Ans. to the Q.No→1

(1)

Product backlog for the first story is mentioned below:

User Story 1: Login Functionality
Task 1: Design Login Page UI
Task 2: Implement Backend Authentication
Task 3: Frontend Development
Task 4: Password Encryption
Task 5: Login API Integration
Task 6: Session Management
Task 7: Security Measures
Task 8: Testing

Product backlog for the second story is mentioned below:

~~Task 1:~~
User story 2: Search Functionality
Task 1: Design Search UI
Task 2: Setup Product Database
Task 3: Implement search Algorithm
Task 4: Frontend Development

Task 5: Search API Integration
Task 6: Filter Functionality
Task 7: Pagination
Task 8: Testing

(11)

The development team can prioritize these user stories during a sprint planning meeting, considering value to the customer and technical feasibility by -

Evaluation:

i. Value to customer:
- Secure Login: Critical for user trust and access.
- Search by category: Enhances user experience and helps find products easily.

ii. Technical Feasibility:
- Secure Login: Moderate to high complexity involving security and authentication.
- Search by category: Moderate complexity, involving search algorithm and frontend integration.

Prioritization Process:

i. Discuss Customer value: Product Owner explains the importance of each story.

ii. Estimate Effort: Team estimates the effort required for each story.

iii. Balance value and Feasibility: Prioritize stories with high value and feasible effort

Prioritization Outcome:

- Sprint Goal: Enable secure user login and improve product search.
- Sprint Backlog: Focus on secure login first, followed by search functionality.

(III)

To track the tasks effectively, the development team can use a Scrum board with columns labeled "To Do," "In Progress," and "Done." Here's how it can be organized:

To Do:

- Design Login Page UI
- Implement Backend Authentication

- Frontend Development
- Password Encryption
- Login API Integration
- Session Management
- Security Measures
- Testing
- Design search UI
- Setup Product Database
- Implement Search Algorithm
- Frontend Development
- Search API Integration
- Filter Functionality
- Pagination
- Testing

In Progress:

- Tasks currently being worked on by the team will be moved here.

Done:

- Completed tasks will be moved here.

This Scrum board helps the team visually track the progress of each task, ensuring transparency and efficiency throughout the sprint.

Ans. to the Q.No→2

Spiral Methodology:

- Risk Management: Focuses on continuous risk assessment and mitigation.
- Adaptability: Iterative process with frequent stakeholder feedback.

Agile Methodology:

- Risk Management: Early and frequent delivery reduces risks.
- Adaptability: Highly adaptable to changing requirements through iterative cycles and customer collaboration.

Extreme Programming (XP):

- Risk Management: Uses practices like Test-Driven Development and Pair Programming to minimize risks.
- Adaptability: Quick response to changes with frequent releases and customer involvement.

So, it is understable that the most suitable methodology is Spiral Model. Because it is best for high-risk projects with evolving requirements due to its strong focus on risk management and adaptability through

iterative cycles. Prototyping helps validate high-risk components early on.

Ans. to the Q. No → 3

Waterfall:

i. Linear, sequential process.
ii. Good for stable, well-defined projects with strict deadlines.
iii. Predictable but inflexible to changes.

Agile:

i. Iterative, flexible approach.
ii. Great for projects with evolving requirements and continuous feedback.
iii. Emphasizes customer collaboration and frequent delivery.

Extreme Programming (XP):

i. Agile-focused on technical excellence and frequent releases.

ii. Ideal for changing requirements but requires skilled developers.

iii. Intense workload and pace.

Spiral:

i. Combines iterative development with risk management.

ii. Suitable for complex, high-risk projects.

iii. Expensive and time-consuming.

Project A has well-defined requirements and a strict deadline. So, the best methodology for it is waterfall because of its predictability and well structure.

Project B has evolving requirements with an uncertain timeline and continuous customer feedback. It suits Agile because of its flexibility and collaboration. Again Extreme Programming (XP) suits it too if the team is experienced and can handle the intense pace.

Ans. to the Q.No → 4

Software engineering ethics focuses on ensuring public safety, reliability, and fairness in software systems. Key principles include:

1. Public Interest: Prioritize the welfare and safety of the public.
2. Quality: Maintain high standards of work and avoid errors.
3. Integrity: Be honest and transparent with stakeholders.
4. Privacy: Protect users' data and ensure confidentiality.

The ACM/IEE Code of Ethics provides guidelines to help engineers make ethical decisions. Key aspects include:

i. Acting in the public good and prioritizing safety.
ii. Being honest about software risks and limitations.

iii. Maintaining competence and continuously improving skills.

iv. Promoting fairness and respecting privacy.

This code helps engineers resolve ethical dilemmas, report risks, and build trustworthy software that positively impacts society.

Ans. to the Q.No→ 5

Functional Requirements:

1. User Authentication: Secure log-in and role-based access to protect sensitive data.
2. Flight Search & Booking: Easy search and booking for a seamless user experience.
3. Payment Processing: Secure transactions via multiple payment method for reliability.
4. Reservation Management: Allow users to view, modify, or cancel bookings for flexibility.
5. Notifications: Send timely updates (email/sms) for bookings and flight changes.

