Ans. to the Ques. no - 1

As a user, if I want to log in securely so that I can access my account, the tasks I need to do is:

1. Design login page Layout & UI elements.
2. Implement authentication API.
3. Create user session management.
4. Implement password hashing and encryption for security.
5. Create error handling for incorrect login attempts.
6. Test login functionality with valid and invalid credentials.
7. Integrate with database for user credentials validation.
8. Perform security testing.

As a user, if I want to search for products by category to find items easily, the tasks I need to do is:

1. Design search page layout with category filter options.
2. Create database schema to categorize products.
3. Develop search API for querying products by category.
4. Implement front-end filtering by category.
5. Integrate search results display and pagination.
6. Optimize search performance.
7. Test responsiveness and usability of the search feature.

Prioritizing user stories during sprint planning:

During the sprint planning meeting, the product owner and the development team would collaboratively prioritize these user stories based on two key factors:

1. Value to the customer:

The log in feature is often considered foundational for most e-commerce platforms, as

without secure authentication, users cannot access their accounts or make purchases.
This user story is likely of high priority to ensure secure access and trust in the platform.
→ The search by category feature adds significant value to the user experience, especially for an e-commerce app, as it helps users find products quickly. This is important but might not be as critical as the login feature for initial use of the application.
→ priority Decision: The log in feature may take precedence in the sprint since it's essential for account access and ensuring a secure environment for customers, but both user stories are crucial for a fully functional e-commerce platform.

2. Technical Feasibility:
→ The log in feature involves user authentication and security, which could require external dependencies and ensuring security best prac-

tices might involve more technical effort. However, it's a standard feature and can often leverage existing frameworks or libraries.

→ The search feature involves database integration, filtering and possibly scaling to handle a large product catalog, which might require additional design and performance considerations.

Tracking Tasks on the scrum Board:

The scrum board is a visual tool used to track the progress of tasks throughout the sprint. Tasks for both user stories will be organized in columns. To Do, In Progress and Done.

1. To Do: This column contains tasks that have been defined but have not yet started. At the start of the sprint, all tasks for the user's stories will be listed here.

2. In Progress: As the development team begins working on the tasks, they will move them to this column. For example, once the UI design work for the login page starts, the task "Design login page layout" will be moved here. Similarly, tasks like "Implement authentication API" and "Develop search API" will be tracked in this column when the team works on them.

3. Done: When a task is completed, tested and reviewed, it will be moved to the "Done" column. For example, after the login functionality is thoroughly tested, the task "Test login functionality" will be moved here. Tasks like "Optimize search performance" and "Test Search functionality" will also appear in this column once finished.

Ans. to the Ques. no-2

Risk management and adaptability in spiral, Agile and Extreme methodologies:

1. Spiral methodology:

→ Risk management : The spiral methodogy is designed for projects with ~~login~~ high risk and uncertainty. It emphasizes a cyclic process of planning, risk analysis, engineering, testing and evaluation. Each cycle involves assessing and managing risks through prototyping and feedback from stakeholders.

→ Adaptability: The spr spiral methodology is adaptable because it allows for continuous refinement and the inclusion of involving requirements. The iterative nature of the cycles enables the project to adjust as new information emerges

2. Agile methodology:

→ Risk management : Agile methodologies address risk by breaking down the project into smaller, manageable chunks and delivering incremental improvements. Each iteration focuses on delivering a potentially shippable product increment which allows for early and frequent delivery, testing and feedback.

→ Adaptability: Agile thrives on adaptability, allowing for continuous feedback loops with clients at the end of each iteration. This frequent review ensures that the product can be adjusted quickly based on the client's envolving needs.

3. Extreme Programming methodology:

→ Risk management: Extreme programming (XP) is a highly disciplined approach that focuses on engineering excellence and continuous feedback. It aims to reduce risks through

practices like Test-Driven Development (TDD), continuous integration (CI), and pair programming. These practices ensure that the software is always in a working state, which mitigates technical risks and helps identify issues early.

→ Adaptability: XP promotes adaptability by encouraging continuous communication with the client, ensuring that customer feedback is rapidly integrated. It's iterative cycles, combined with practices like refactoring and simple design, allow teams to respond quickly to changes.

