Answer to the question no: 01

Product backlog for the E-Commerce Application.

For: User story: 01

1. Design Login UI - Create a wireframe and implement a front-end login page.
2. Set up Authentication System - Implement authentication using Django's or OAuth.
3. Encrypt User Credentials - Use hashing algorithms for password security.
4. Implement Session Management - Ensure users remain logged in ~~securi~~ serely.
5. Write Unit Tests - Test login functionality for security and correctness.

for user 02

1. Design Search UI - Create a user-friendly search bar and category filter.

2. Implement Backend search Logic - Use database queries or an indexing system for fast retrieval

3. Optimize Database for search - Index product names and categories for efficiency,

4. Integrate Search with frontend - Ensure AJAX or API-based communication between frontend and backend.

5. Write Unit Tests - Verify the search functionality ~~work~~ works correctly.

During the sprint planning meeting, the development team will prioritize these user stories based on:

1. Customer Value - The login functionality is critical as it provides access to user accounts. Therefore it should be prioritized first

2. Technical feasibility - If authentication has dependencies on external services (OAuth, database setup), the team ensure these are in place before implementation

3. Efforts Estimation - Tasks requiring more time might be split across multiple sprints.

Proposed sprint order:

Sprint 1: User authentication
Sprint 2: Product search functionality.

Tracking Progress with a Scrum Board.

A Scrum Board will be used to track Progress.

To Do	In Progress	Done.
Design Login UI	set up Authentication system.	Encrypt user credentials.
Implement Session Management		Write Unit Tests for Login.
Design Search UI	Implement Backend Search Logic	Optimize Database for Search
Integrate Search with Frontend		Write Unit tests for Search

Answer to the question no: 2

Comparison of spiral, Agile and Extreme Programming (XP) In Risk Management And Adaptability.

1. Spiral Model

Risk Management.

- The spiral model is explicitly designed for risk Management.
- Each phase involves risk analysis before proceeding to the next iteration.
- High risk components are addressed early through prototyping and feasibility studies.

Adaptibility:

* The model allows for iterative refinements, but changes may be costly if they emerge late in the cycle.
* works well for projects requiring extensive risk management.

* Spiral Model is best for high-risk projects where through risk evalusion is needed before development.

2. Agile Methodology.

Risk.

1. Agile Mtg. mitigates risk by delivering working software in short iterations.
2. Frequent client feedback ensures that risk are addressed early.

Adaptibility

* Agile is highly adaptive and supports incremental feature additions.
* The client is actively involved, which ensures the evolving product remains aligned with business needs.
* However, Agile might struggle with handling deep technical risk if not planned well.

Agile is Best for projects with uncertain or evolving requirements where client involvement is high.

3. Extreme programming (XP)

* Risk Management

- XP focusses on continuous feedback and rapid iterations, reducing the risk of developing unwanted features.

- Practices like test-driven development (TDD), pair programming and continuous integration minimize technical risks.

* Adaptability

* XP allows for frequent changes due to evolving client needs.

* However, it works best for small to medium teams and may not handle long, high-risk projects.

XP is best for projects with frequent requirement changes where technical excellence is a priority.

Best choice : Agile with Risk considerations.

Given the high risk and evolving requirements, Agile is the most suitable methology, but it should incorporate elements of risk management from the Spiral Model:

* Use short spints to iteratively refine features based on client feedback
* Prioritize riskier features first using a backlog refinement process.
* Develop prototypes or proof-of-concept models to address uncertainty.
* Combine Agile with risk analysis techniques.

Agile provides the best balance between risk management and adaptability while ensuring cost-effective and manageable software evolution. However, for particularly high-risk components, an Agile-Spiral hybrid approach may be beneficial.

Answer to the question no: 3

comparison Development Models: Waterfall, Agile, Extreme programming (xp), and Spiral

