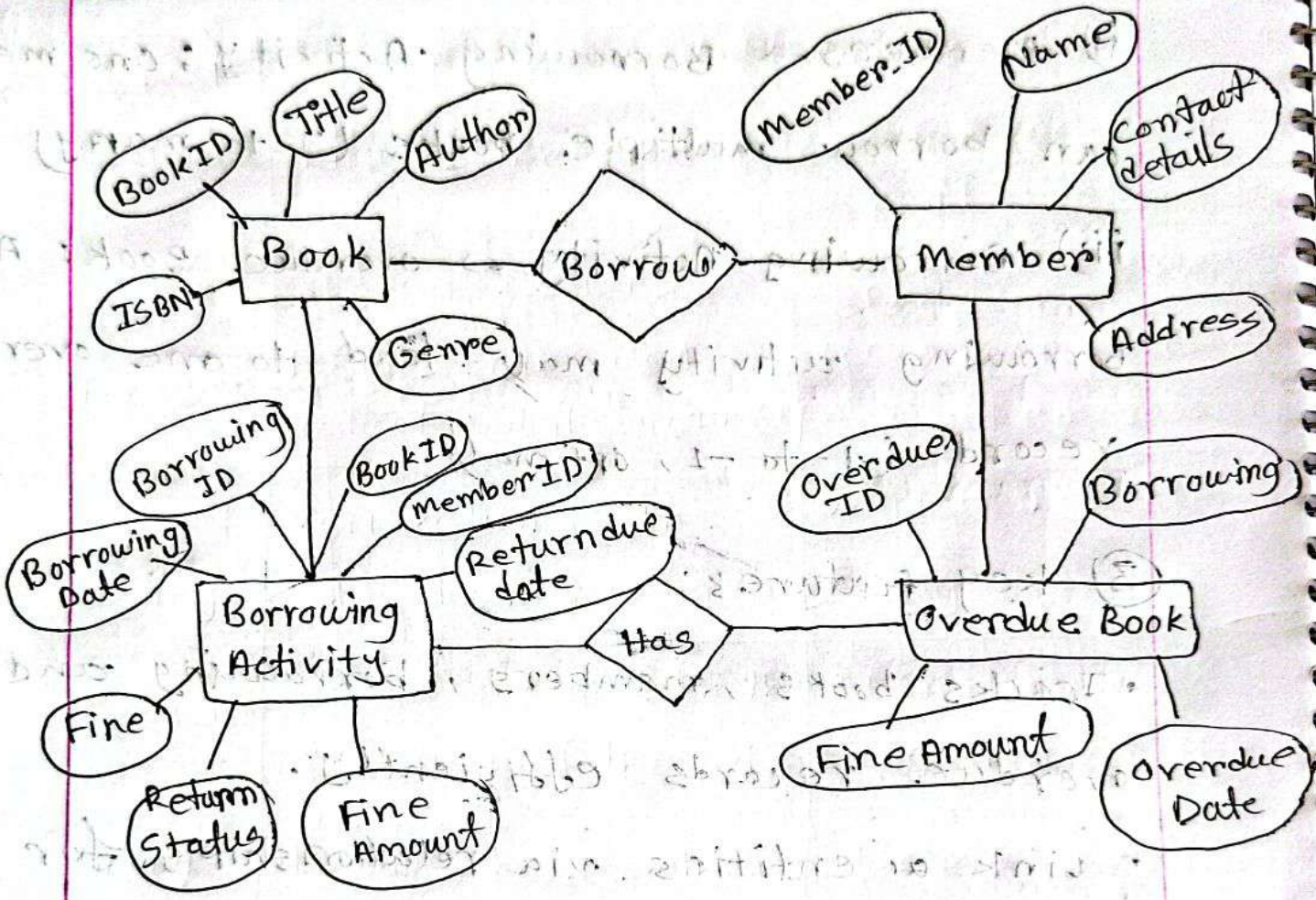
(3) key features:

- Tracks books, members, borrowing and overdue records efficiently.
- Link entities via relationships for better data organization and management.