Most suitable methodogy for High-Risk and Envolving Requirements:

Agile is likely most suitable methodology for a project with significant risk and envolving requirements for several reasons:

(1) Flexibility and Iterative process: Agile's iterative approach allows for frequent reassessment of the product and requirements. As the project progresses, the development team can adapt to changing needs or emerging risks. This is especially valuable when client needs are uncertain or subject to change.

(2) Continuous feedback and stakeholder involvement: Agile emphasizes frequent communication with stakeholders through regular meetings.

(3) Risk mitigation via incremental delivery: By delivering small, manageable increments, Agile reduces the likelihood of major project failures. The team can focus on delivering valueable functionality early.

(4) Adaptability to changing requirements: The Agile methodology allows for flexibility in managing changes to requirements throughout the development process.

While spiral offers strong risk management, it can be resource-intensive, and XP focuses more on technical practices that may not directly ~~ddre~~ address higher level strategic risks. Agile, with its emphasis on adaptability, client feedback and iterative development, is the best fit for handling both the high risks ~~en~~ and envolving requirements of the projects.

Thus, Agile provides the best balance of risk management and adaptability for projects where requirements are uncertain and change over time.

Ans. to the Ques. no-3

Comparison of Waterfall, Agile, Extreme & spiral methodologies:

① Waterfall:

→ Characteristics: Waterfall is a sequential development process where each phase must be completed before moving on to the next. It's structured and rigid, with little room for changes once a phase is completed.

→ Risk management: Waterfall is not inherently built for managing risks, as it typically focuses on upfront planning. Any changes or issues that arise later in the process are more dificult and costly to address.

→ Predictability: It offers high predictability due to its rigid structure and clearly defined stages. The project timeline, scope and deliverables and established early on.

→ Customer Collaboration: There is limited customer collaboration.

② Agile:

→ Characteristics: Agile is iterative and incremental, focusing on delivering small, functional increments of the product in short cycles.

→ Risk management: Agile manages risks by breaking down the project into smaller tasks and allowing for regular review and adaptation.

→ Predictability: Offers less predictability.

→ Customer Collaboration: Agile is centered around frequent customer feedback and collaboration.

③ Extreme Programming (XP):

→ Characteristics: XP emphasizes Test-Driven Development; pair programming and continuous integration to deliver high quality code.

→ Risk management: Can. mitigates technical risks by ensuring code quality.

→ Predictability: Similar to agile but XP is less predictable.

→ Customer Collaboration: XP involves the customer frequently.

④ Spiral:

→ Characteristics: Spiral is a risk-driven development methodology. It involves frequent cycles of planning, risk analysis, engineering and testing.

→ Risk management: It is the best for managing risks.

→ Predictability: Spiral also introduces a level of predictability by continuously reviewing and refining the project as it progresses.

→ Customer collaboration: ~~Spiral~~ Spiral allows for ongoing customer feedback during each iteration

Best methodologies for projects A and B:

For project A, which has well-defined requirements and a strict deadline, the waterfall methodology is likely the most suitable choice. Here's why:

→ Predictability and structure: Waterfall's rigid linear approach is ideal when the project requirements are well understood upfront.

→ Clear Timeline and Deliverables: Since the requirements are fixed, waterfall allows the team to establish a clear, detailed timeline for each phase of the project.

→ Customer Collaboration: Although waterfall doesn't focus on ongoing collaboration, since the requirement are clear from the outset,

→ Risk management: While waterfall lacks flexibility in responding to change, the well-defined requirements and non-evolving nature of the project reduce the need for extensive risk management.

Project-B:

For project B, which has evolving requirements and requires continuous feedback, an Agile methodology would be the most suitable. Here's why-

→ Adaptability to change: Agile's iterative nature allows the development team to adapt to evolving requirements as they arise.

→ Customer Collaboration: Agile places a strong emphasis on continuous collaboration with the customer.

→ Risk management: By breaking the project into smaller chunks, agile reduces the overall risk of failure.