Non-Functional Requirements:

1. Performance: Fast response times (<2 seconds) for smooth interactions.

2. Scalability: Handle increasing users and transactions without performance loss.

3. Availability: 99.9% uptime for reliable access.

4. Security: Encrypt data and comply with regulations (e.g., GDPR) for user trust.

5. Maintainability: Modular design for easy updates and long-term sustainability.

These requirements ensure the system is user-friendly, secure, and scalable, delivering a reliable and efficient experience for both users and admins.

Ans. to the Q. No→6

The V-Model illustrates the relationship between development activities and corresponding testing activities in a structured, plan-driven process.

Development Activities:

1. Requirement Analysis: Define user needs
   - Corresponds to Acceptance Testing
2. System Design: Create high-level system architecture
   - Corresponds to System Testing
3. Architecture Design: Break system into modules.
   - Corresponds to System Testing
4. Module Design: Design individual components.
   - Corresponds to unit Testing.

Testing Activities:

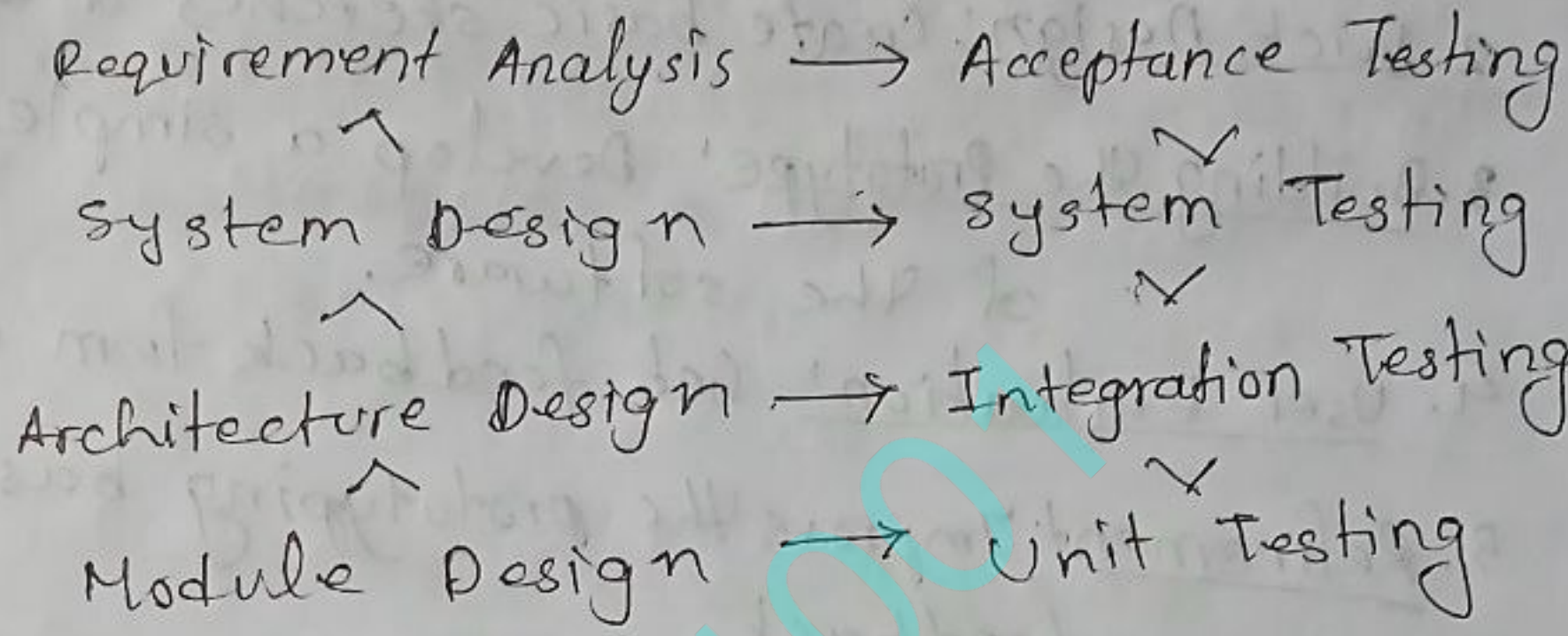
1. Unit Testing: Verify individual components
2. Integration Testing: Test interactions between modules.
3. System Testing: Validate the complete system against design.
4. Acceptance Testing: Ensure the system meets user requirements.

Relationships:

- Each development phase on the left has a direct correspondance to a testing phase on

the right.

- Testing activities validate and verify the outputs of development activities, ensuring correctness and alignment with requirements.



The V-model ensures a systematic and disciplined approach, where each development activity is validated by a corresponding testing activity, ensuring high-quality, reliable software.