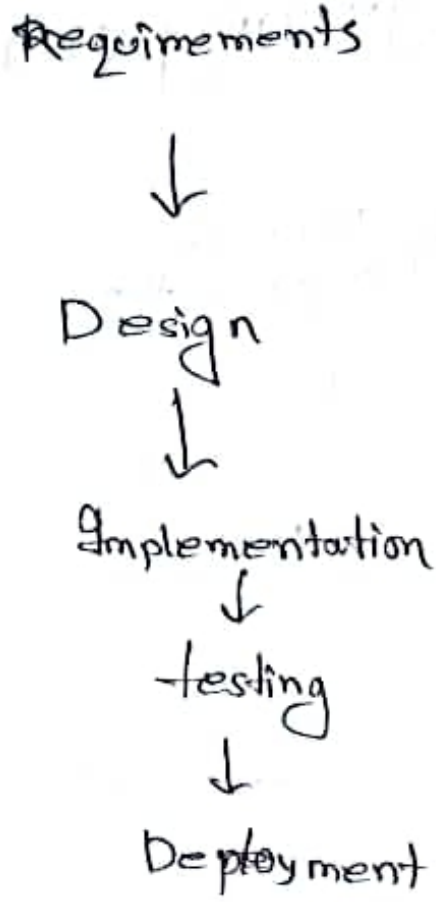
Methodology	Predictability	Customer collaboration	Risk management	flexibility
Waterfall	Hy High	Low	Low	Low
Agile	Medium	High	Medium	High
Extreme Programming (xp)	Low	Very High	Medium	Very High
Spiral	Medium-high	Medium	Very High	Medium-

Detailed Analysis

1. Waterfall Model

This model is best for projects with well-defined requirements and strict deadlines.

Procedure



* work well when requirements are stable and changes are minimal
* Ideal for fixed timelines and budgets.

Cons

1. Not flexible
2. Limited customer involvement
3. poor risk management.

2. Agile methodology

Best for projects with evolving requirements and continuous customer feedback.

Procedure

1. Highly flexible
2. High customer involvement
3. Better risk mitigation

Cons

1. Less predictability
2. Requires active customer participation.

Extrem

8. Extreme programming (XP)

Best for projects with rapidly changing requirements that demand technical excellence.

Pros

1. Continuous feedback, test-driven development (TDD), and pair programming
2. Very high adaptability
3. High code quality and ~~freq~~ frequent releases.

Cons

1. Requires highly disciplined developers and strong teamwork
2. Suited for small-teams.

4. Spiral Model

Best for projects with high risk and need for both structure.

Pros

1. Combines elements of waterfall and Agile.
2. Strong risk management.
3. Suitable for prototyping needed projects.

Cons

1. More complex and costly
2. Requires experienced teams.

so. for project A.

Best methodology: Waterfall

for Project B

Best methodology: Agile (Scrum/Kanban) or XP.

Answer to the question no: 4.

Principals of Software Engineering Ethics.

Software Engineering ethics is a set of moral ~~principal~~ principles that guide ~~softe~~ software professionals in their work, ensuring they act responsibly and with integrity.

1. Public Interest and safety
2. Integrity and Honesty
3. Confidentiality and Privacy
4. Fairness and Non-Discrimination
5. Competence and Professional Development.
6. Intellectual property and copyright
7. Social Responsibility.

Issues Related to Professional Responsibility

1. Security and Privacy violations.
2. Software Reliability and safety
3. Intellectual Property violations.
4. Workplace Ethics.
5. Conflict of Interest.

ACM/IEEE Code of Ethics in Software Engineering

1. Public Interest
2. the client and Employer Responsibility
3. Product Quality
4. Professional Integrity
5. Fairness & Equinity

Answer to the question no: 3

Functional Requirements.

1. Flight Booking and Ticket Management

2. Pay Payment Processing

3. User Authentication and Role Management

4. Flight Status and Notifications

5. Reservation Modification and Cancellation.

Non-functional Requirements.

1. Performance and scalability

2. Secuil Seculnity and Data Protection

3. Usability and Accessibility

4. System Availability and Reliability

5. ~~Syst~~ Maintainability and Extensibility

Conclusion

Both functional and non-functional requirements are essential for the Airport Reservation System. By addressing both aspects, the system becomes efficient, scalable and user-friendly, ultimately improving passenger satisfaction and operational ~~efficient~~ efficiency for Airlines.

Answer to the question no: 06

V-model of testing in a plan-Driven software Process.

The v-model (validation and Verification Model) is an extension of the waterfall model, where testing activities are directly linked to each stage of development.

Phases of the v-model.

1. Development phases (Left side of v-model)

    1. Requirement Analysis

    2. System Design

    3. Architecture Design

    4. Module Design

    5. Implementation.

2. Testing phases (Right side of v-model)

    1. Unit testing

    2. Integration Testing

    3. System Testing

    4. Acceptance testing

Key benefits of v-model.

1. Early Defect Detection.

2. Clear Test planning

3. Better Documentation

4. Reduces cost of Fixing Bugs.

Conclusion:

The v-model ensures structured and systematic software development by linking each development phase with a corresponding testing phase.

Answer to the question number 07.

Ans Prototype development in Software Engineering

* Prototype development is an iterative approach where a preliminary version of the software is build to refine requirements before full scale development.