→ Flexibility with Timeline: Although Agile doesn't guarantee a fixed timeline, it's short cycles allow for quick deliverables and continuous refinement. This helps manage uncertainty. ~~and~~

→ Extreme programming (XP): If the project involves technical complexity or requires high quality code with frequent feedback,

Ans. to the Ques. no-4

Principles of software engineering ethics:

Software engineering ethics is about making responsible and moral choices when creating software, key principles include:

① Public welfare: Software should be safe and beneficial to society, ensuring that users privacy and security are ~~proc~~ protected.

② Professional Responsibility: Engineers must be responsible for their work.

③ Fairness: should treat everyone fairly.

④ Confidentiality: Engineers must keep sensitive information private and secure.

⑤ Honesty:

Issues related to professional responsibility:

1. Accountability
2. Conflict of interest
3. Negligence
4. Intellectual property.

ACM/IEEE code of Ethics and decision-making:

1. Public interest.
2. Honesty
3. Quality
4. Confidentiality.
5. Avoiding Harm
6. Learning.

How the code guides ethical decisions:

The ACM/IEEE code of ethics helps engineers decide what to do in tricky situations, like when they face pressure to release software quickly or hide problems. For examples:

→ If a software flaw is found, the engineer should prioritize fixing it rather than rushing to meet deadlines.

→ Engineers should be honest with clients about software issues, even if it causes delays.

→ If a product could harm user's privacy, the engineer should address that, even if it's hard.

Ans. to the Ques. no - 5

Functional requirements for the airport reservation systems:

① Flight Search:

→ The system should allow users to search for available flights based on parameters like destination, departure date, and number of passengers.

② Booking a Flight:

→ Users should be able to select a flight and make a reservation by entering person and payment details.

③ Payment Processing:

→ The system should securely handle payment transactions, including credit/debit card processing or alternative payment methods.

④ Reservation management:

→ The system should allow users to view, modify or cancel their flight reservations.

⑤ User authentication:

→ The system must verify user identities through login credentials or other methods.

Non-functional requirements for the airport reservation system:

① Performance (Response Time):
→ The system should be able to process flight searches, bookings and payment transactions within a few seconds.

② Scalability:
→ The system should be able to handle increasing numbers of users and transactions as the airline's customer base grows.

③ Availability (Uptime):
→ The system should be available 24/7 with minimal downtime.

④ Security (Data Encryption):
→ All sensitive data, such as payment information and personal details, should be encrypted both in transit and at rest.

⑤ Usability (User interface):
→ The system should have an intuitive and user-friendly interface for booking flights and managing reservations.

Ans. to the Ques. no-6

V-Model of testing phases in a plan-Driven software process:

The V-model (Verification and Validation model) is a software development and testing methodology that emphasizes the relationship between development phases and corresponding testing activities. It is a plan-driven approach where each development phase is directly linked to a testing phase, creating a V-shaped diagram. Here's how it works:

Development Phases	Testing Phases
Requirements Analysis	→ Acceptance testing
System Design	→ System Testing
Architecture Design	→ Integration Testing
Module Design	→ Unit Testing
Coding (Implementation)	

~~Protol~~ Ans. to the Ques. no - 7

Prototype development in software engineering involves creating an early, simplified version of a system to visualize functionalities, gather user feedback and refine requirements. It is a crucial approach for handling unclear or evolving requirements by allowing iterative improvements before the final development.

Key stages ~~of~~ in prototyping model:

1. Requirement gathering & Analysis.
2. Quick Design.
3. Prototype Development.
4. User Evaluation & Feedback.
5. Refinement & Iteration.
6. Finalization & Development.

How prototyping helps in refining software requirements:

→ Clarifies user needs
→ Identifies missing or unclear requirements.
→ Reduces ambiguity.
→ Validates feasibility.

Benefits of the prototyping model:

① User feedback & satisfaction.
② Risk Reduction.
③ Iterative Development.

Ans. to the Ques. no-8

Process improvement cycle in software engineering:

The process improvement cycle in software engineering is a systematic approach to enhancing software development processes to improve quality, efficiency and productivity. It follows an iterative approach to identify weakness, implement improvements and monitor outcomes.

Key Stages:

1. Process Assessment.
2. Process Definition & measurement.
3. Process improvement planning.
4. Process implementation.
5. Monitoring and evaluation.
6. Continuous improvement.

