Sadia Chowdhury
IT21062

Ques 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.

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

ii) Describe how the development them can pri'oritize these user stories during a sprint planning meeting. Considering value to the customer and technical feasibil

iii) Illustrate how these tasks will be tracked using a scrum board. Use proper terms like "To Do", "In Progress" and "Done".

Answer:

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 them complete each user story.

User story 1: 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, OAuth, or session-based authentication)

Task 4: Implement password hashing and secure storage (e.g. using bcrypt or Argon2)

Task 5: Implement session management (e.g. handling login states with cookies or tokens)

Task 6: Write unit tests for the login functionality (e.g. successful login, incorrect password, user not found)

Task 7: Perform security testing (e.g. vulnerability checks like SQL injection, cross-site scripting)

Task 8: Implement logout functionality and session expiration.

Scanned with CamScanner

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

Task 1 : Design the search UI with category dropdown or filter panel.

Task 2 : Create a database schema for product categories.

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

Task 4 : Integrate fronted with the search API to display products dynamically based on selected categories.

Task 5 : Write unit tests for product search functionality.

Task 6 : Optimize search query for performance (e.g. indexing categories or using caching)

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

II

During sprint planning, the team will priotitize these user stories based on :

Value to the customer :

User Story 1 (login): High priority. Secure login is crucial for user trust and account access, enabling care functionalities like order history, saved items and personalized recommendations.

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

Technical feasibility:

User story 1 (log in): Moderate complexity. Requires careful implementation of security measures and integration with authentication services.

User story 2 (category search):

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

Scanned with CamScanner

(III)

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 or deployment.

Scanned with CamScanner

Question 2:

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

Considering the high risk management and adaptability, Which methodology would be the most suitable for a project with significant risk and evolving requirements, and why?

Solution:

How the spiral, Agile, and Extreme Programming (XP) methodologies Address Risk Management and Adaptability:

1. Spiral Model: The spiral model is highly effective in managing risks, especially in project with significant novelty and uncertaintly.

Scanned with CamScanner

Risk Management: This model explicitly focuses on identifying, analyzing and mitigating risks at each iteration or 'spiral'.

Adaptability: The iterative nature of the spiral model allows for frequent reassessment of goals and alignment with client needs.

Limitations: It can be complex to implement and may requires significant documentation and risk analysis expertaise. The model can become expensive if iterations and risk analysis are not well controlled.

2. Agile Methodology:

Agile methodologies, such as scrum, focus on flexibility and collaboration to address evolving requirements.

Risk Management: Risks are reduced through short development cycles and frequent delivery of functional software, ensuring that issues are identified and resolved early.

Scanned with CamScanner

Adaptability: Agile thrives in environment where client requirements are likely to change. The use of a product backlog ensures the team prioritizes features that align with the clients evolving needs.

Limitations: Agile depends on a high level of collaboration and client involvement, which may not always be feasible. If risks are highly technical or require deep upfront analysis, Agiles. Focus on short iterations might not address them adequately.

3. Extreme Programming (XP):

XP emphasizes technical excellence and adaptability through constant feedback and iterative development.

Risk management: Practices like pair programming continuous, integration and test-driven development (TDD) ensures that high-quality code & mitigate technical risks.

Adaptability: Like Agile, XP embraces envolving requirements, with customer. feedback incorporated

Scanned with CamScanner

after each iteration. Simple design and refac-toring ensure the product can adapt to chang ing needs without become overly complex or expensive.

Limitations: XP requires strong discipline in following its practices, and deviations can lead to technical dept. It is more suited to smaller teams with close customer collabo-ration.

Recommended Methodology: Agile Given the high risks and evolving requirements, Agile methodo-logy is the most suitable for this project.

Reasons:

1. Adaptability to change: Agile is designed to handle evolving client requirements through incremental development and regular feedback. It ensures that each sprint delivers value and aligns with updated client priorities.
2. Risk Mitigation through iteration: Frequent testing and integration reduce technical risks.

Scanned with CamScanner

3. Collaboration : Agile emphasizes collaboration between the development team and the client ensuring constant communication and alignment.

4. Cost effectiveness : Agiles iterative nature prevents the project from veering too far from its goals, saving costs associated with late stage corrections.

While the spiral Model excels in projects with a heavy emphasis on risk analysis, its complexity & cost make it less practical for projects with rapidly changing requirements. XP is a strong contender but may be less effective for managing risks is larger teams or highly uncertain environments where frequent client involvement is not guaranted.

Scanned with CamScanner

Solution of Question 03:

project A: Well-Defined requirements, strict Deadline so I reccommended methodology is Waterfall Model.

Why waterfall?

i) Predictability: With well-defined requirements the sequential nature of waterfall ensures a structured and timely delivery.

ii) Clear Milestones: The strict deadlines are easily managed as each phase has specific deliverables.

iii) Minimal change: Since requirements are fixed, the rigidity of waterfall is not a limitation but an advantage.

project B: Evolving Requirements, Uncertain Timeline, continuous Customer Feedback. So this is Agile Model.

Why Agile:

Scanned with CamScanner

i) Adaptability: Agile thrives in environments with evolving requirements, enabling the team to adapt to customer feedback during each sprint.

ii) Collaboration: Continuous customer feedback ensures the product aligns with changing expectations.

iii) Incremental Delivery: Allows for delivering functional increments of the product, keeping stakeholders engaged and reducing risks.

Scanned with CamScanner

Solution of Ques 04:

Software engineering ethics are a set of moral guidelines that govern the professional conduct of software engineers. These principles emphasize responsibility, accountability, and the well-being of society. Key principles include:

i) Public Interest
ii) Client and Employer
iii) Product Quality
iv) Professionalism
v) Confidentiality
vi) Intellectual Property
vii) Social Impact

Professional Responsibility Issues:

Software engineers face numerous ethical challenges:

i) Data privacy
ii) Algorithmic Bias
iii) Software reliability
iv) Cyber Security
v) Social Justice

Scanned with CamScanner

ACM/IEEE Code of Ethics:

The ACM/IEEE Code of Ethics provides a comprehensive framework for ethical decision making in software engineering. It ~~allow~~ outlines eight principles:

1) Public: Software engineers shall act consistently with the public interest.

2) Client & Employer: Act in a manner that is in the best interests of their client and employer consistent with the public interest.