Key stages of Prototyping:

1. Requirement Gathering
2. Quick Design
3. Prototype Development
4. User Evaluation
5. Refinement
6. Finalization

Benefits of Prototyping:

- User Feedback
- Risk Reduction
- Iterative Development
- Improved Requirement clarity.

CHALENGES IN PROTOTYPING

- Scope Creep
- Highen Initial cost
- User Misconceptions.

Best Practices for Effective Prototyping:

- Define Clear Objectives
- Engage Stakeholders Early
- Use Low-fidelity models First
- Iterate Based on Feedback.

Answer to the question no: 08

Process Improvement Cycle in software Engineering

The process improvement cycle aims to enhance software development efficiency, quality and predictability. It follows a structured approach, often aligned with frameworks like CMMI.

Key stages of the process Improvement Cycle:

1. Process Assessment - Identify existing weaknesses and inefficiencies in software process.
2. Process Definition - Establish new or modified processes based on best practices.
3. Implementation
4. Monitoring and Measurement
5. Evaluation and Refinement.

Commonly used Process Metrics

1. Productivity Metrics
   - Lines of code (LOC)
   - Function Points (FP)
2. Quality Metrics
   - Defect Density
   - Mean Time to failure
3. Time & Efficiency Metrics
4. Customers satisfaction Metrics

Answer to the question no: 10

Core Principles of Agile Software Development

1. Individuals and interactions over processes and tools

2. Working software over comprehensive documentation

3. Customer collaboration over contract re-negotiation

4. Responding to change over following a plan.

Application of Agile in Different Software Development Environments.

1. Setup & Small Teams

2. Enterprise Software Development

3. Embedded & Regulated Industries

4. Game Development

Answers to the question no: 9.

Software Engineering Institute Capability Maturity Model (SEI CMM)

The capability Maturity Model (CMM), developed by the Software Engineering Institute (SEI) is a framework for assessing and improving software development processes.

Five level of Capability and Maturity

1. Initial (Level 1) - Chaotic & Ad Hoc
2. Repeatable (Level-2) - Basic project Management
3. Defined (Level-3) - Standardized Processes
4. Managed (Level-4) - Measured & Controlled
5. Optimizing (Level-5) - Continuous Improvement

Benefits of Agile Methods

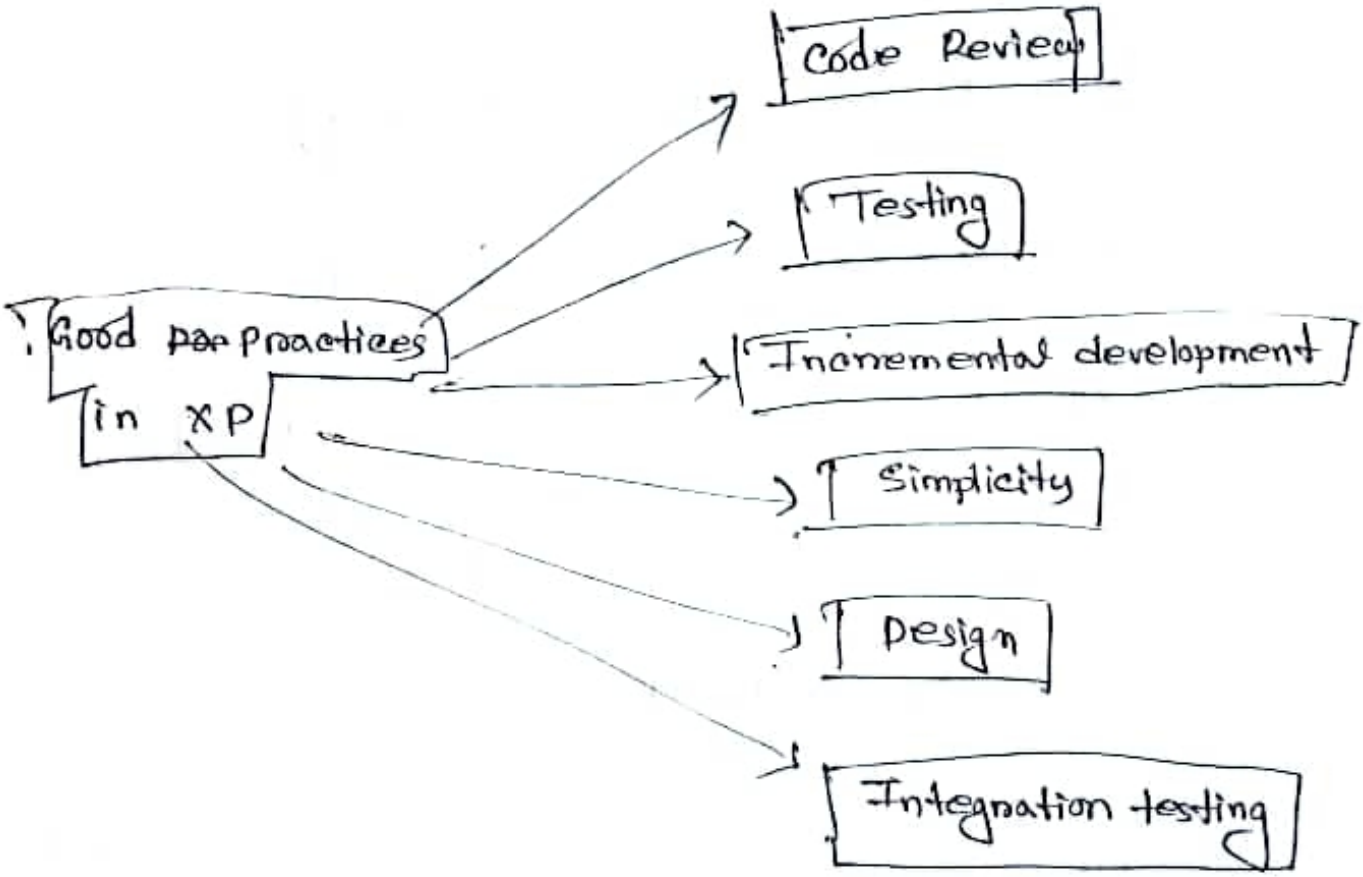
- ☑ Faster time to market
- ☑ Higher customer satisfaction
- ☑ Flexibility.
- ☑ Better risk management
- ☑ Higher team productivity.