Commonly used process metrics:

1. Effort metrics.
2. Defect Density
3. Cycle time.
4. Customer Satisfaction index.
5. Process Compliance metrics.

How process metrics help in monitoring and improvement:

→ identify bottlenecks
→ Improve quality.
→ Enhance productivity.
→ Enable data.

Ans. to the Ques. no-9

The capability maturity model (CMM) developed by the software engineering institute (SEI) is a framework for assessing and improving software development processes. It provides a structured approach to process maturity, ensuring consistency, predictability and quality in software development.

Five levels of SEI CMM:

1. Level 1 - Initial (chaotic, Adhoc)
2. Level 2 - Repeatable (Managed at project Level)
3. Level 3 - Defined (Standardized processes)
4. Level 4 - Managed (Quantitatively controlled)
5. Level 5 - Optimizing (continuous process improvement)

CMM Level	Contribution to SDP
Level-1 Initial	Unstructured, unpredictable processes lead to inconsistent results.

CMM Level	Contribution to SDP
Level-2 Repeatable	- Reduces project risks by ensuring repeatable success through basic project management.
Level 3 - Defined	- Standardized processes improve collaboration, efficiency and quality.
Level 4 - Managed	- Data-driven decision-making enhances process control and software reliability.
Level 5 - Optimizing	- Continuous improvements lead to innovation, high efficiency, and adaptability.

Ans. to the Ques. no - 10

Core principles of Agile software development:

① Individuals and interactions over processes and tools.
→ Focus on team collaboration and communication rather than rigid workflows.

(2) Working software over comprehensive documentation.

(3) Customer collaboration over contract negotiation.

(4) Responding to change over following a plan.

⇒ Application of Agile principles in different environments:

(1) Startups & small Teams:
→ Agile suits startups due to it's flexibility and rapid iterations.
→ Focuses on quick MVP (minimum viable product) releases and customer feedback.

(2) Enterprise software Development:
→ Large organizations use scaled agile agile framework (SAFe) or scrum of scrums to manage multiple teams.

(3) Regulated industries (Healthcare, finance)
→ Agile is adapted with additional documentation to meet legal and compilance requirements.

④ Embedded systems & Hardware development:

→ Agile is challenging but possible with modular development and incremental software updates.

Benefits of Agile methods:

→ Faster Delivery: Frequent releases ensure quicker value to users.

→ Improved Quality: Continuous testing and feedback reduce defects.

→ Higher Customer satisfaction: User involvement ensures the product meets expectations.

→ Better Team Collaboration: self-organizing team improve efficiency and innovation.

Ans. to the Ques. no - 11

Extreme programming release cycle:

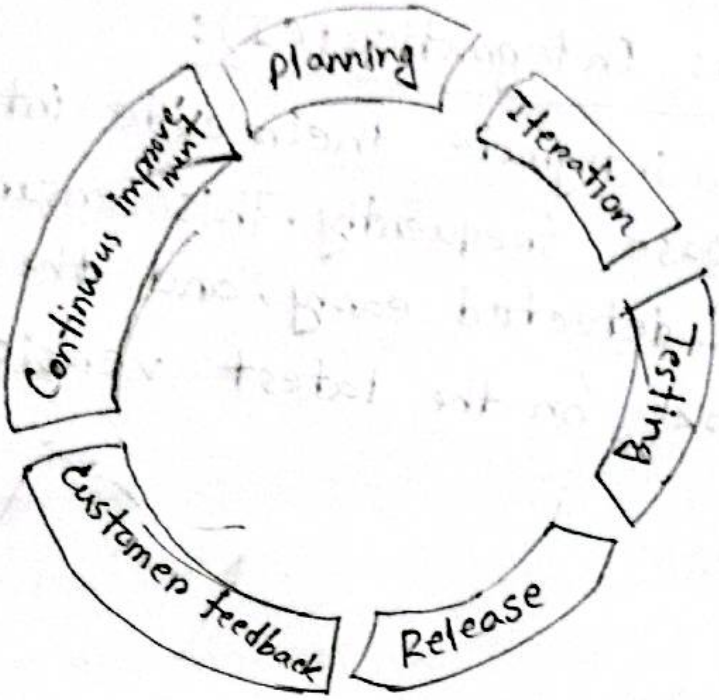


Fig: XP release cycle

Influential programming practices in XP:

(1) Pair programming: Two developers work together at one workstations, one writting the code and the other reviewing it. This practice helps reduce defects and improves collaboration.