3) Product: Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

4) Judgement: Software engineers shall maintain integrity and independence in their profession consistent with the public.

5) Management: Software engineering managers and leaders shall subscribe to and ~~leaders~~ promote an ethical approach to the management of software development.

6) Profession

Scanned with CamScanner

7. Colleagues

8. Self: Software engineers shall participate in lifelong learning regarding the principles.

The code of ethics provides a valuable resonance for software engineers to navigate ethical dilemmas and make responsibility.

Solution of Ques 05:

Certainly, here are five functional and five non functional requirements for an airport reservation system, along with their contributions:

Functional Requirements:

1. User ~~Restr~~ registration and login
2. Flight Search and Booking
3. Payment Processing
4. Reservation Management
5. Notifications and Alerts

Scanned with CamScanner

Non Functional Requirements:

1. Performance
2. Scalability
3. Security
4. Availability
5. Usability

How these Requirements contribute:

1. Performance
2. Usability
3. Security
4. System maintenance.

Scanned with CamScanner

Solution of ques-6:

The V-Model is a sequential software development lifecycle model that emphasizes a strong link between each phase of development and its corresponding testing phase. It's characterized by a "V" shape, visually representing this relationship.

Left side (Verification phase): The side focuses on ensuring the system is built correctly.

- Required Analysis
- System Design
- Architectural Design
- Module Design

Right Side (Validation phase): This side focuses on ensuring the correct system is built.

i) Coding: Corresponding to unit testing verifies the functionality of individual units of code.

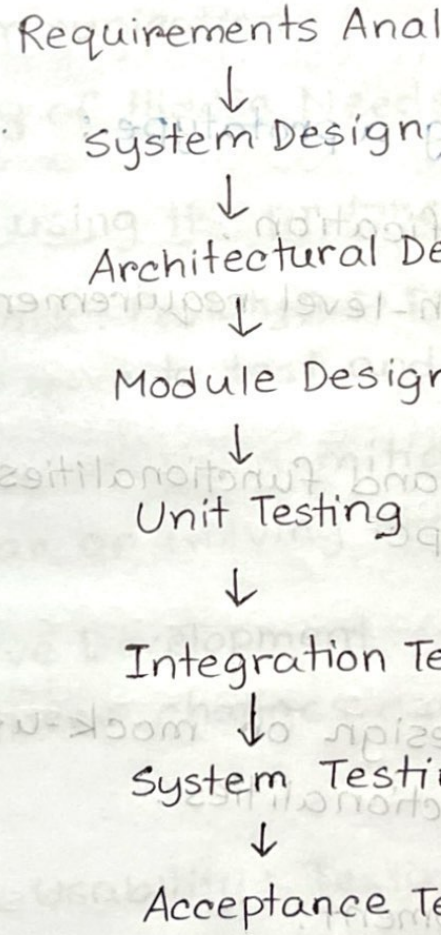
ii) Module Integration: Corresponds to Integration Testing.

Scanned with CamScanner

iii) System Integration : Corresponds to system testing.

iv) System Validation : Corresponds to acceptance Testing.

Visual representation:



Scanned with CamScanner

Solution of ques 7:

Prototype development is an iterative process in software engineering where a preliminary version of a software application is created to understand user requirements, test concepts and explore design solutions.

Key stages in creating a prototype:

1. Requirement Identification:
   - gather initial, high-level requirements from stakeholders
   - Focus on features and functionalities critical for creating the prototype.

2. Prototype Design:
   - Develop a rough design or mock-up that emphasizes core functionalities.

3. Prototype Development:
   - Build the prototype, ensuring it demonstrates the primary features and provides a basis for user iteration.

Scanned with CamScanner

4. User Evaluation:

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

How prototype helps Refine Software requirements:

i) Early validation of Requirements

ii) Clear communication

iii) Discovery of Hidden Needs

Benefits of using the prototype Model:

1. User feedback: Prototypes provide a tangible product for user to test and entique.

2. Risk Production: Helps mitigate risks associate with unclear or evolving requirements.

3. R. Iterative Development: Support an iterative approach where changes can be made incremen-tally.

4. Improved Usability: Testing the prototype with users ensures a user-friendly design.

Scanned with CamScanner

Solution of ques 08:

The process improvement cycle in software engine ring focuses on refining and optimizing software development processes to improve quality, efficienc and reliability.

key stages in the process improvement cycle:

1. Process Assessment: collect data on process performance using metrics like defect density cycle time.

2. Goal Setting: Define clear, measurable objectives for process improvement.

3. Process Redesign: Propose and plan changes to existing processes to address identified inefficie cies.

4, Implementation: Apply the proposed process changes incrementally.

5. Monitoring and Measurement: Collect & analyze process metrics to access the impact of chang

Commonly used process Metrics in Software Engineering :

process metrics provide quantitative data to monitor and assess software processes.

1. Productivity Metrics : Measure the output produced relative to the input effort.

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

3. Cycle Time : Measures the time taken to complete a specific process or task.

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

5. Customer Satisfaction Metrics : Measures user satisfaction with the delivered product or process.

How process Metrics help in Monitoring and improving software processes :

1. Data-Driven Decision Making.

Scanned with CamScanner

2. Early problem Detection

3. Performance Benchmarking

4. Encourging Accountability

5. Continuous Feedback Loop

Solution of Ques - 09:

The capability Maturity Model (CMM), developed by the software Engineering Institute (SEI), is a structured approach to evaluating and improvi software development processes.

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

The five levels of CMM and their impact:

1. Level 1: Initial (Upperedictable)

- Processes are ad hoc, dependent on individuals.

Scanned with CamScanner

2. Level 2: Repeatable (Basic Control)

-Basic project management practices are established.

3. Level 3: Defined (Standardized)

- Improves collaboration and consistency.

4. Level 4: Managed (Measured)

- Metrics are used to monitor and control processes

5. Level 5: Optimizing (continuous Improvement)

- Focus on innovation, defect prevention and adaptability.

How each level drives process and performance improvements:

1. Starting point:

2. Building Foundations.

3. Scaling efficiency

4. Achieving Precision

5. Sustaining Excellence

Scanned with CamScanner

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 : Standardized and optimized processes reduce waste and improve productivity.

iv) Customer Confidence : Reliable delivery boosts trust and strengthens client relationships.

v) Risk Mitigation : Structured practices at every level minimize uncertainties and project risks.