Ans. to the Q. No → 7

Prototyping is creating an early version of software to refine ideas and gathero

feedback before building the final product.

key stages of Prototype development are -

1. Requirements Gathering: Understand user needs.
2. Quick Design: Create basic sketches or wireframes.
3. Building the Prototype: Develop a simple version of the software.
4. User Evaluation: Get feedback from users.
5. Refinement: Improve the prototyping based on feedback.
6. Iteration: Repeat the process until the prototype meets user needs.
7. Implementation: Develop the final product.

How Prototyping ~~makes~~ helps Refine Requirements:

1. Clarifies Needs: Makes requirements clear.
2. Early Issue Detection: Find problems early.
3. Better communication: Helps everyone understand and discuss the project.

Benefits of Prototyping:

i. User Feedback: User shape the product, improving satisfaction.

ii. Risk Reduction: Identifies problems early.

iii. Iterative Development: Continuous improvement leads to a better product.

Ans. to the Q. No → 8

The software process improvement cycle (often called PDCA - plan - Do - Check - Act) is a structured approach to enhance software development. It involves:

1. Plan: Identifying areas for improvement, setting goals, and planning actions.
2. Do: Implementing changes and collecting data.
3. Check: Analyzing data to see if changes are effective.
4. Act: Standardizing successful changes or

revising the plan.

Common process metrics include:

i. Size: Lines of Code (LOC), Function Points

ii. Effort: Person-hours, Cost

iii. Quality: Defect Density, Defect Rate, Mean Time to Failure (MTTF), Mean Time Between Failures (MTBF)

iv. Efficiency: Productivity, Development Speed

v. Customer Satisfaction: Surveys, Feedback

Metrics help by providing a baseline, identifying trends, measuring the impact of changes, enabling data-driven decisions, and facilitating communication. They're crucial for continuous improvement, leading to better software.

Ans. to the Q. No → 9:

The SEI CMM is a framework for software process improvement with five maturity levels:

1. Initial: Ad-hoc, chaotic, individual effort.
   contribution: Recognizes need for process.

2. Managed: Basic project management, tracking cost/schedule.
   Contribution: Brings discipline, increase predictability.

3. Defined: standardized, documented process.
   contribution: ~~Brings~~ Consistency, improved quality.

4. Quantitatively Managed: Data-driven control. metrics used.
   contribution: Targeted improvements, informed decisions.

5. Optimizing: Continuous process improvement.
   Contribution: Increased efficiency, quality, satisfaction.

Each level builds upon the previous, leading to better project management, higher quality software, and improved organizational performance.

Ans. to the Q. No → 10

Agile Software development values flexibility, collaboration, and iteration. It's core principles are: individuals / interactions over processes / tools; working software over documentation; customer collaboration over contracts; and responding to change over plans.

Agile is applied in various environments, from startups to enterprises, and is well suited for web and mobile development.

Benefits: Faster time to market, increased flexibility, higher quality, improved customer satisfaction, boosted team morale.

challenges: Requires strong communication, can be difficult to scale, needs customer involvement, potential for scope creep, document-

ation balance.

Agile is ideal for complex, evolving projects with customer involvement, but may be over-kill for simple projects. Its suitability depends on the project, organization, and stakeholder ~~by~~ buy-in.

Ans. to the Q. No → 11

Here's an overview of the Extreme Programming release cycle along with its influential programming practices:

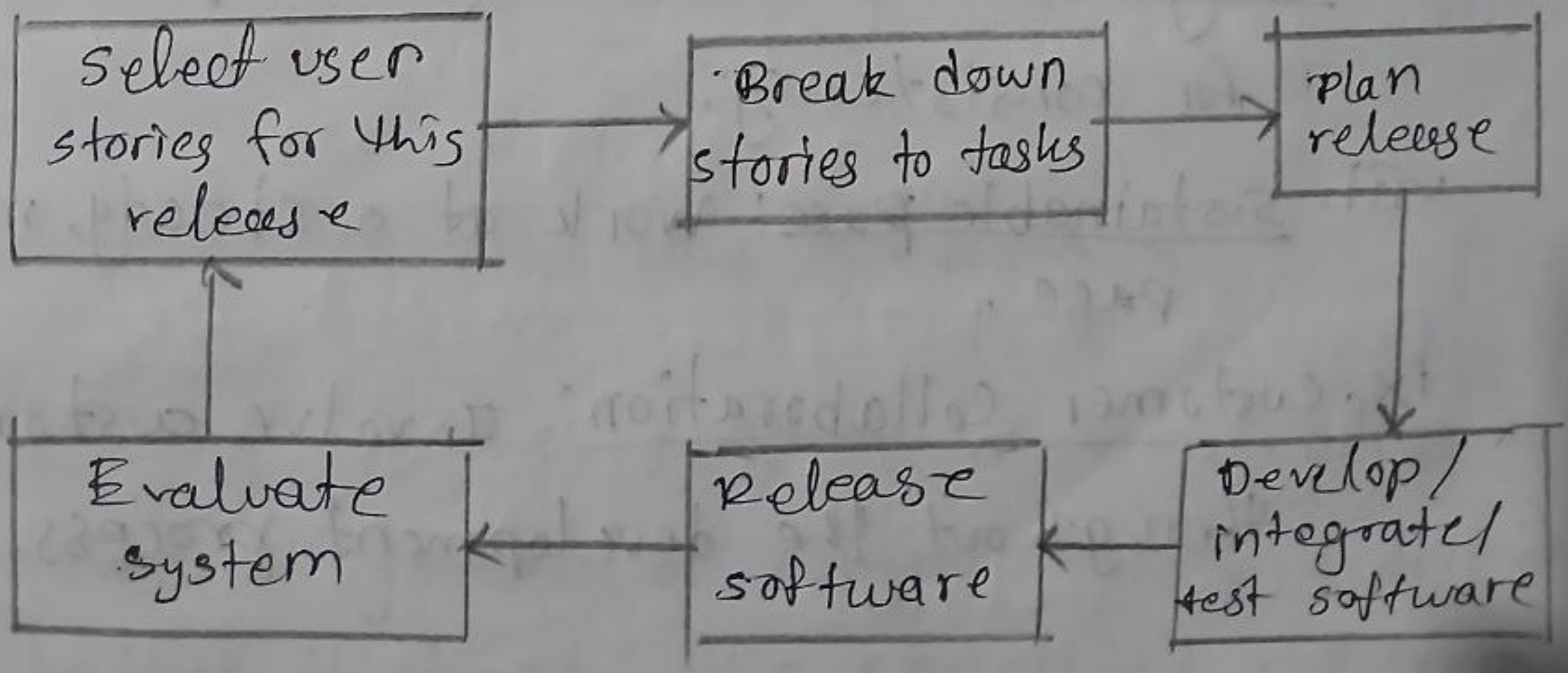


Fig: The XP release cycle

The influential programming practices in XP are:

i. Pair Programming: Two developers work together at one workstation to improve code quality.

ii. TDD: Write tests before coding to ensure robust code.

iii Continuous Integration: Regularly integrate code changes and run automated tests.

iv. Refactoring: Continuously improve code without changing functionality.

v. Simple Design: Create the simplest solution that works.

vi. Collective code Ownership: Any team member can modify any part of the code.

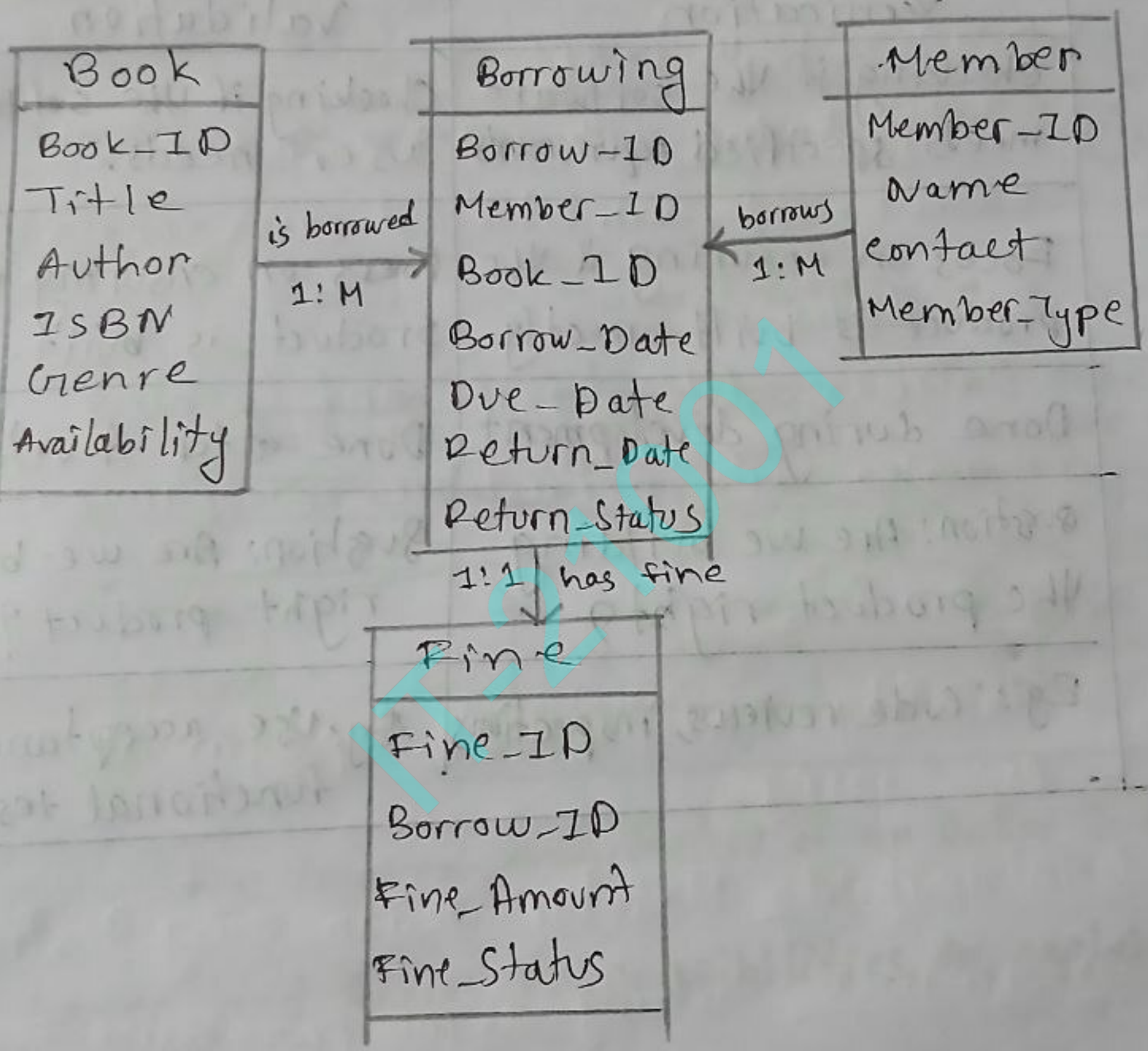
vii. Coding standards: Follow agreed-upon practices for consistency.