Summary:

The ERD captures the relationships and key attributes needed to manage the digital library's operations IF effectively tracks.

ERD Diagram:



Question-13:

What is called Testing? Differentiate between Validation and verification.

Answer: Testing in software engineer

is the process of evaluating a software system or component to find defects or errors. It's critical step in the software development lifecycle to ensure the quality reliability and functionality of the final product.

Testing refers to the process of systematically evaluating a product or system to ensure it functions as intended and meets specified requirements while "validation" checks. If the product meets the actual needs of the user wheres "verification" confirms if the product is built according to the defined specifications essentially ensuring its "built right rather than right for the user.

Difference between validation and verification

Aspect	validation	verification
Definition	Ensures the product meets the users needs and expectations	Ensures the product is built according to the specified requirements
Focus	Focuses on the end product and its functionality	Focuses on the process and intermediate work
Type of Testing	Verification is the static testing	Validation is dynamic testing
Execution	It doesn't include the execution of the code	It includes the execution of the code
Timing	It comes before validation	It comes after verification
Performance	Verification finds about 50 to 60% of the defects	Validation finds about 20 to 30% of the defects

Q6

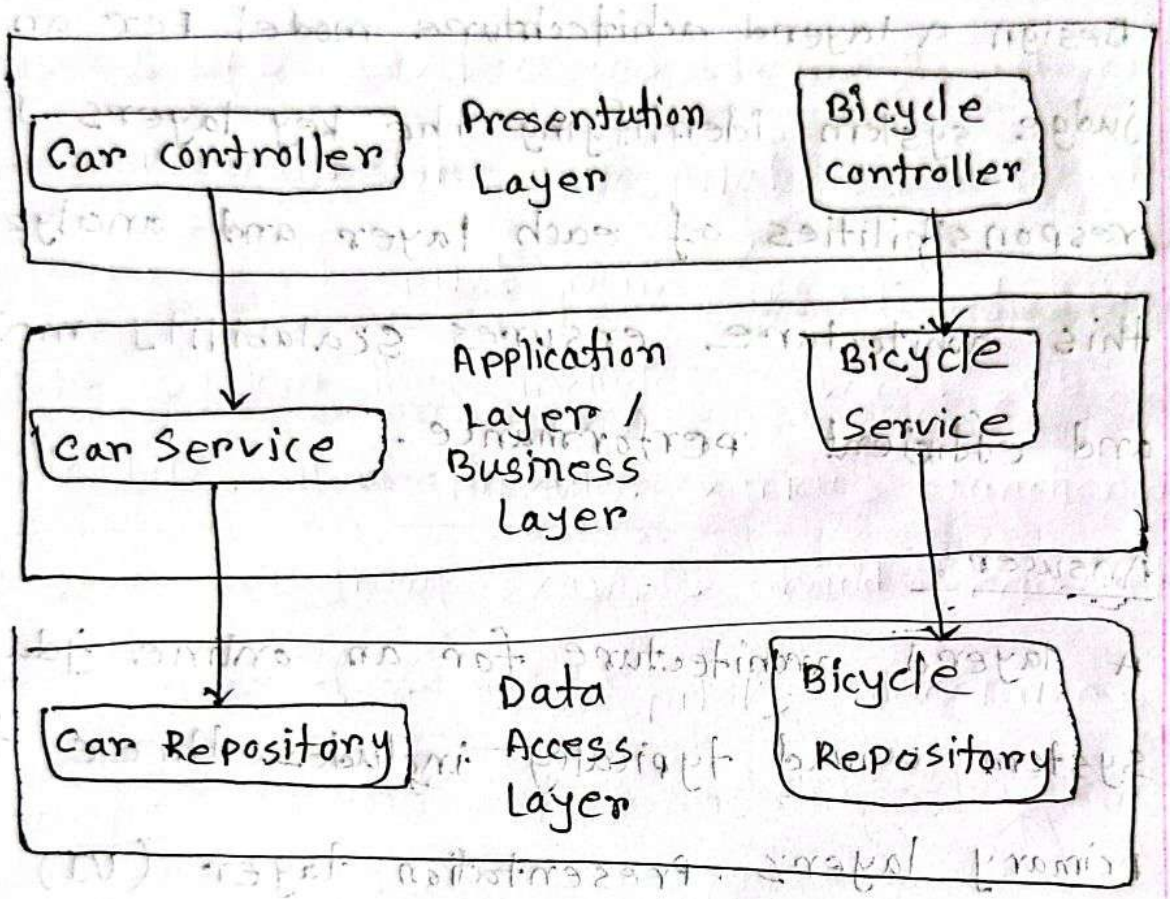
Question -14:

Design a layerd achitechture model For an online judge system identifying the key layers. Explain the responsibilities of each layer and analyze how this achitecture ensures scalability maintainability and efficient performanie.

Answer:

A layerd architecture for an online judge system would typically includes three three primary layers. Presentation layer (UI)' · Application layer (Buginees Logic) and Datd Access Layer (Database Interaction) with each layer having well-defined responsibilities.

Layered Architecture Diagram:



Layered Architecture for online Judger System:

1. Presentation Layer:
   i) Handles users interactions
   ii) Technologies : React, Angular - on mobile Interfaces

. Application Layer:

i) orchestrates requests between layer, manages archi tication and routes submissions.

ii) Technologies : Node Js . Django or Srping Boot.

3. Data Layer:

i) Manages storages for users, problems.

ii) Technologies : SQL and caching

Benefits :

i) scalability : Independent scaling of layers.

ii) Maintainability : Moduler design for easy updates

iii) Performance : Optimized request handling.

Conclusion :

This layered architecture organizes an online judge system int distinct independent layers Each layer handles specific responsibilities, enabling the system to progress user requests efficiently scale to accomodate growing traffic and be easily maintained or upgraded.

Question-15:

Draw a DFD (Level-0 and Level-1) and UML case Diagram for a Hospital Management System. A hospital management system is a large system that includes several subsystem or modules that provide various functions. Your UML case diagram example should show actors and use cases for a hospital's reception.

Answer:

Data Flow diagram Hospital management system is used to create an overview of Hospital management without going in too much detail.

Context Level (0 Level) DFD of Hospital Managem

System: The 0 Level DFD for Hospital Managemen system depicts the overview of whole hospital management system.