Scanned with CamScanner

~~Que~~:

Solulation of Ques 11:

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 and deliverables.

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 & provides inputs.

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

Scanned with CamScanner

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

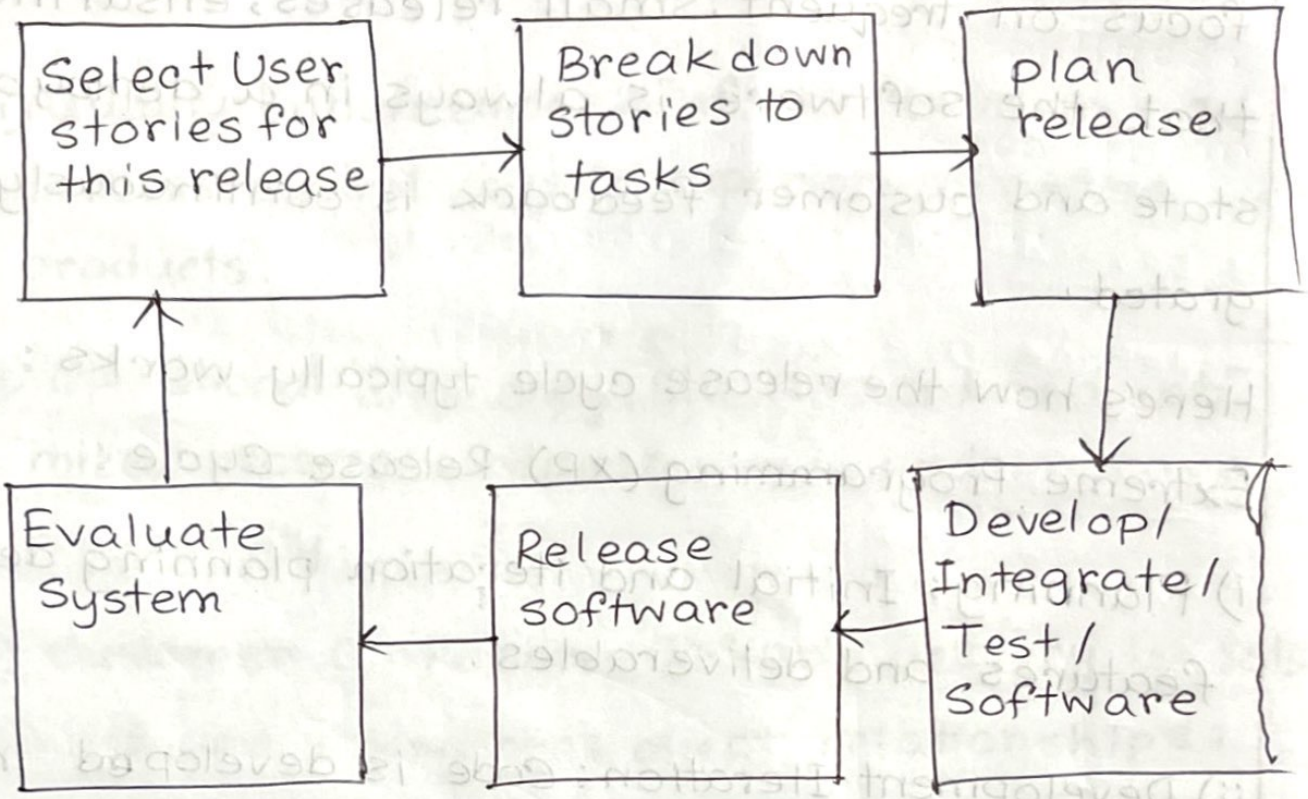


Diagram of release cycle of XP

Influential XP Programming Practices :

1. Pair Programming.
2. Test Driven Development (TDD).
3. Continuous Integration (CI)

4. Refractoring

5. Collective Code Ownership

6. Simple Design

7. Customer Involvement

Solution of ques 12:

Using the scenerio of an Entity Relationship Diagram (ERD) for the digital library management system, we'll break down the entities, their attributes and relatiops between them based on the requirements provided.

1. Entities and Attributes:

Book

Attributes:

- BookID (Primary key): Unique identifier for each book.
- Title: Title of the book
- Author: Author of the book
- ISBN: International Standard Book Number
- Genre: Genre of the book

Member

Attributes

- MemberID (primary key): Unique identifier for each member.
- Name: Name of the member.

Scanned with CamScanner

Contact Details : Contact information (phone number, email etc)

Address : Physical address of the member.

Borrowing Activity

Attributes :

Borrowing ID(Primary Key) : Unique identifier for each borrowing activity.

BookID(Foreign Key) : Refers to the borrowed book

Member ID(Foreign key) : Refers to the member who borrowed the book

Borrowed Date : The date the book was borrowed

ReturnDue Date : The date the book is due for return

Return Status : Whether the book has been returned or not.

Relationships :

Book Borrowing Activity (One to Many)