viii. Sustainable pace: Work at a steady, manageable pace.

ix. Customer Collaboration: Involve customers throughout the development process.

Ans. to the Q. No > 12

The Entity-Relationship Diagram using the given scenario is shown below:



Ans. to the Q. No → 13

Testing is the process of evaluating a software system or component to determine whether it meets the specified requirements

and to detect defects.

Differentiation between validation and verification:

verification	validation
checking if the software meets specified requirements	Checking if the software meets user needs.
Focus on ensuring if the product is built correctly.	Focus on ensuring the right product is built.
Done during development	Done after development
Question: Are we building the product right?	Question: Are we building the right product?
Eg: Code reviews, inspections	Eg: Use acceptance testing functional testing.

Ans to the Q. No → 14

Layered Architecture model for an online judge system identifying the key layers is shown below:

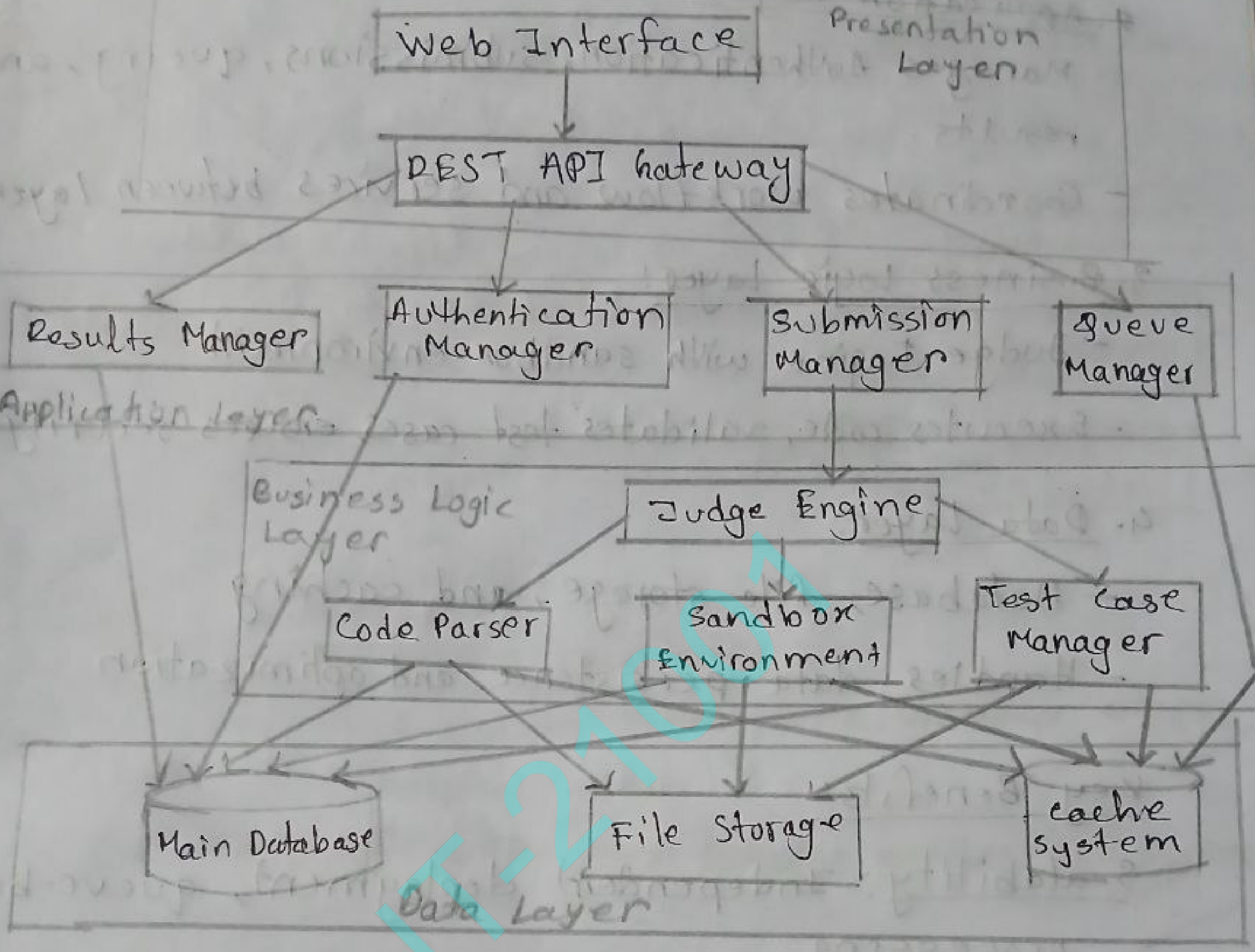


Fig: Layered Architecture of an Online Judge.