(2) Test-Driven Development (TDD): Tests are written before the actual code is implemented. This

ensures that the code meets the requirements and is robust from the start.

③ Continuous Integration (CI):

Developers integrate their code into the main codebase frequently. This ensures that issues are detected early, and the team always works on the latest version of the code.

Ans. to the Ques. no - 13

Testing is the process of evaluating a system or it's components to identify defects, ensure functionality and verify that it meets specified requirements. It involves executing the software with test cases to detect errors, gaps or missing requirements.

Difference between validation and verification:

Verification	Validation
① Ensures the product is being built correctly.	① Ensures the right product is being built.
② Focuses on reviews, inspections and walk-throughs.	② Actual execution and testing of the product.
③ Performed before development or during early stages.	③ Here, ~~early~~ after development, during or after testing.
④ Static methods.	④ Dynamic methods.
⑤ Ex: Checking design documents, requirements.	⑤ Running test cases on the software.

Ans. to the Ques. no - 12

Entity-Relationship Diagram (ERD) for the Library management System:

Entities & their attributes:

① Book:

⇒ Attributes:

↳ Book_ID (Primary key)

↳ Title

↳ Author

↳ ISBN

↳ Genre

② Member:

⇒ Attributes:

↳ Member_ID (Primary key)

↳ Name

↳ Contact_Details

③ Borrowing Activity:

⇒ Attributes:

↳ Borrowing_ID (Primary key)

↳ Borrow_ID

↳ Return_Due_Date

↳ Return_Status

④ Overdue_Fine :

⇒ Attributes:

↳ Fine_ID (Primary key)

↳ Amount

↳ Borrowing_ID

↳ Fine_Date

Relationships :

① Member to Borrowing Activity :

→ One to many (A member can borrow many books over time, but each borrowing activity is linked to a single member.

② Book to Borrowing Activity :

→ One to many (A book can be borrowed multiple times).

③ Borrowing Activity to Overdue Fine :

→ One to one or One to many (A borrowing can have at most one fine, but a fine is always tied to one borrowing activity).

Ans. to the Ques. no-14

In an online judge system, a layered architecture can help organize the system into distinct components. Each layer has it's own specific responsibilities, making the system more modular, scalable, maintainable and efficient.

① Presentation Layer (User Interface Layer):
- ↳ User interaction
- ↳ Displaying results
- ↳ Input collection
- ↳ Authentication
- ↳ Technology.

② Application Layer (Controlled Layer):
- ↳ Request Handling
- ↳ Session management
- ↳ Orchestration

③ Business Logic Layer (service Layer)
- ↳ Code evaluation
- ↳ Grading system
- ↳ Error handling
- ↳ Test case management

④ Data Layer (Persistence Layer):

↳ Database management

↳ Persistent storage

↳ Data Retrieval

Architecture Diagram:

Presentation Layer (User Interface)

↓

Application Layer (Request handling, Session management)

↓

Bussiness Logic Layer (Code Evaluation, Grading)

↓

Data Layer (Database, Persistent storage)

This architecture ensures:

① Scalability

② Maintainability

③ Efficient Performance

Ans. to the Ques. no - 15

Level-0 Data Flow Diagram (DFD):

→ Provides a high-level overview of the system, showing the major processes and interactions between the system and external entities.