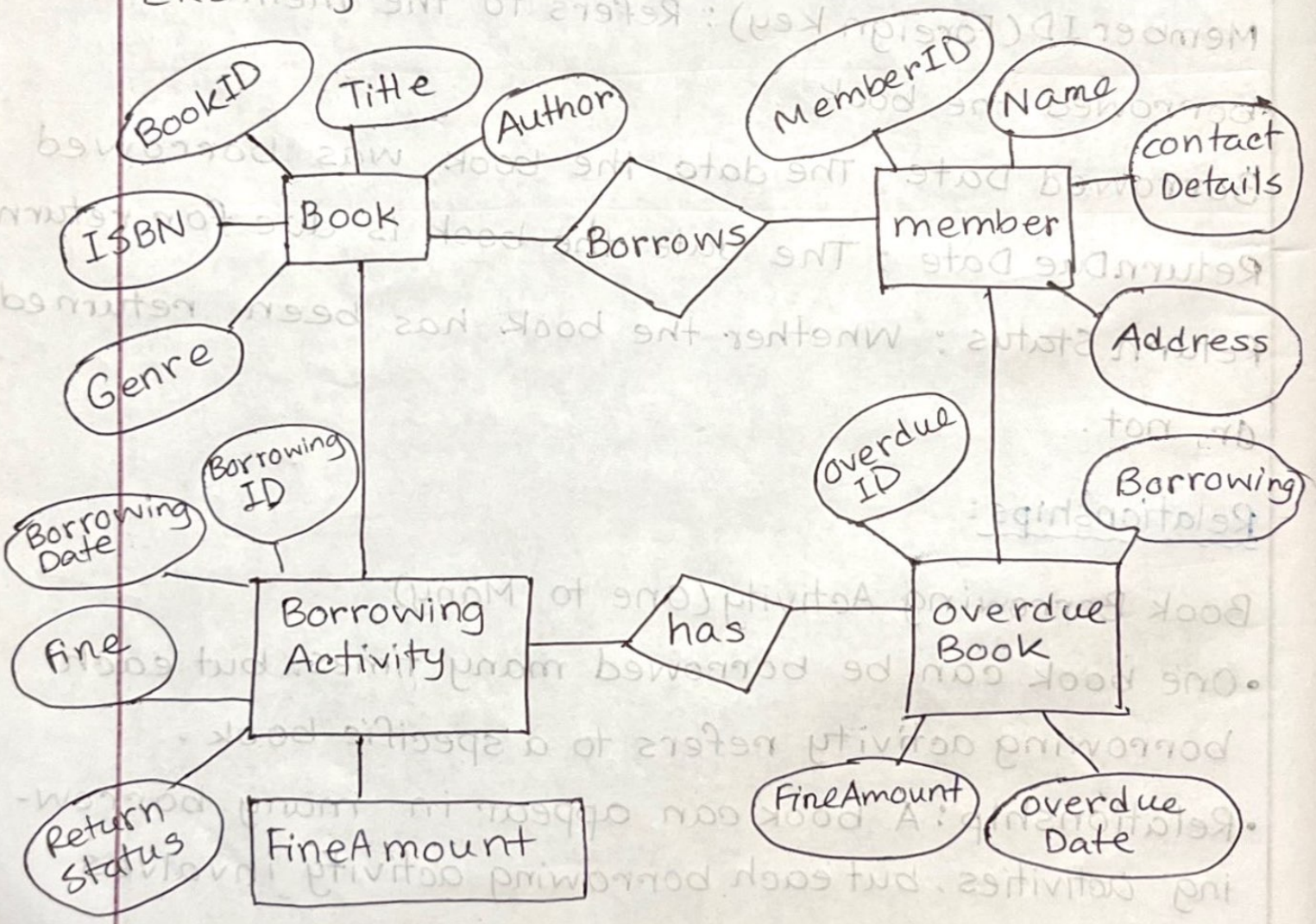
- One book can be borrowed many times, but each borrowing activity refers to a specific book.
- Relationship : A book can appear in many borrowing activities, but each borrowing activity invalves only one book.

Scanned with CamScanner

Member borrowing Activity (One to Many)

- One member can borrow multiple books, but each borrowing activity refers to only one member.
- Relationship: A member can have many borrowing records, but each borrowing activity invalves only one member.

ERD Diagram:



Solution of ques 13:

Testing: In software Engineering, Testing refers to the process of evaluating and verifying that a software application or system works as intended. It involves executing the software to find defects, ensure it meets specified, requirements and confirm it's overall functionality, performance and security.

Testing is a crucial part of the software Development Life Cycle (SDLC) and it typically occurs after the software has been developed and it before it is released to user.

Difference between Verification & Validation:

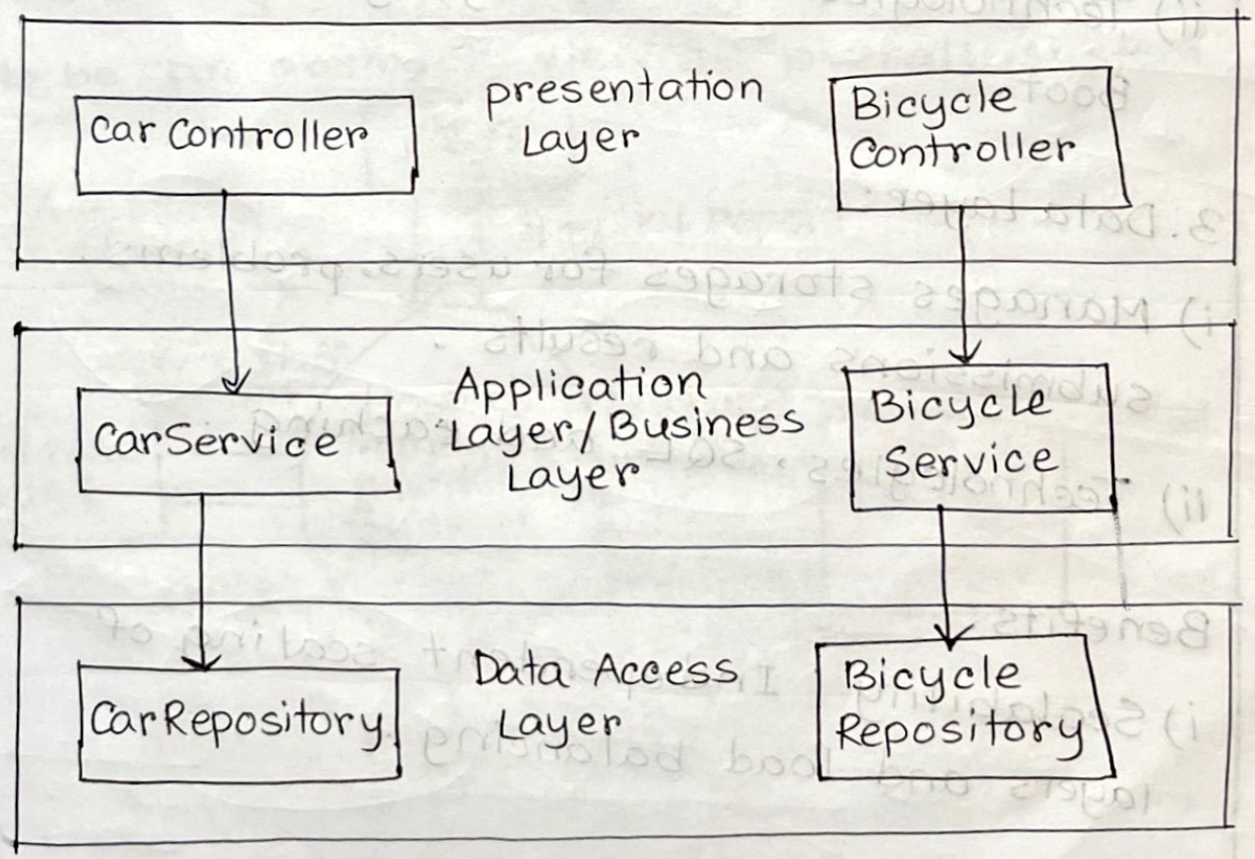
Verification	Validation
Verification refers to the set of activities that ensure software correctly implements the specific function.	Validation refers to the set of activities that ensure that the software that has been built is traceable to customer requirements.
It includes checking documents, designs, codes and programs	It includes testing & validating the actual product.