Each Layer and their responsibilities are explained below:

1. Presentation Layer:
   - Web Interface & REST API Gateway
   - Handles user interaction, requests, validation, and sessions.

2. Application Layer:
- Manages Authentication, submissions, queing, and results.
- Coordinates workflow and services between layers.

3. Business Logic Layer:
- Judge Engine with sandbox environment
- Executes code, validates test cases, ensures security

4. Data Layer:
- Database, file storage., and caching
- Handles data persistence and optimization

Key Benefits:

Scalability: Independent deployment, queve-based processing

Maintainability: Modular design, easy testing

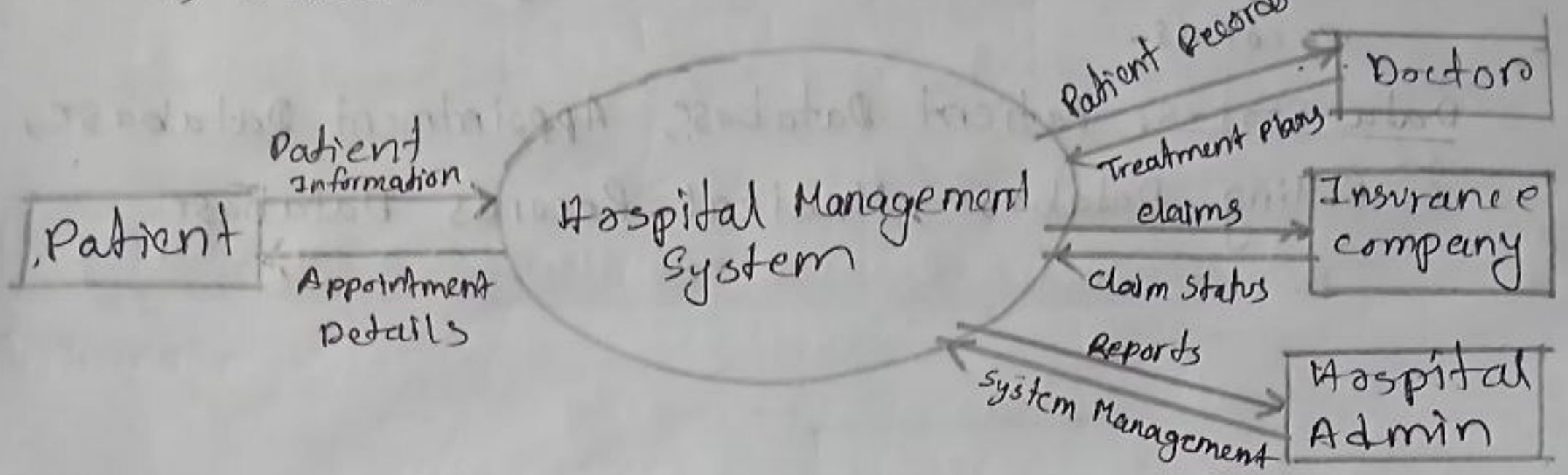
Performance: Caching, asyne processing, load balaneing

Security: Isolated execution, multi-layer validation

This design separates concerns while maintaining workflow between components.