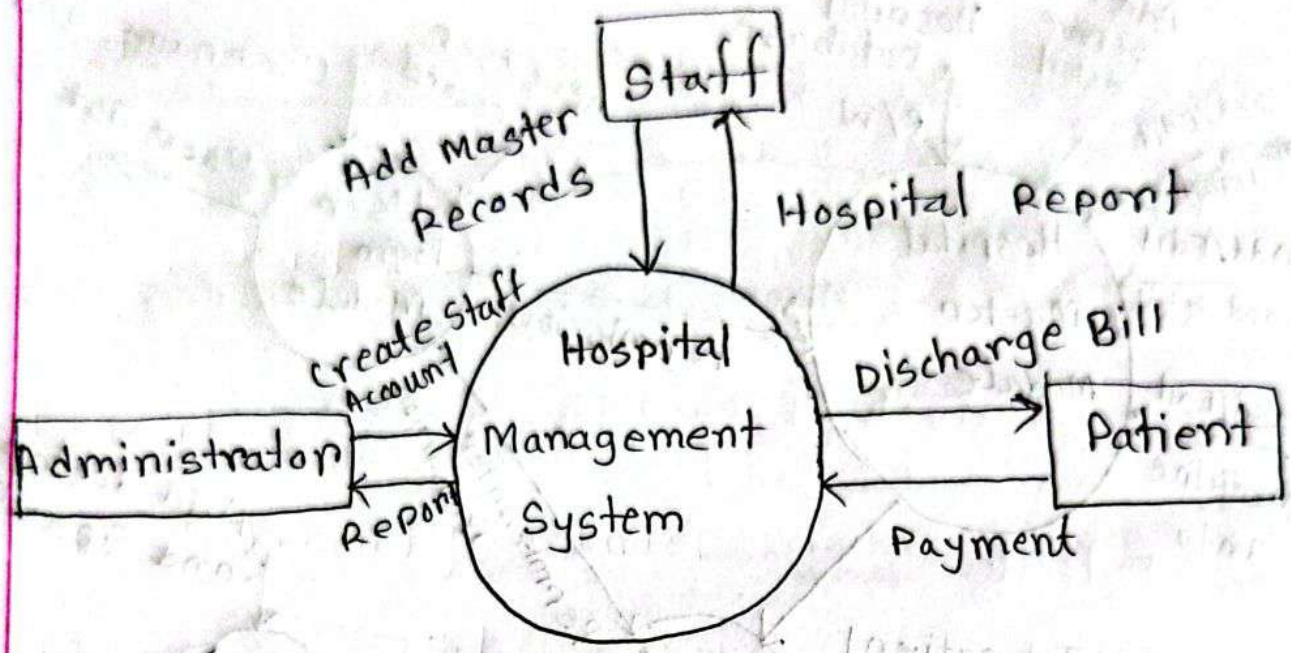
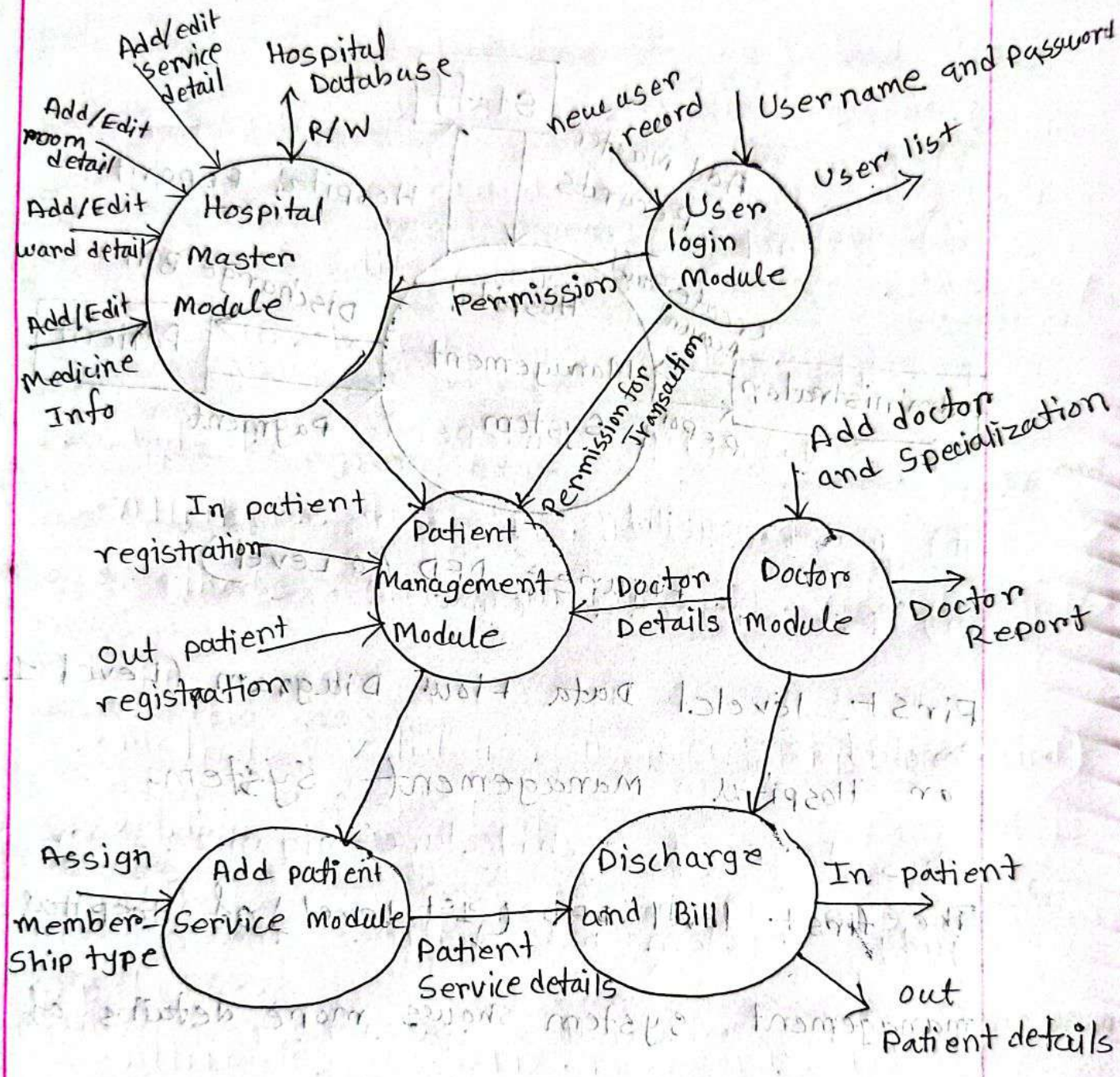