Verification is the static testing.	Validation is dynamic testing.
It doesn't include the execution of the code.	It includes the execution of the code.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
It comes before validations.	It comes after verifications.
verification is for prevention of errors	Validation is for detection of errors

Solution of 14 :

A Layered architecture for an online judge system would typically includes three primary layers: Presentation Layer (UI), Application Layer (Business Logic), and Data Access Layer (Database Interaction) with each layer having well-defined responsibilities and interactions only with the layer directly above or below it, promoting modularity and maintainability.

Layered Architecture diagram:



Layered Architecture for online Judge System:

1. Presentation Layer:

i) Handles users interactions.

ii) Technologies: React, Angular on mobile interfaces.

2. Application Layer:

i) Orchestrates requests between layer, manages authentication and routes submissions.

ii) Technologies: Node Js, Django or Spring Boot.

3. Data Layer:

i) Manages storages for users, problems, submissions and results.

ii) Technologies, SQL and eaching.

Benefits:

i) Scalability: Independent scaling of layers and load balancing.

Scanned with CamScanner

ii) Maintainability: Modular design for easy updates and debugging.

iii) Performance: Optimized request handling, secure sandboxing, efficient data retrieval.

Solution of question 15:

Context Level (0 Level) DFD of Hospital Management System: The 0 Level DFD for hospital management system depicts the overview of whole hospital management system. It is supposed to be an abstract view of overall system.

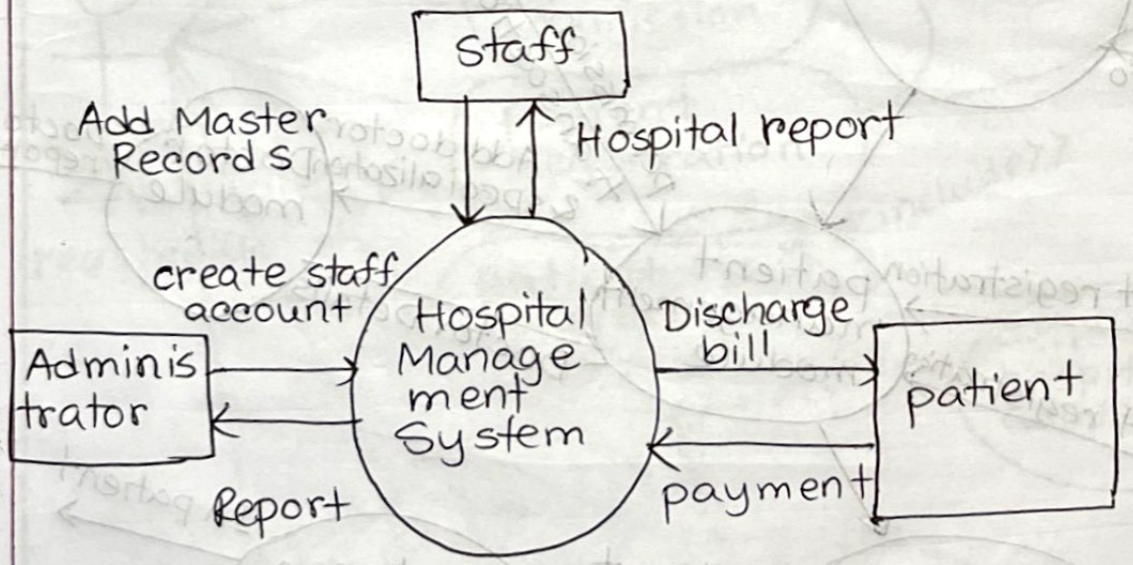
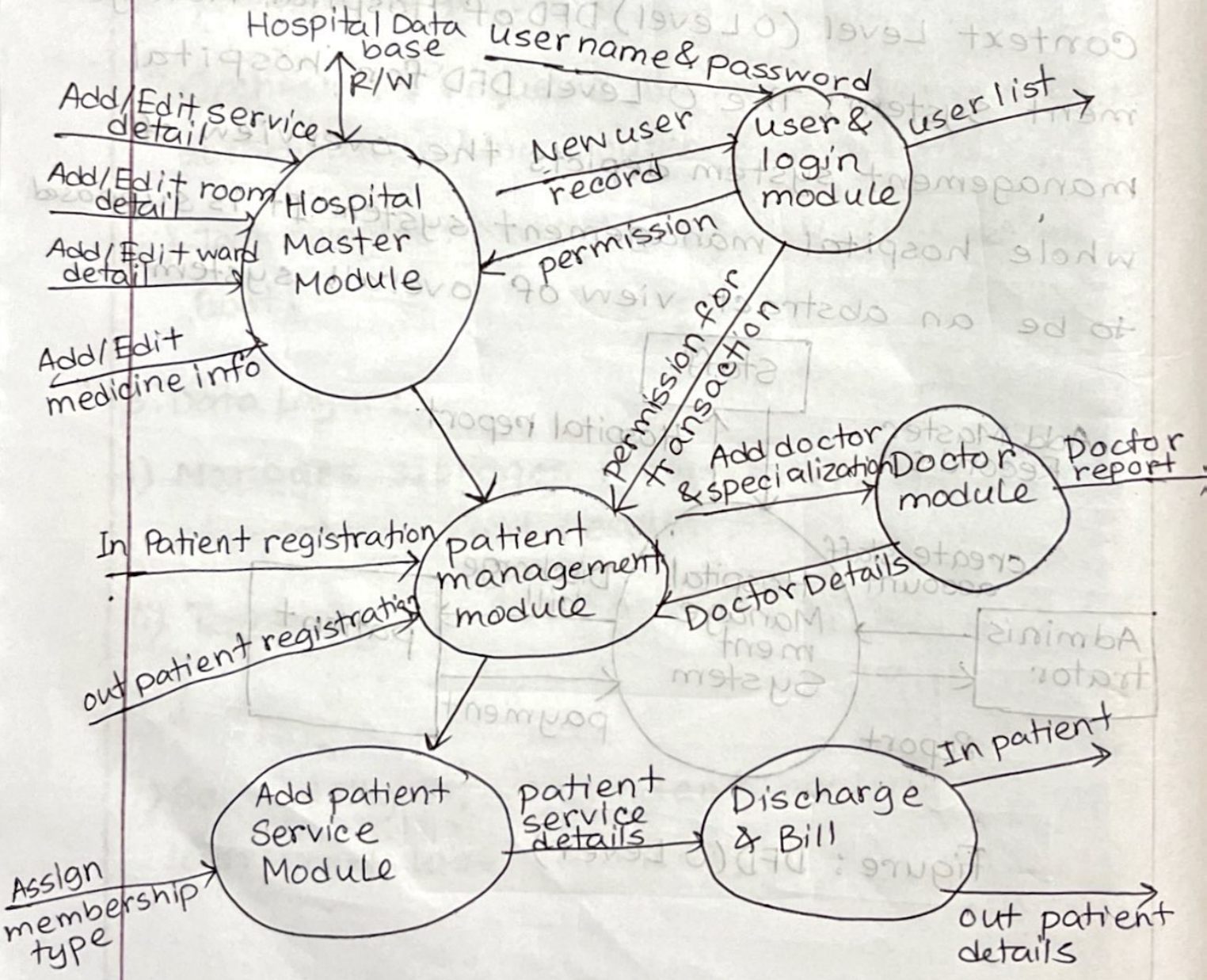