Ans. to the Q. No > 15

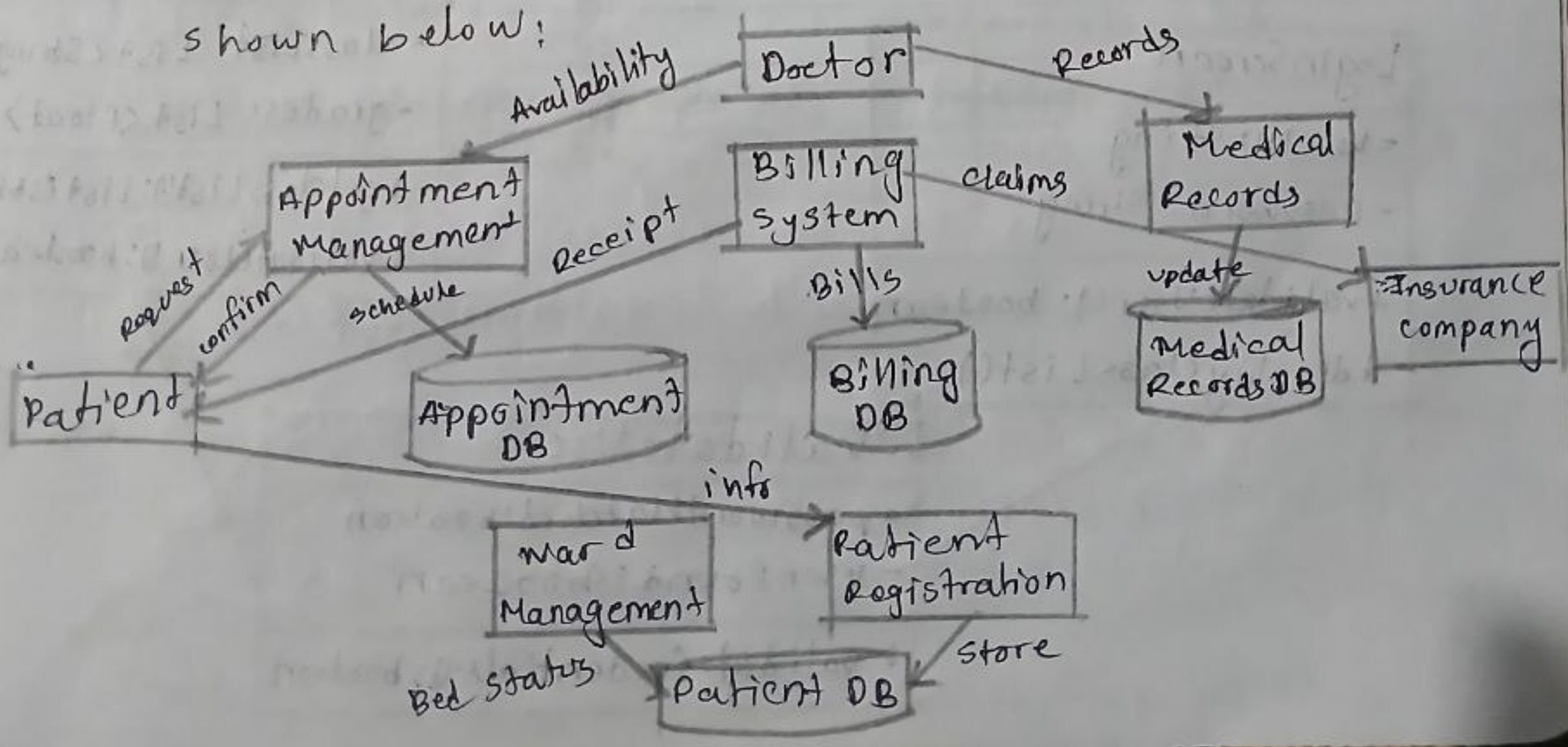
0-level DFD of Hospital Management System is shown below:



In 0-level DFD:

- Shows system boundaries
- External entities: Patient, Doctor, Insurance company, Hospital Admin
- Basic information flows between entities and system.

1-Level DFD of Hospital Management System is shown below:



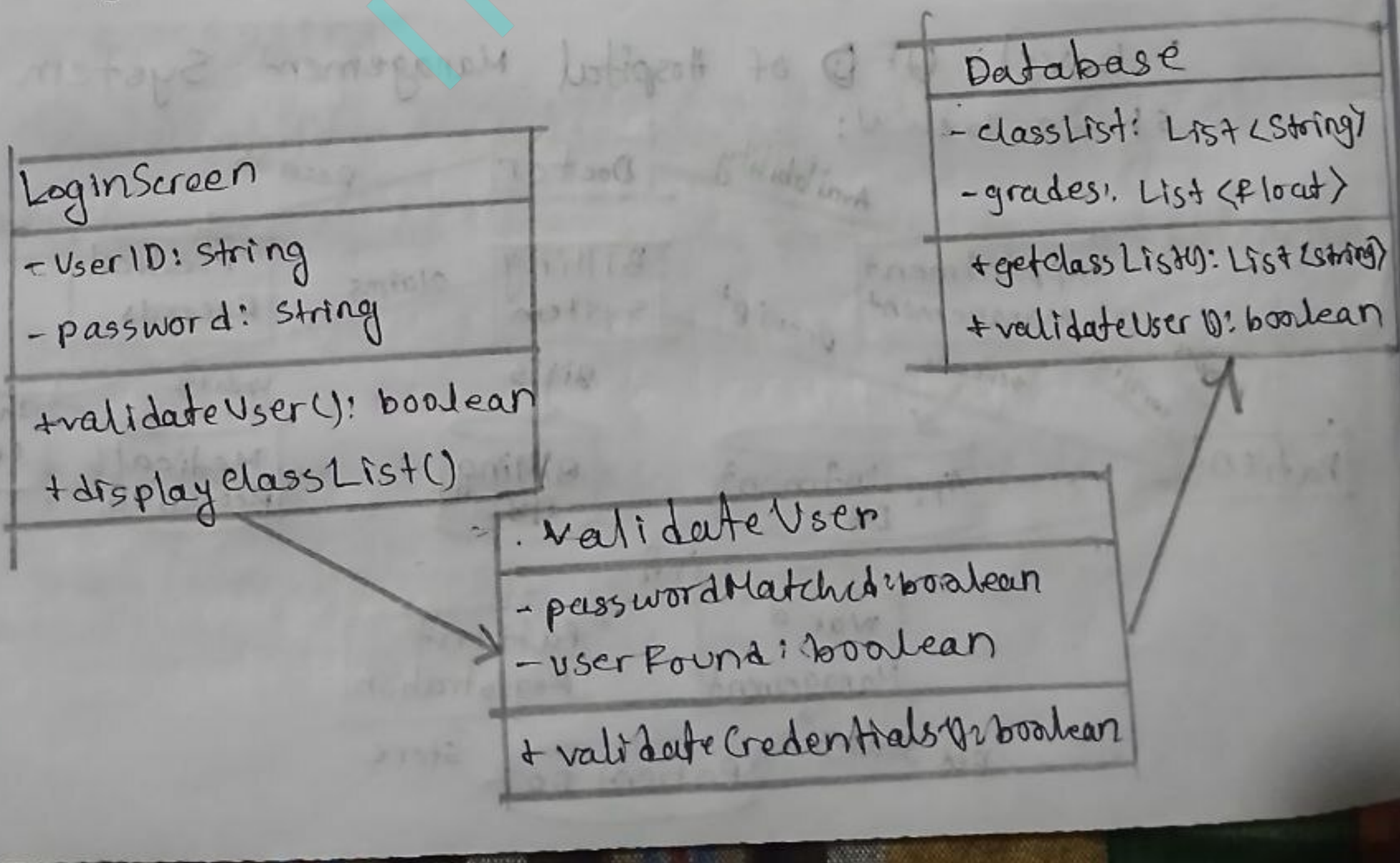
In Level-1 DFD:

Major Processes: Patient Registration, Appointment management, Ward management, Billing System, Medical Records.

Data stores: Patient Database, Appointment Database, Billing Database, Medical Records Database

Ans. to the Q. No → 16

A UML class diagram based on the sequence diagram is shown below:



Ans. to the Q. No → 17

Quality Assurance (QA) is a process-oriented, focusing on preventing defects by establishing and improving the software development process. It involves planning, documentation, training, and audits to ensure quality is built-in.

Quality Control (QC) is product-oriented, focusing on identifying defects in the software product itself. It uses testing, inspections, and reviews to detect and correct bugs.

Differences between QA and QC are!

Feature	QA	QC
Focus	Process	Product
Nature	Proactive (prevention)	Reactive (detection)
Goal	Build quality into process	Find and fix defects

Impediments!

QA: Lack of management support, inadequate training, poorly defined processes, resistance to change, time/budget constraints.

QC: Insufficient testing, tight deadlines, lack of resources, inadequate bug tracking, communication issues.

Ans. to the Q. No → 18

Role of QA at each phase of the SDLC is mentioned below!

1. Requirement Analysis!
   - QA activities! Review requirements for clarity and testability.
   - Goal: catch issues early.

2. Design!
   - QA activites! Review design documents and prepare test plans.
   - Goal! Ensure design meets requirements.

3. Implementation (Coding):
   - QA activities! Develop test cases and review code.
   - Goal! Ensure design meets requirements

4. Testing!
   - QA activities! Execute tests to check functionality.
   - Goal! verify software meets requirements.

5. Deployment!
   - QA activities! Test the release and check configurations.
   - Goal! Ensure smooth transition to production.

6. Maintenance!
   - QA activities! perform regression tests and monitor performance.
   - Goal! Maintain quality after release.

Ans. to the Q. No- 19

RAD is a software development model emphasizing speed and iteration.

key phases: i. Business modeling,
ii. Data Modeling
iii. ~~App~~ Process Modeling
iv. Application Generation
v. Testing / Turnover

Principles: User involvement, iterative development, working software focus, timeboxing, prototyping.

Advantages: Faster delivery, reduced costs, improved user satisfaction, increased flexibility, reduced risk.

Limitations: Requires user availability, strong teamwork, may not suit complex projects, potential for scope creep.

Ans. to the Q. No→20

A student clicks the Test case table:

Decision	x input	y input
y==0	5	0
x==0	0	3

JUnit Test Class in Java

```
import org.junit.jupiter.api.Test;
import java.io.ByteArrayOutputStream;
import java.io.PrintStream;
import static org.junit.jupiter.api.Assertions.*;

public class DecisionTest {
    private void process(int x, int y, C c){
        if (y==0) c.println("y is zero");
        else if (x==0) c.println("x is zero");
        else for(int i=1; i<=x; i++)
            if (i%y==0) c.println(i);

    @test void test_y_is_zero(){ ... }
    @test void test_x_is_zero(){ ... }
}
```

Ans. to the Q. No → 21

To test code with JUnit 4, use @Before for setup, @Test for tests, @Test(expected=Exception.class) or try-catch for exceptions, and @Rule Timeout for timeouts. Test valid/invalid inputs and exceptions. Assertions (assertEquals, etc.) verify outcomes. Example: A Calculator class's divide

method can have tests for normal input, divide-by-zero (using both exception-handling methods), and a timeout test for a long operation. Good test names and specific assertions are key.