Challenges:

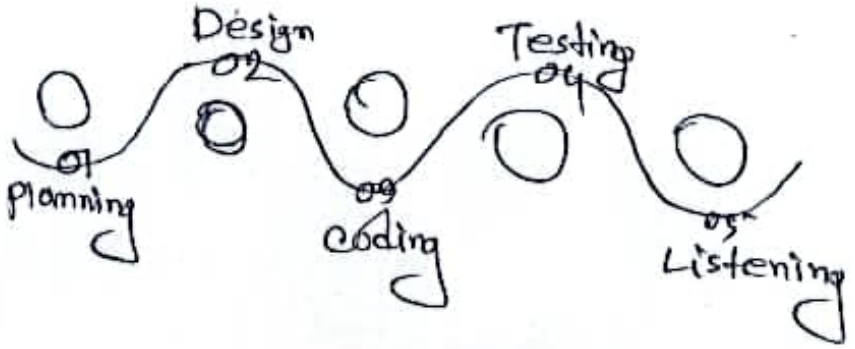
1. Difficult to scale
2. Requires active customer involvement
3. Less Predictability
4. High dependency on team collaboration.

Answer to the question no:

Good practice in Extreme Programming.



Release cycle of XP



Xp release cycle · phases

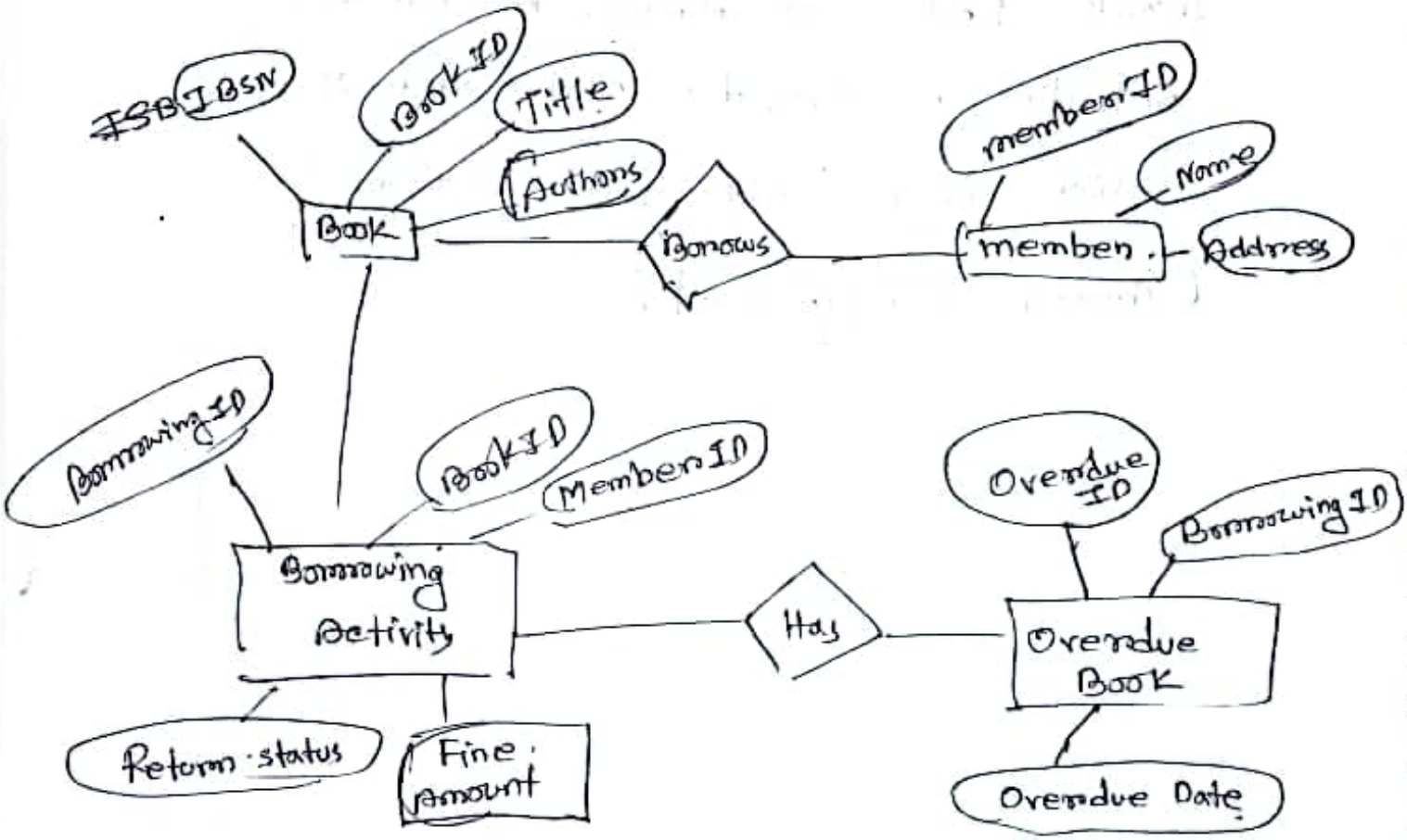
1. Exploration
2. Planning (Release planning)
3. Iteration (coding & Testing)
4. Release (Deployment & Feedback)
5. Maintenence & continuous Improvement.

ardent

Answer to the question no: 12

Entity Relationship Diagram (ERD)

For Digital library management.



The ERD captures the relationships and key attributes needed to manage the digital library's operations. It effectively tracks books, members, borrowing activities. Overdue status and fines which is essential for efficient library management.

Answer to the question no: 13

Testing

Testing in software engineering is the process of evaluating or software application to ensure it functions correctly, meets specified requirements and is free from defects. It involves executing a programme on system with the intent of finding errors; verifying functionality and ensuring reliability.

Key Difference between verification and validation.

Verification = Are we building the Product right?

Validation = Are we building the right Product?

Difference between validation and verification.

Aspect	verification	validation
Definition	Ensure the software meets design specifications.	Ensures the software meets user needs and requirements.
purpose	To check whether the product is built correctly	To check whether the right product is built
· Focus	Process - oriented	Product oriented
Techniques used	Reviews, inspections, walkthroughts.	Testing, user Acceptance Testing (UAT).