Figure: DFD (0 Level)

First Level Data Flow Diagram (Level 1 DFD) of 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.

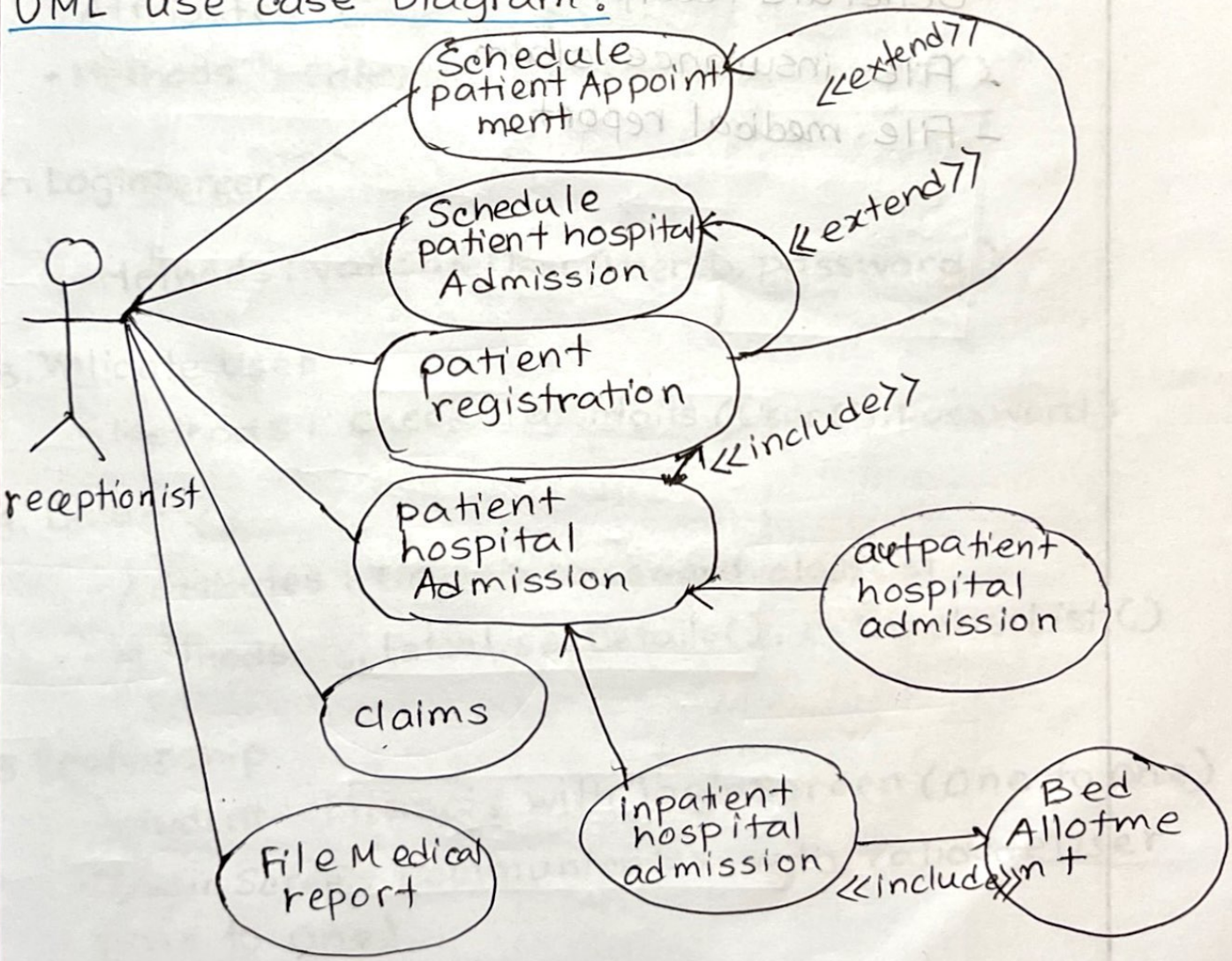


The important process to be carried out are :

