Iffate Jahan Nilima
ID: IT-21063

Question-1

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 seewely so that I can accep my accound.
- 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 team can prioriti these user stories during a sprint planning meeting. considering value to the customer and technical feasibility.

III. Illustrate how these tasks will be tracked using a scrum board. Use proper term like "To do", "In progress" and "Done"

Answer:

I

For the e-commerce application, the product Backlog is created by breaking the user stories into actionable tasks.

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

Task 2: Implement client-side validation for login

Task 3: Set-up backend authentication API

Task 4: Implement Password Hashing and secure storage

Task 5: Implement session management

Task 6: Write unit tests for the login functionality

Task 7: Perform security testing

Task 8: Implement logout functionality.

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.

Task 4: Write unit test for products search functionality.

Task 5: Optimize search query

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

- Value to the customer:

User story 1 (log in): High priority. Secure log in is crucial for user trust and account access.

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

"III"

The scrum board will track the progress of tasks during the sprint. The columns are "To do", "In progress", and "done"

To Do	In progress	Done
user story 1 (login)	user story 1 (login)	user story (login)
- Design the login page UI	- Implement frontend validation	- Write unit test for login
- Conduct security testing	- Write unit tests for login	
User story 2 (search)		User story 2 (search)
- Design search UI, with category dropdown		Write unit tests for search
- Create database schema		
- Implement search API		
- Integrade frontend with search API		
Optimize search query		

Question-2

A software development team is about to start a project for a new innovative product. The project has a several high risk components due to its novelty The client is open to interactive changes, but the team must ensure that the software evolves in a manageable, cost effective way.

now discuss, how the spiral, Agaile and extreme methodolgies address risk management and adaptability. which methodology would be the most suitable for a project with the significant risk and evolving requirement and why?

Answer: Risk Management and adaptability in spiral Agile and extreme methodologies.

- Special methodology: focuses on risk management through frequent cycles of planning, risk analysis and refinement. It is adaptable due to iterative revisions but it can be more formal and document-heavy

- Agile Methodology: Uses short iteration and constant client feedback, allowing for early identification of risks and frequent adjustments based on evolving requirements.

- Extreme Programming (XP): Emphasizes continu integration and test-driven development, ensuring high-quality code and minimizing defects.

•characteristics:

•Predictability: Highly predictable.

• customer collaboration: Minimal customer involvement after the requirement.

•Risk management: Limited risk management.

2. Agile Methodology

overview: Agile is an iterative and incremental approach.

•characteristics:

• Predictability: Less predictable in term of timeline and scope.

• Customer collaboration: High customer involvement with regular feedback.

•Risk management: focuses on managing risk through continuous testing and feedback.

3. Extreme programming (xp)

Overview: xp is an agile methodology that emphasizes technical excellence continuous integration collaboration and frequent releases.

characteristic:

•Predictability: Less predictable due to frequent changes

• Customer collaboration: very high customer involvement.

Risk management: Strong emphasis on reducing technical risk through practise like ~~test~~ driven development and pair programming.

4. Spiral methodology:

Overview: Spiral is a risk-driven model that combines iterative development with regular risk assessment.

Characteristic:

- Predictability: moderate predictability.
- Customer collaboration: Regular feedback and interaction allow for high collaboration with the customer.
- Risk management: Emphasizes proactive risk management by identifying and addressing risk at every phase of cycle.

Question-3

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

- compare and constract the waterfall, Agile, extreme and spiral development models. Among project A and B which methodology is best?

Answer:

Project A

- Best Methiodology : waterfall

- Why: Since Project A has well-defined requirements and a strict deadline, Waterfall is the most suitable methodology.

- How it addresses the project:

- Predictability: Waterfall's clear, linear progression ensures that the project stays on schedule with milestone

- Customer collaboration: Minimal, as requirement

Project B

Best methodology: Agile or spiral

- Why: Project B requires continuous customer feedback and has evolving requirements. both Agile and spiral suitable.

- How it addressed the project:

- Predictability: Agile is less predictable.

- Customer collaboration: Agile emphasizes close collaboration with customer.

- Risk management: Risks are addressed continuously in Agile by identifying and resolving issues during each sprint. This iterative process minimizes the risks of significant issues

Summary of Methodology suitability:

- Project A: Waterfall is the most suitable methodology due to its predictability, structure and ability.

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

Question -4:

Explain the principles of software engineering ethics, highlighting the issues related to professional responsibility. Discuss how the ACM / IEEE code of ethics guides ethical decision-making in software engineering practice.

Answer

Principle of software Engineering ethics.

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

2. Professional competence: Engineers should work within their areas of expertise

3. Honesty and integrity: Transparency, honesty, and integrity in communication, reporting progress and managing expectations are essential.

4. Confidentiality: Engineers must respect the confidentiality of client, employee and users information.

ACM / IEEE code of Ethics and Ethical decision-making:

1. Public interest: Engineers are encouraged to publics best interest, ensuring their work is safe and secure.

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

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

Question-5:

Given the story of the Airport Reservation System. Identify at least five functional requirements for the system. In your answer, explain how each requirement contribute to overall performance, usability and security of the system.

Answer:

ARS is designed to manage flight bookings, passenger reservation, and other related service in a airport. Below are five functional

and five non-functional requirement for such a system, explaining how each contribute to the system's Performance usability and security.

Functional Requirements:

1. User Registration and login
2. Flight search and Booking
3. Payment processing
4. Flight reservation management
5. Notification System.