Answer to the question no: 14

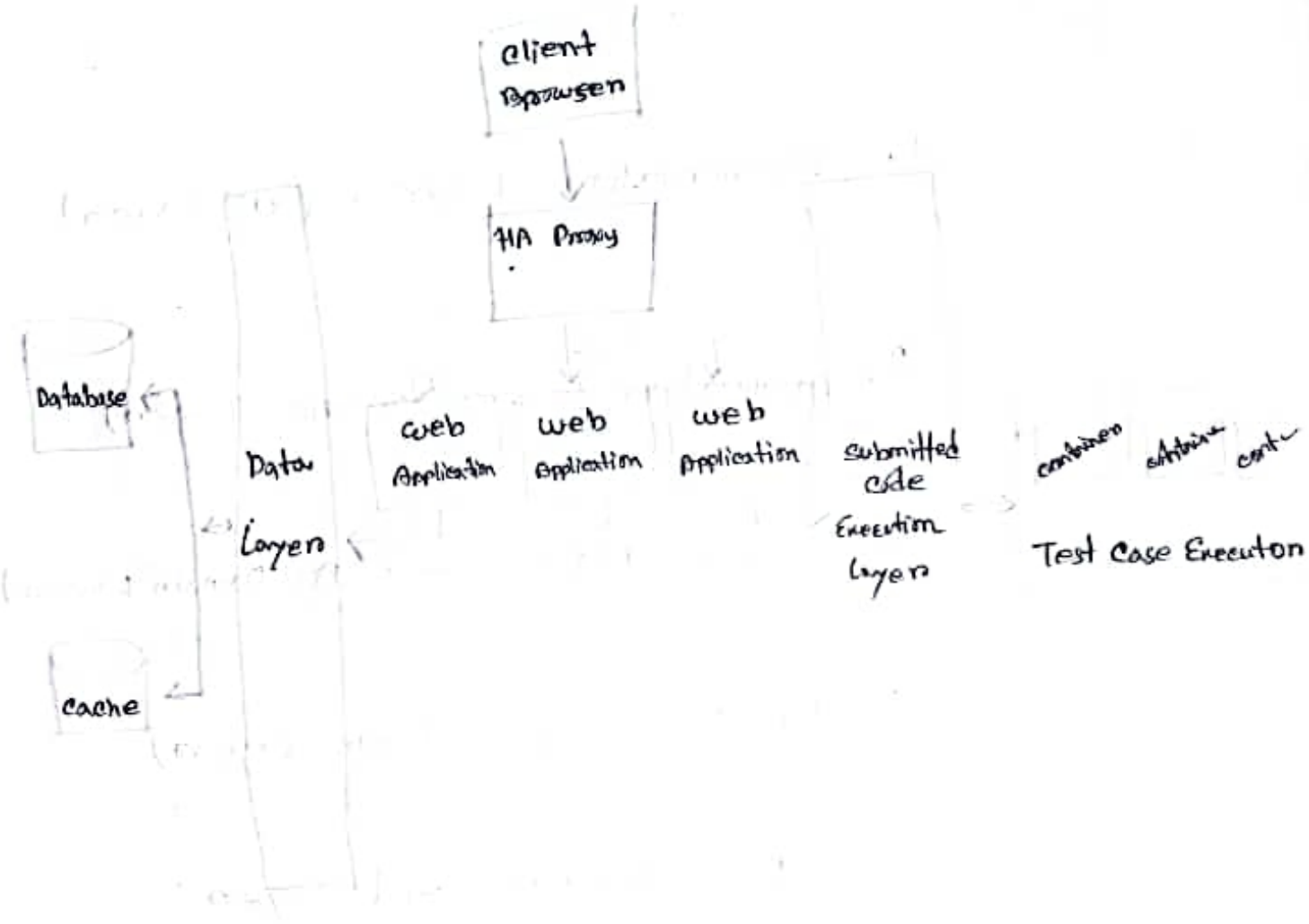


Figure: Layered Aahi Architecture Model for an online Judge system.

The layered architecture for an online Judge System typically follows a multi-layered approach that dev divides the system into distinct layers.

Layers

1. Presentation Layer (UI Layer)

2. Application Layer (Service Layer)

3. Business Logic Layer (Domain Layer)

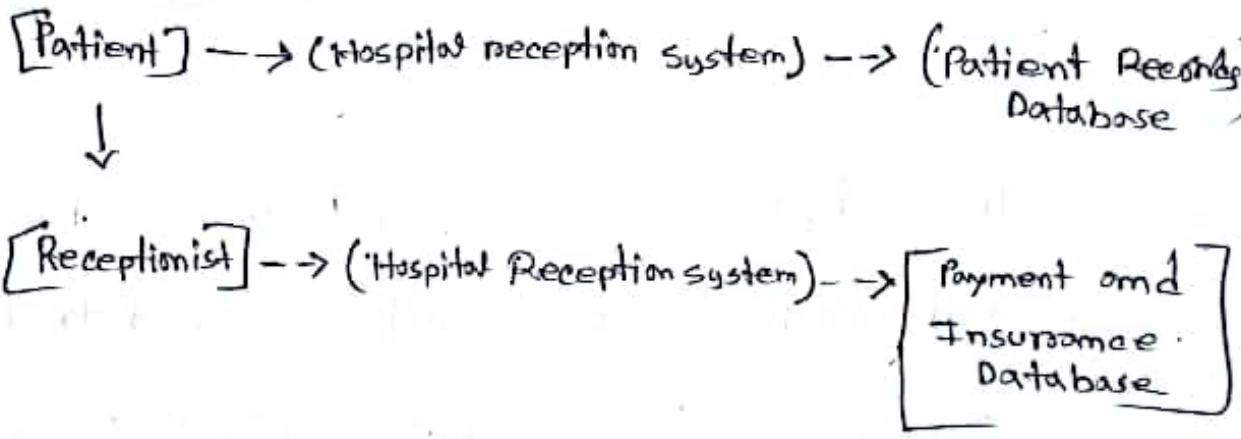
4. Data Layer (persistence Layer)