1. User & login
2. Hospital master
3. Patient registration
4. Doctor module
5. Add patient Service
6. Discharge billing

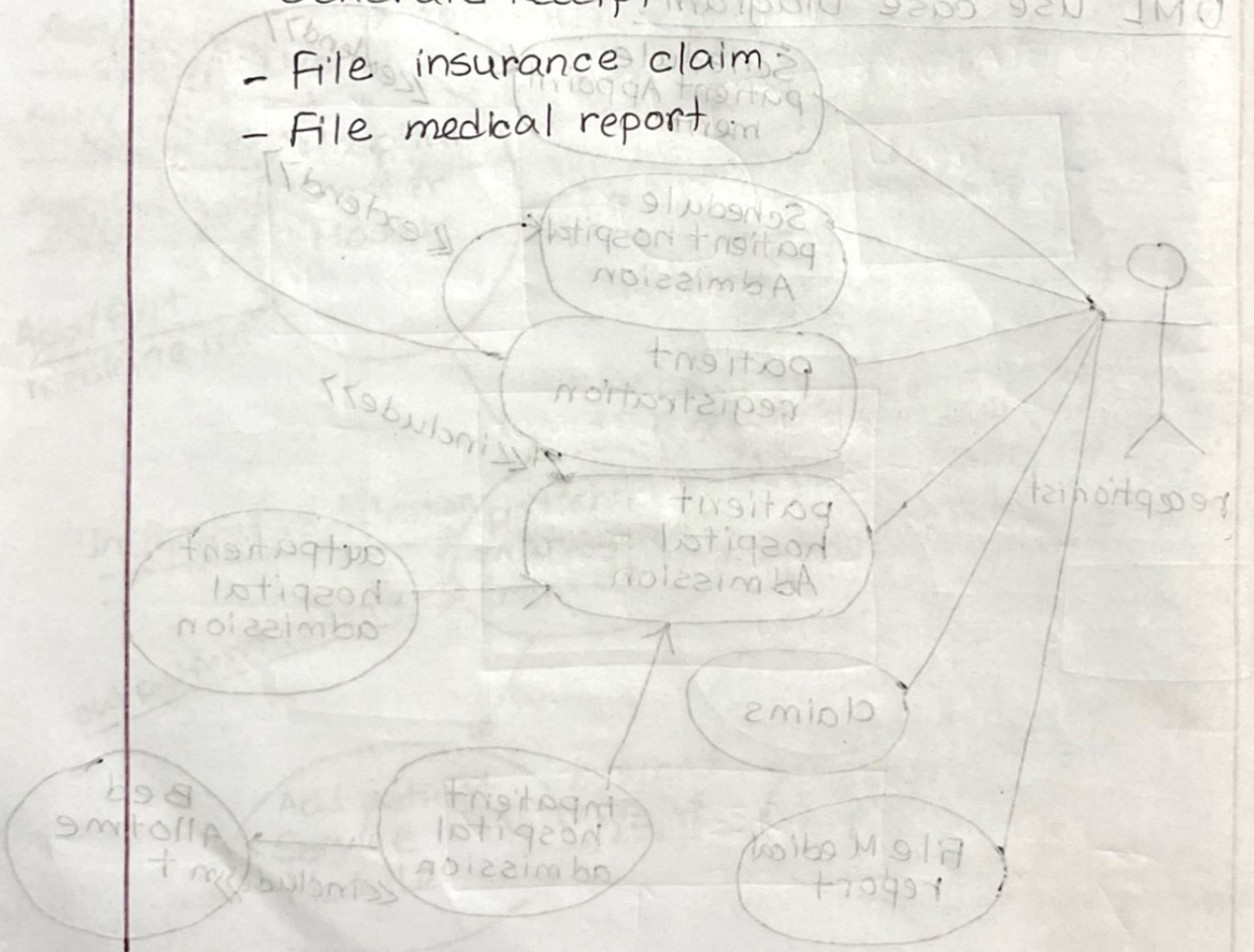
Hospital receiption

UML use case Diagram :



In this diagram :

- Actor : Receptionist
- Use cases :
  - Schedule Appointment
  - Admit Patient
  - Collect Patient Information
  - Allocate Bed
  - Receive Payment
  - Generate Receipt
  - File insurance claim
  - File medical report



Solution of Ques 16 :-

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

Identified classes from the sequence Diagram :

1. Student
   - Attributes : User ID, password
   - Methods : clickloginButton(), Viewlassisf()

2. LoginScreen
   - Methods : validateUser(UserID, password)

3. Validate User
   - Methods : checkCredentails(UserID, password)

4. Database
   - Attributes : UserID, password, classList
   - Methods : FetchUserDetails(), returnclassList()

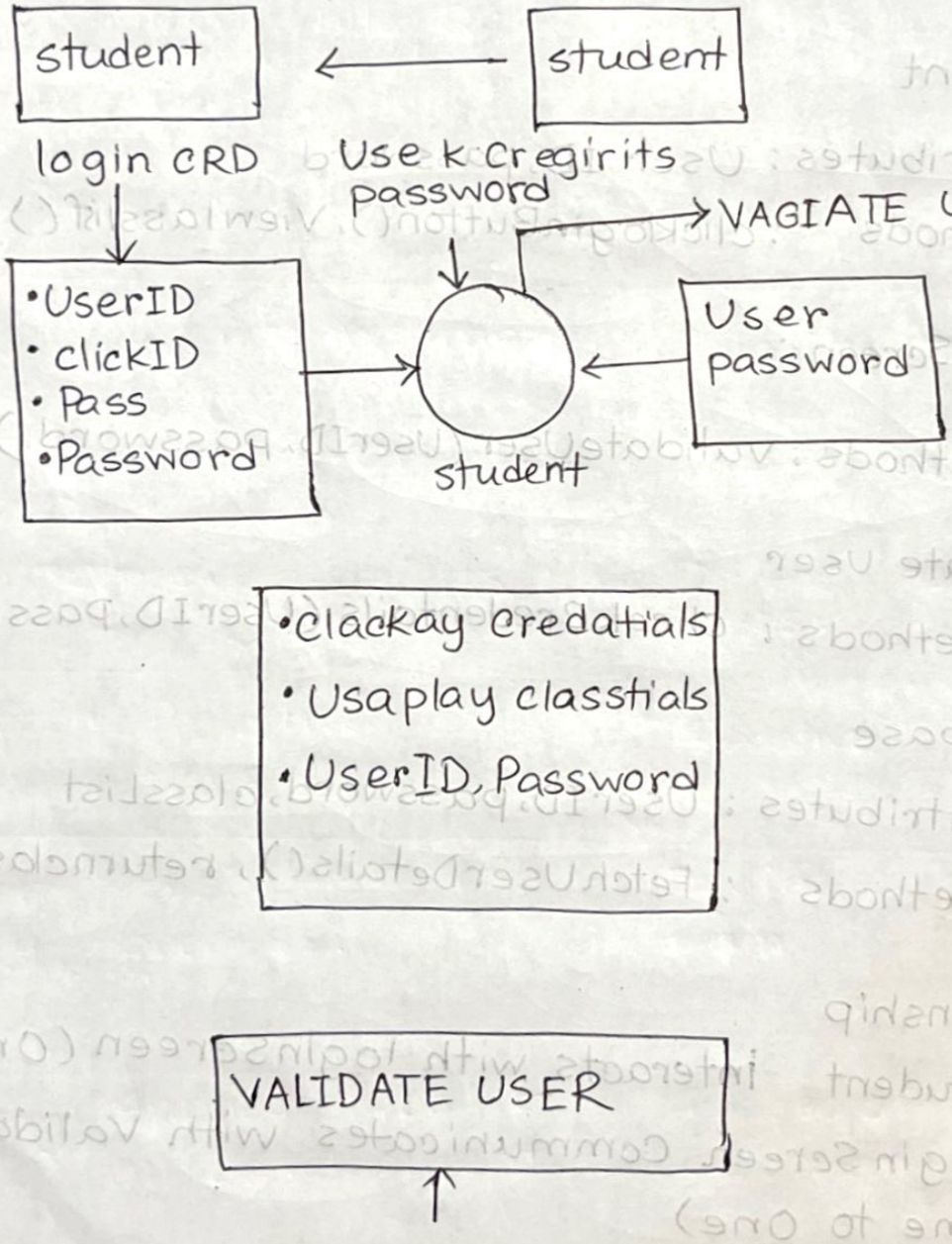
5. Relationship
   - student interacts with loginScreen (One to One)
   - LoginScreen Communicates with ValidateUser (One to One)