Figure: DFD (0 Level)

First levelcl Data Flow Diagram (Level 1 DFD) or Hospital Management System.

The first level DFD (1st level) of Hospital management system shows more details of processing Level 1 DFD list all the major sub processes that makes the entire system:

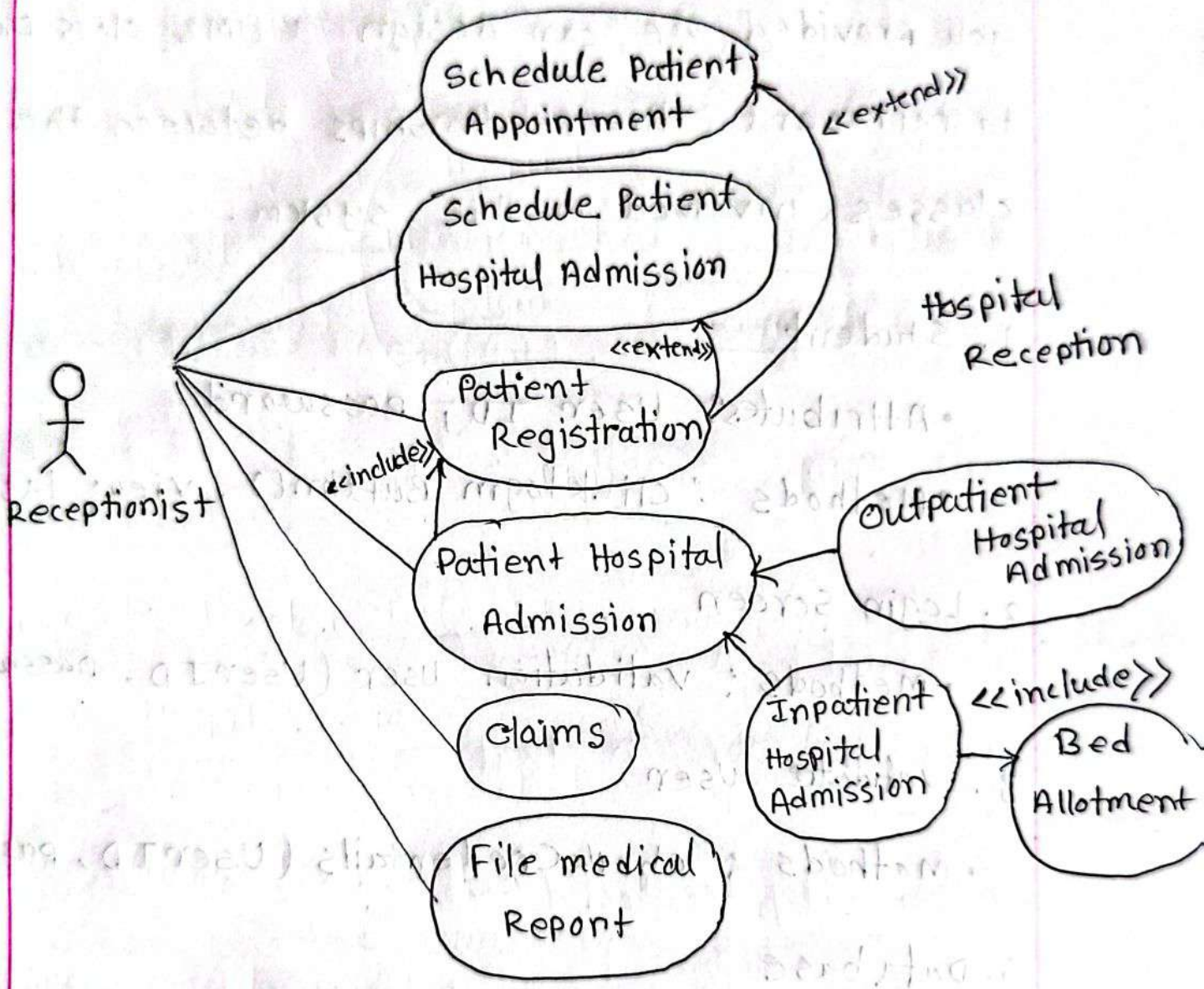


The important process to be carried out are:

1. User and login
2. Hospital master
3. Patient Registration
4. Doctor Module