5. Execution / Backend Layer.

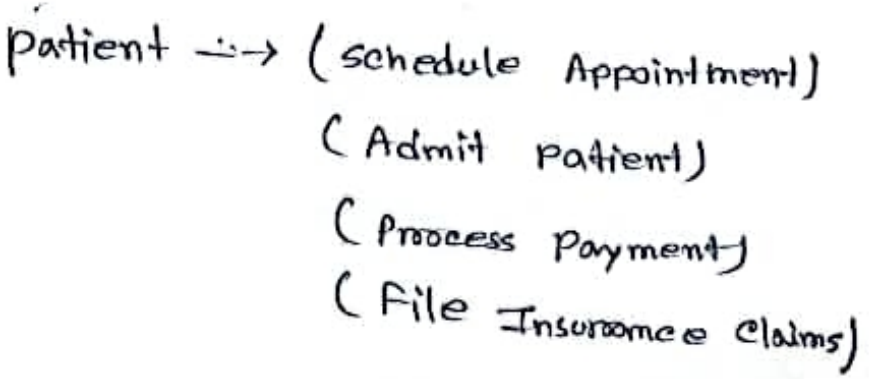
Answer to the question no: 15

To design DFD (Data Flow Diagram) and UML Use case Diagram for the Hospital management System.

- Level - 0 - DFD (~~cont~~ context Diagram)



Level - 1 - DFD



[Receptionist] --> (collect Patient Information)

(Schedule Appointment)

(Admit Patient)

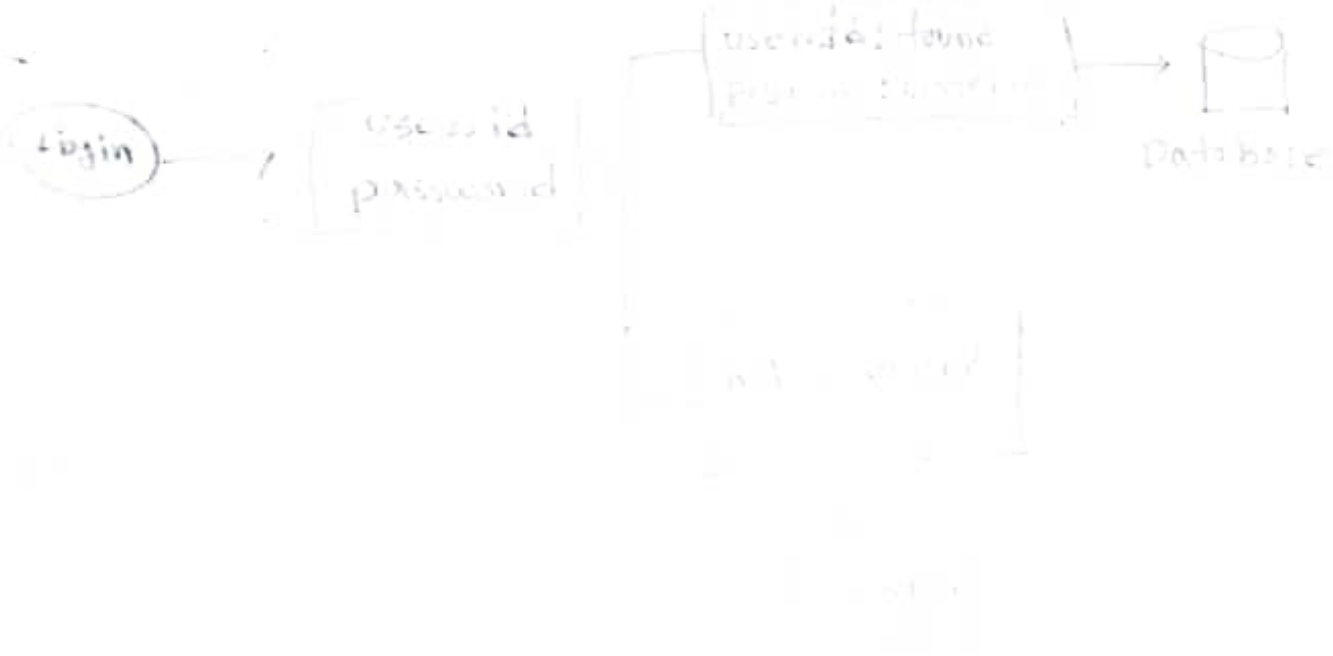
(Process Payment)

(File Insurance Claims)

(Issue Receipts/Reports)

The division of responsibilities between receptionist and patient, along with clean data flows, ensures smooth operations with minimal delay in processing requests.