Scanned with CamScanner

-ValidateUser fetches data from Database (one to one)

- Database stores user details and class lists

Now, let me generate the UML class Diagram based on this structure:



Here, is the UML class Diagram for the student login system based on the provided sequence diagram. It represents key classes such as student, loginScreen, ValidateUser and Database along with their attributes, methods and relationships.

Solution of Ques 17:

Quality Assurance (QA) vs Quality Control (QC) – Explanation, Differences and Impediments:

Explanatory Discription:

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

Quality Assurance (QA) is a proactive, process-focused approach that ensures quality is built into the development process. QA aims to prevent defects by establishing standard processes, guidelines and best practices.

Scanned with CamScanner

Quality control is a reactive, product-focused approach that involves inspecting and testing the final product to identify and correct defets before release.

In short, QA ensures processes are designed for quality, while QC verifies the quality of the final product.

Difference between QA and QC:

Aspect	Quality Assurance	Quality control
Focus	Process-Oriented	Product-Oriented
Objective	Prevent-defects	Defect and correct defets.
Approach	Proactive	Reactive
Respon sibility	Everyone in the process.	Dedicated testing team
Timing	Applied through-out development.	Applied after development.

Scanned with CamScanner

Impediments to QA and QC:

For QA:

i) Lack of standardized processes.

ii) Poor management support.

iii) Resistance to change.

iv) Limited Resources.

v) Time constraints.

For QC:

1. Late Defect Detection
2. Inadequate Testing
3. Time pressure
4. Dependence on QA
5. Limited automation

QA and QC are complementary but distinct. QA establishes processes to ensure quality from the start, while QC verifies the final product meets requirements. To achieve high-quality outcomes, organizations must implement both effectively and address their respective impediments.

Scanned with CamScanner

Solution of Ques 18:

Role of Quality Assurance (QA) in 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. It focuses on preventing defects, improving processes and ensuring that the final product meets customer expectations.

QA as a discipline was introduced after world war II. where weapon testing was done to ensure reliability before actual use. Similarly, in software development, QA ensures that the product is tested and validated before deployment to avoid failures.

QA Role at each phase of SDLC

SDLC phase	Role of Quality Assurance (QA)
1. Requirement Aanalysis	- ensure clarity, completenss, and feasibility of requirements. - Conduct requirement reviews to prevent ambiguilty. - Define acceptance criteria and quality standards early.
2. Planning	- Identify risks and mitigation strategies. - Define QA objectives, scope and test strategy. - Plan resources, timelines and test environments.
3. Design	- Review system architecture and design for quality aspects like security, scalability and performance. - 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 use experiences.

Solution of ques 19:

Rapid Application Development (RAD) Model in Software Engineering:

The Rapid Application (RAD) model is an iterative and adaptive software development methodology that focuses on quick prototyping and fast feedback loops instead of extensive planning and documentation. It was introduced in the 1990s by James Martin as an alter-

Scanned with CamScanner

native to traditional models like the waterfall model.

Key phases of the RAD model:

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

Principle of the RAD Model:

i) Prototyping Over planning
ii) Active User Involvement
iii) Iteractive and Incremental Development.
iv) Component-Based Development.
v) Quick Delivery

Advantanges of the RAD model:

i) Faster Delivery
ii) Higher User satisfaction
iii) Flexibility
iv) Reduced Risk
v) Reasable components.

Scanned with CamScanner

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 prototyping, user feedback, and reusability. It is ideal for projects that require fast iterations and envolving requirements, ensuring that the final product is not only delivered quickly but also meets high quality standards and user satisfaction.

Scanned with CamScanner

Solution of ques : 20

White box testing ensures that all possible brances of a program- are tested by covering decision points such as if, else and loops. Below, we create a test cases table for the given java code and then implement a Junit test class in Java.

Test case 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 does not 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 does not execute	-3	2	"" (No out Put)

JUnit Test Class in Java

```
import static org.junit.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

Solution of ques 21:

JUnit 4 Black Box Unit Testing Approach:

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

To test exception handling, setup function & timeout rule, we develop JUnit 4 test cases for the following production code.

production code (calculator.java)

```
public class calculator{
    private int memory;
public Calculator(){this.memory=0;}
public int divide (int a, int b)
{
    if(b==0)
        throw new ArithmeticException("cannot
        divide by zero");
    return a/b;
}
```

Scanned with CamScanner

```
public double sqrt (double number)
{
    if (number<0)
        throw new IllegalArgumentException
        ("Negative number");

    try
    {
        thread.sleep(500);
    }
    catch (Interrupted Exception e) { }

}
public void storeInMemory (int value)
  {
    this.memory = value;
  }
public int getMemory ()
    {
      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.rule.Timeout;

public class CalculatorTest {

    private Calculator calculator;
    public Timeout globalTimeout = Timeout.millis
                                   (1000);

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

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

```
@Test
    public void testsqrtComputationTime()
    {
        double result = calculation.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