5. Add Patient service 6. Discharge billing

UML use case Diagram:



In this diagram:

- Actor : Receptionist
- Use cases:
  - Schedule Appointment
  - Admit patient
  - Collect Patient Information
  - Allocate Bed
  - Recieve Payment
  - Generate Receipt.

Question-16:

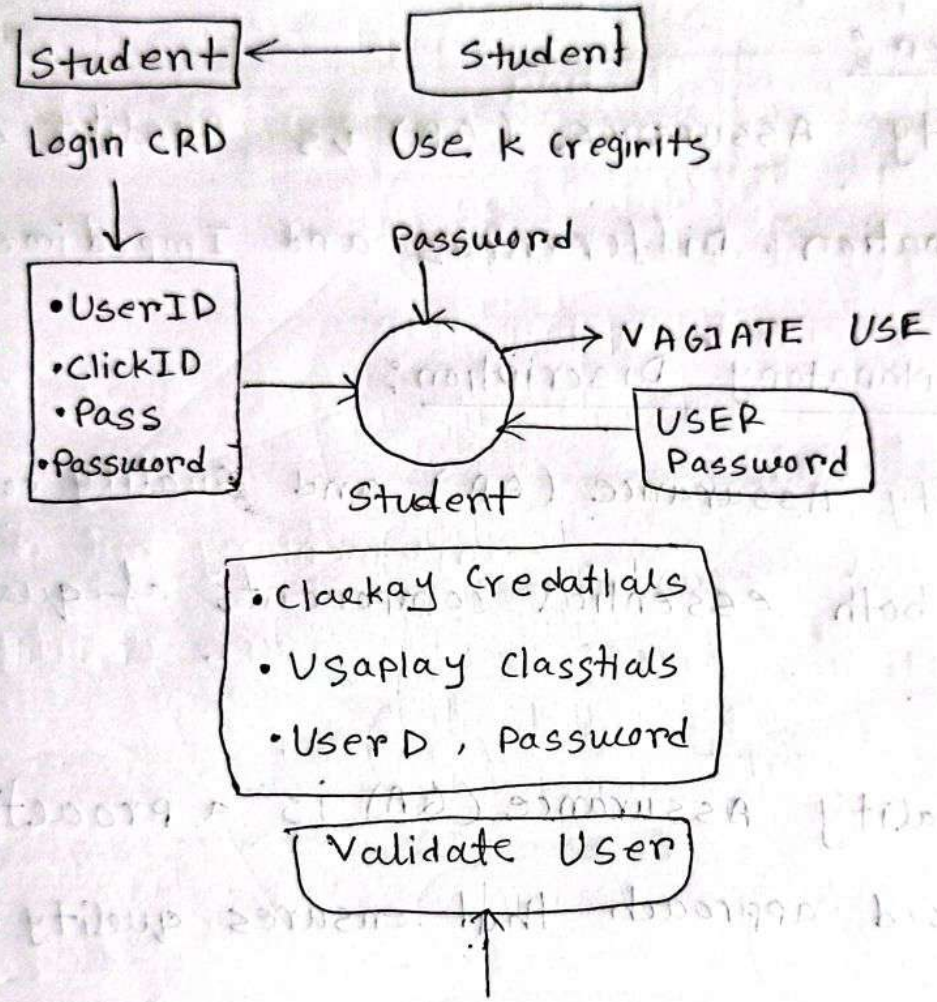
Answer: Based on the UML sequence Diagram you provided we can design a UML class Diagram to represent the relationships between the key classes involved in this system.