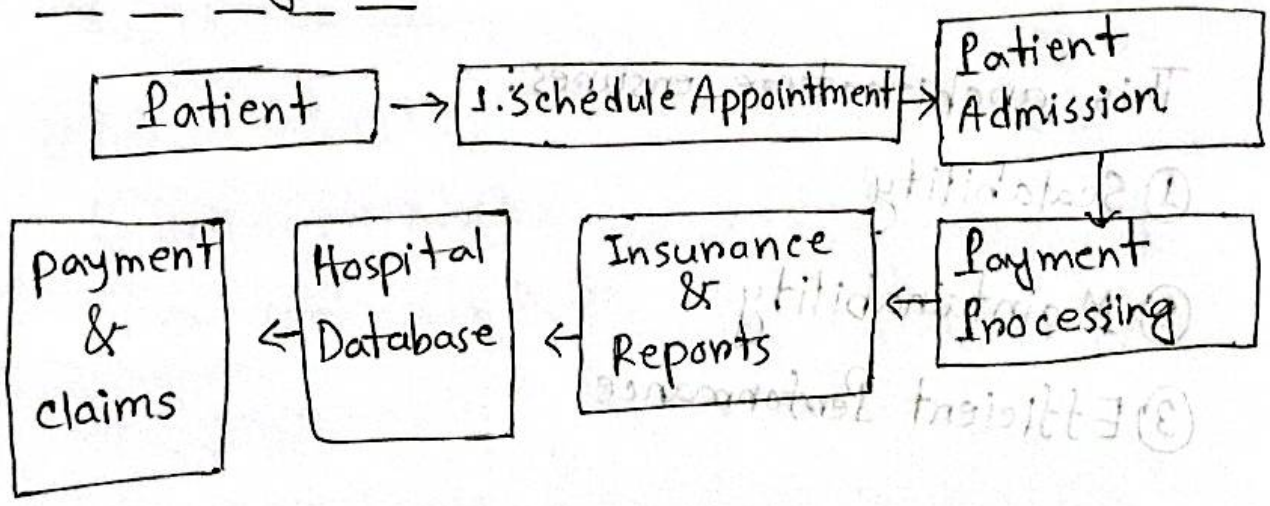
Entities:

① Patient.
② Receptionist.
③ Hospital Database.
④ Insurance company.

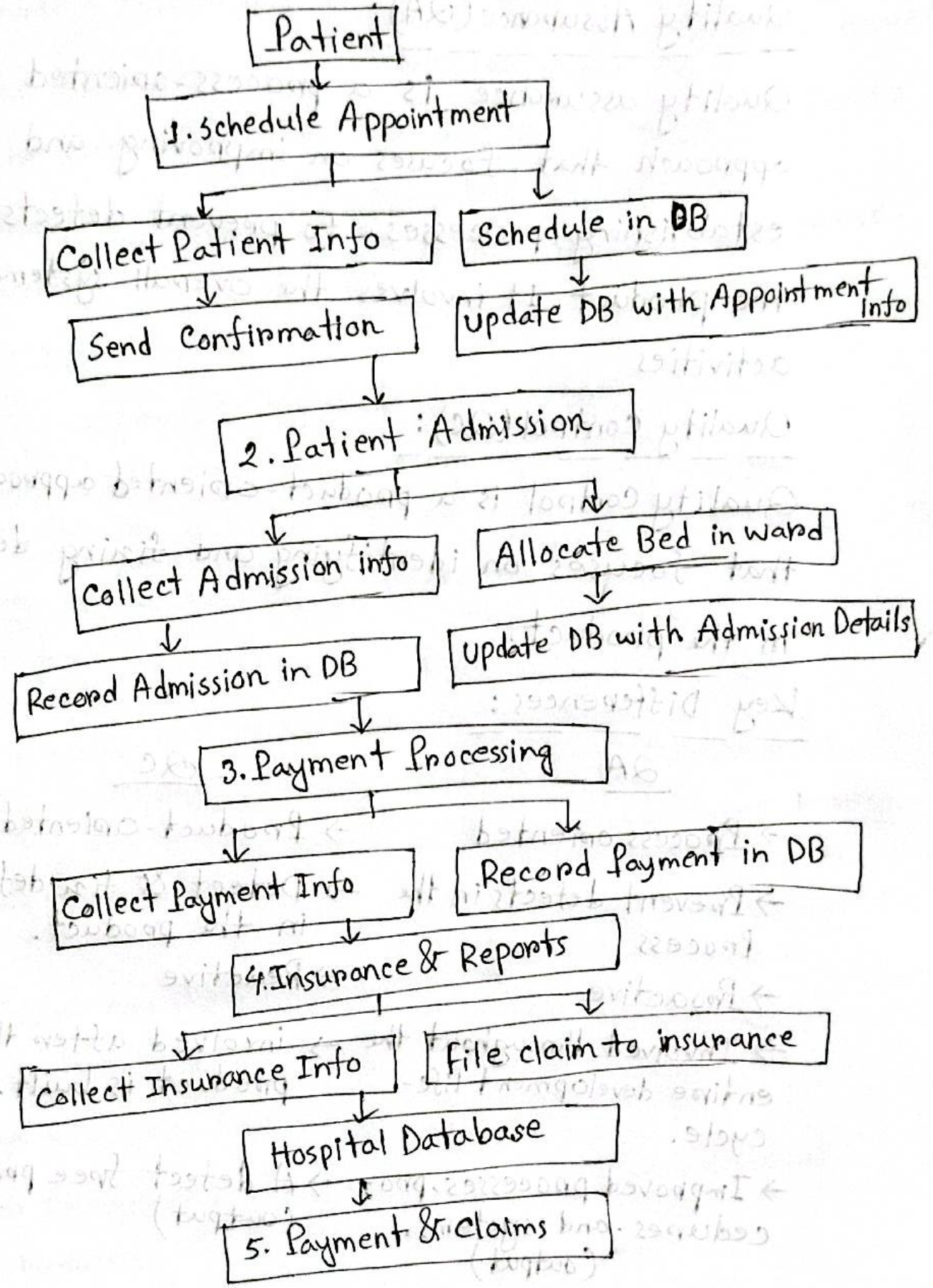
Processes:

① Schedule Appointment
② Patient Admission
③ Payment processing
④ Insurance claims & Reports

DFD Diagram:



Level-1 Data Flow Diagram:



Ans. to the Ques. no - 17

Quality Assurance (QA):

Quality assurance is a process-oriented approach that focuses on improving and establishing processes to prevent defects in the product. It involves the overall systematic activities

Quality Control (QC):

Quality Control is a product-oriented approach that focuses on identifying and fixing defects in the product.

Key Differences:

QA	QC
→ Process oriented	→ Product-oriented
→ Prevent defects in the Process.	→ Detect & fix defects in the product.
→ Proactive	→ Reactive
→ Involved throughout the entire development life-cycle.	→ involved after the product is built.
→ Improved processes, procedures and systems. (output)	→ A defect free product (output)

Ans. to the Ques. no - 18

Quality Assurance plays a critical role throughout the software Development Life cycle (SDLC) to ensure the product meets the desired quality standards and functions as expected.

Here's a breakdown of QA's role at each phase of the SDLC:

① Requirements gathering & analysis:
→ requirements review and verification
→ identifying any gaps or inconsistencies
→ Suggesting improvements or clarifications.

② Design:
→ reviewing design documents for testability.
→ Verifying that non-functional requirements
→ identifying potential risks or challenges early on.

③ Implementation/Development:
→ Static Code analysis.
→ writing and executing unit tests.
→ Performing integration testing on individual modules.

④ Testing:
→ System testing
→ identify defects

⑤ Deployment & maintenance:
→ Validates the final product
→ Monitors performance, security and user feedback for continuous improvements.

Ans. to the Ques. no - 19

The Rapid Application Development (RAD) model is an iterative and adaptive software development approach that focuses on fast delivery while ensuring quality and user satisfaction.

Key phases of the RAD model:

① Business modeling:
② Data modeling
③ Process modeling
④ Application generation
⑤ Testing & Deployment.

Principles of RAD:

→ User involvement
→ Prototyping.
→ Iterative Development
→ Component Reusability.

Advantages of RAD model:

→ faster delivery.
→ Improved user satisfaction
→ flexibility.
→ Higher quality.

RAD supports faster and high quality software delivery::

The RAD model accelerates development by minimizing rigid planning, using automation tools and engaging users ~~do~~ early. By focusing on incremental updates and frequent testing, it ensures that the software is both quickly delivered and highly reliable maintaining user satisfaction throughout the process.

Ans. to the Ques. no-21

To develop test codes using JUnit 4 for function testing while incorporating exception handling, setup function and timeout Rule.

① Setting up JUnit 4 in IDE:

Firstly, need to add Junit4 correctly to IDE (e.g., Eclipse, IntelliJ or Vs' code).

② Production code example:

③ Writing JUnit 4 Test cases

④ This JUnit test effectively validates exception handling, setup function initialization and timeout constraints for unit testing production code.