Answer to the question no : a 16



UML = Unified Modeling Languag.

A UML class diagram is a ~~va~~ visual tool that represents the structure of or system by showing its classes attributes, methods and the relationships between them.

Answer to the question no: 17

Quality Assurance (QA) and Quality control are both critical elements of the overall quality management system in software development, but they differ significantly in terms of their focus, methods and objectives.

Differences Between QA and QC

Aspect	Quality Assurance (QA)	Quality Control (QC)
Focus	Process-oriented	product oriented
Objective	To prevent defects by improving processes	To identify and fix defects in the product.
Nature	Proactive	Reactive.

Aspect	Quality Assurance (QA)	Quality control (QA)
Focus	Process-oriented	product-oriented
Scope	Broaden, involving all aspects of the process	Narrower, focusing specifically on testing and product validation.
Activities	Process definition, process improvement, auditing, reviews.	Testing, inspection, defect detection, verification
Tools	Process management tools, standards, reviews.	Testing tools, defect tracking tools, checklists.
Timing	Done Throughout the project & lifecycle	Done at the end of the development phase before release.

Answers to the question no:

Quality Assurance (QA) in software Development Life cycle(SDLC)

Role of QA in Each phase of SDLC:

1. Requirement Gathering and Analysis

- Objective : Ensure that the requirements
- QA Role

Requirement Reviews

validation of Requirements

Risk Analysis

2. System Design

3. Development (coding)

4. Testing (verification & validation)

5. Deployment

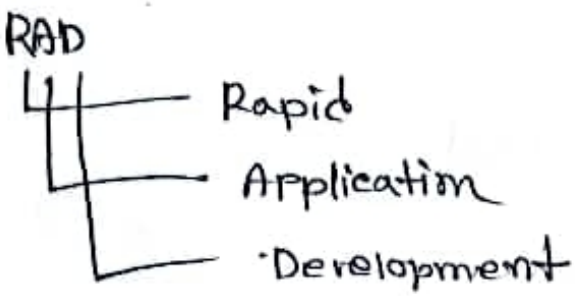
6. Maintenance

Key Goals of QA in SDLC:

1. Preventing - Defects
2. Continuous Improvement
3. Customer satisfaction

The role of QA at each phase of the Software Development Life cycle (SDLC) is to improve processes, reduce risks, prevent defects and ensure the final product meets both user and business requirements.

Answer to the question no: 19



The Rapid Application Development (RAD) model is an incremental software development process that focuses on rapid prototyping and fast delivery of software solutions.

Key phases of the RAD Model:

1. Requirements planning phase
2. User Design phase
3. Construction phase
4. Cutover phase

Key principle of RAD

1. User Involvement

2. Iterative Development

3. Time - Boxing

4. Rot Prototyping

5. Parallel Development

Advantages

1. Faster Development
2. Flexibility and Adaptability
3. Increased User Involvement
4. Improved Quality
5. Reduced Risk.

Answer to the question no: 20

To implement the described functionality in Java, we will use Junit for unit testing and simulate the decision logic in the method.

Decision Table:

Decision	X input	Y input	Expected Output
y == 0	Any	0	"y is zero"
x == 0	0	Any	"x is zero"
i % y == 0 (loop Iteration)	1, 2, 3, ...	Non-Zero	Numbers divisible by y

Answer to the question no: 21.

Exception Testing:

- The @Test(expected = Exception.class) annotation is used to test that an exception is thrown when expected.

- you can also use the try-catch block inside the test method to check for specific exceptions if needed.

Setup - function:

- The @Before annotation defines a setup method, which is run before each test. This is useful for initializing shared resource, test data or mocks that are used in multiple tests.

Timeout Rule

- The @Test (timout = <time_in_millisecond>) annotation is used to specify that a test should fail if it takes longer ~~that~~ than the specified timeout period.

- Alternative, the (Timeout) rule can be used to apply the timeout in a more flexible manner.