1. Student
   - Attributes: User ID, password
   - Methods: clickloginButton(), viewlassList()
2. Login Screen
   - Methods: validation User (UserID, password)
3. validate User
   - methods: checkCredentails (UserID, password)
4. Database
   - Attributes: UserID, password, classlist
   - Methods: FetchUserDetails(), return classlist()

Relationships:

- Student interacts with loginscreen (one to one)
- Login screen communicates with validate User

UML class diagram :



Here, is the UML class. Diagram for the Student login system based on the provided sequence diagram. It represents key classes.

Question -17 :

Answer :

Quality Assurance (QA) vs Quality control (QC) explanation, Differences and Impediment :

* Explanatory Discription:

Quality Assurance (QA) and Quality control (QC) are both essential components of quality management.

* Quality Assurance (QA) is a proactive, process focused approach that ensures quality is built

* Quality control (QC) is a reactive, product Focused approach that involves inspecting and testing the final product to identify and correct.

Difference Between QA and QC

Aspect	Quality Assurance	Quality control
Focus	Process - Oriented	Product - Oriented
objective	Prevent - Defects	Defect and correct defects
Approach	Proactive	Reactive
Responsibility	Everyone in the process	Dedicated Testing team
Timing	Applied Throughout development	Applied After Development

Impediments to QA and QC :

For QA :

i) Lack of standardized Process
ii) Poor management support
iii. Resistance to change
iv. Limited Resources
v. Time constraints

For QC:

1. Late Defect Detection
2. Inaquate Testing
3. Time Pressure
4. Dependance on QA
5. Limited Automation

Conclusion:

QA and QC are complementary but distinct QA establishes processes to ensure quality from the start while QC verifies.

Question - 18:

Role Answer:

Role of Quality Assurance (QA) is Software Development life cycle (SDLC):

The goal of Quality Assurance (QA) is not just to find bugs early but to ensure that

quality is built into the software development process from the beginning. T

QA as a discipline was introduced after world war II, where weapon testing was done to ensure reliability before actual use.

QA Role at Each Phase of SDLC

SDLC Phase	Role of Quality Assurance (QA)
1. Requirement Analysis	- Ensure clarity, completeness and Feasibility of requirements - conduct requirement reviews - Define acceptance criteria
2. Planning	- Identify risks and mitigation strategies. - Define QA objectives, scope and test - Plan resources, timelines and test environments

3. design	- Review system architecture and design for quality aspects - Identify potential Failure points early

In conclusion, QA is not just about Finding and Fixing bugs - it ensures the entire process is structured to produce a high quality product. By embedding QA at every phase of SDLC, organizations can reduce costs, enhance reliability and deliver better user experiences.

Question - 19:

Answer:

Rapid Application Development (RAD) model in Software Engineering:

Intro: The Rapid Application (RAD) model is an iterative and adaptive software development methodology that focuses on quick prototyping and fast feedback loops instead of extnsive planning and documentation. It was introdueed

Key phases of the RAD model:

i) Business Modeling
ii) Data Modeling
iii) Process Modeling
iv) Application Generation and Testing

Principle of the RAD model:

i) Prototyping over planning
ii) Active user Involvement
iii) Iterative and Incremental Development

Advantages of the RAD Model:

i) Faster Delivery
ii) Higher User satisfaction
iii) Flexibility
iv) Reduced Risk
v) Reusable Components

How the RAD Model Ensures Faster Delivery while maintaining Quality and user satisfaction:

i) Speed Through Prototyping
ii) Continuous User feedback
iii) Iterative Testing Ensures Quality
iv) Focus on Reusable components.

The RAD model accelerates software development without compromising quality

by emphasizing prototype, user feedback and reusability.

Question - 20 :

Answer :

White box testing ensures that all possible branches of a program are tested by covering decision points such as if, else and loops.

Test cases (Table) :

Decision statement	x Input	y Input	Expected output
If (y==0)	5	0	"y is zero"
else if (x==0)	0	3	"x is zero"
For loop doesn't run	0	2	"" (No output
For loop prints numbers divisible by y	4	2	"2", "4"
For loop prints numbers divisible by y	4	3	"3"
For loop with negative y	5	-2	"2", "4"
Negative x, loop doesn't execute	-3	2	"" (No execute)

JUnit Test class in Java

```
import static org.junit.jupiter.api.Assertions
import org.junit.jupiter.api.Test;
import java.util.Arraylist;
import java.util.List;
Class DecisionTest {
 private List <String> output = new Arraylist<
 private void println (String message) {
   output add (message);
}
private void process (int x, int y) {
      if (y == 0) {
        println ("y is zero");
      } else if (x == 0) {
        println ("x is zero"); }
      else {
        for (int i = 1, i <= x; i++) {
        If (i % y == 0) {
          println (String.valueof (i)); } } }
```

@Test
```
Void testYIsZero(){
    output.clear();
    process(5,0);
    assertEquals(List.of("y is zero"), output);
}
```

@Test
```
void testLoopDoesnotRun(){
    output.clear();
    process(0,2);
    assertTrue(output.isEmpty());
}
```

@Test
```
void testEdgeCasexNegative(){
    output.clear();
    process(-3,2);
    assertTrue(output.isEmpty());
}
}
```

Explanation of the JUnit Test class:

i. Simulating

ii. Implementing the original logic

iii. JUnit Test Methods

In conclusion, This JUnit test class effectively tests all possible branches in the given Java program. White box testing principles are applied to ensure complete code coverage.

Question-21:

Black Box Unit testing is earlier and more precise than Black Box system testing it can find errors very early even before the entire list first version is finished. Now, consider the production

codes that need Function testing. Suppose you have JUnit 4 API in your IDE and you are said to develop test codes for these production codes showing the application of the Exception, Setup Function and Timeout Rule. How do you solve it?

Answer:

JUnit 4 Black Box Unit Testing Approach

Black Box Unit Testing detect early by testing functions independently without knowing internal logic.

To test

Production code (Calculator.java)

```
public class calculation {
    private int memory;
```

```
public calculator () { this. memory = 0; }

public int divide (int a, int b) {
if (b==0) throw new ArithmeticExceptin(
" cannot divide by zero");
return a/b;
}
public double sqrt (double number) {
if (number<0) throw new IllegalArguement
Exception ("Negative number");
try { Thread. sleep (500); } catch (Interrupted
  Exception e) { }
}
public void storeIn Memory (int value) {
   this. memory = value;
}
public int get memory () {
   return this. memory;
  }
}
```

In JUnit 4, we can apply:

i. Exception Handling
ii. Setup Function
iii. Timeout Rule

JUnit 4 Test class For Function Testing:

```
import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.Timeout;

Public class calculatorTest {
  private calculator calculator;
  public Timeout globalTimeout = Timeout.
                                 millis(1000);
```

(a) Before

```
  public void setup() {
    calculator = new calculator();
    calculator.storeInMemory(10);
  }
```

```
@Test (expected = ArithmeticException.class)
public void divideByZero() {
    calculator.divide(5,0);
}
```

```
@Test
public void testsqrtComputationTime()
{
    double result = calculator.sqrt(16);
    assertEquals(4.0, result, 0.01);
}
```

```
@Test
public void testMemoryFunction() {
    assertEquals(10, calculator.getMemory());
}
```

- Explanation of the JUnit 4 test class:

i. Setup Function

ii. Exception Handling Test

iii. Timeout Rule

iv. Function Execution Test

v. Setup Verification

In conclusion,

This Junit 4 test class effectively applies exception testing setup functions and timeouts. The black box approach ensures Function correctness without accessing internal logic. These techniques help detect errors early before full system testing. This method ensures early error detection and efficient function validation.