Non-functional Requirements:

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

Question-6

Illustrate and explain the V-model of testing phase in a plan-driven software process, detailing the relationships between development activities and corresponding testing activities.

Answer:

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

Key Relationship:

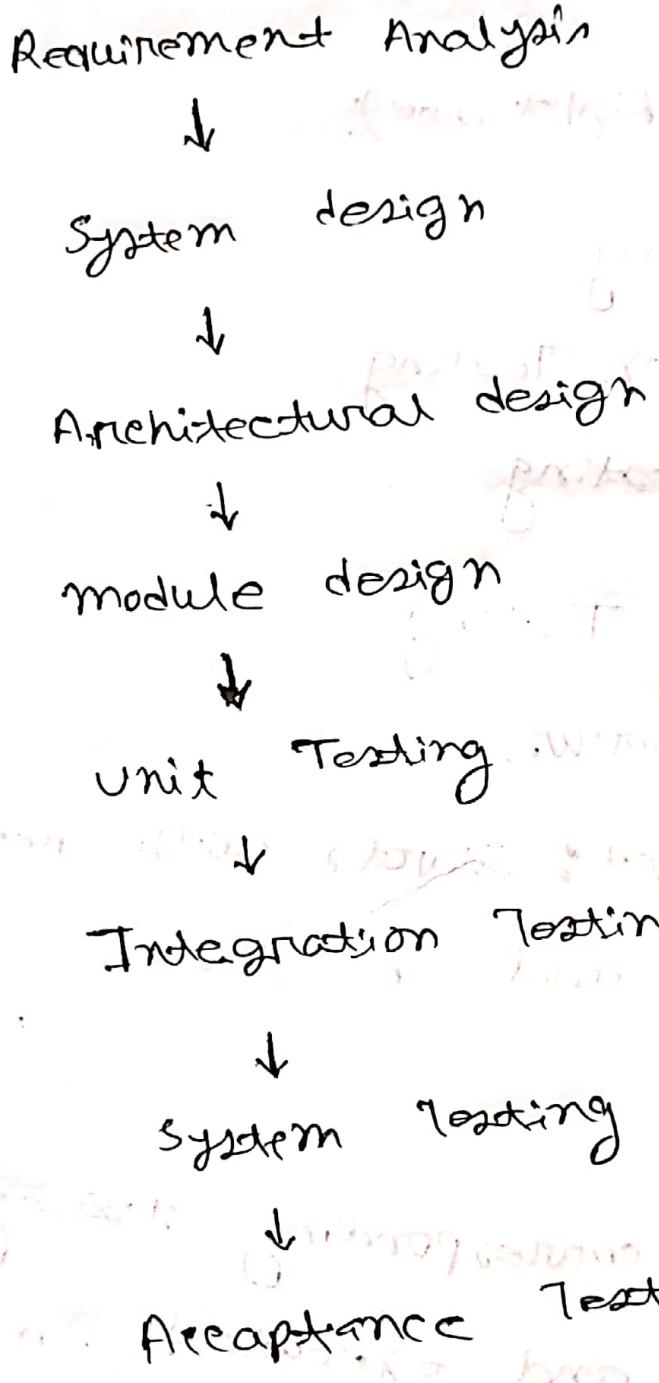
- Verification (Left side):
- Requirement Analysis
- System design
- Architectural design

- Module design
- Validation (Right side):
- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

How it works:

1. Development: Starts with requirements analysis and progresses through design phase
2. Testing: corresponding testing phases are planned and executed in parallel with development.
3. Verification: Ensures the product is built correctly.

• Visual representation:



Question -7

Answer: Prototype Development is a software development model where a basic version of the system, known as a prototype is created to gather feedback and define requirements.

key stages of prototype development:

1. Requirement gathering: Basic, high-level requirement are called to build the prototype.

2. Prototype development: A simple, working model is created with key feature for initial testing.

3. Refinement: The prototype is revised based on feedback, adding feature and improving functionality.

4. User feedback: Users interact with the prototype and provide feedback on functionality and design.

*Benefits of the prototyping Model:

1. User feedback.
2. Risk Reduction
3. Iterative.
4. Better requirement claraficaction.

Question -8

Answer:

The process improvement cycle focuses on continuously enhancing software development process to improve quality, efficiency and performance. It typically follows PDCA.

1. Plan: Identify areas for development and define goals.

2. DO: Implement changes or improvement.

3. Check: Measure the impact of the changes using process metrics.

4. Act: If the changes were successful standardize them; if not, refine

*key stages in process improvement:

1. Assessment.
2. Analysis.
3. Implementation.
4. Evaluation.
5. Standardization.

How metrics help in monitoring and improving processes.

- Defect Density helps identify quality issues early
- Cycle Time and Velocity help optim process efficiency.

Question - 9

Answer: The software Engineering institute capability Maturity Model is a frame-work used to asses and improve software development process within an organization

Five levels of the SEI CMM:

1. Level-1 - initial (Ad hoc):

At this level processes are often chaotic and unorganized.

2. Level 2 - managed (Repeatable):

At this level basic project management process are established.

3. Level 3- defined (proactive):

Processes are well-defined and standardized across the organization. There is a focus on continuous process improvement, with practise for quality assurance, design and development.

4. Level 4- Quantatively managed:

The organization begins using quantitative data and metrics to monitor and control processes, and improve quality.

5. Level 5- optimizing (continuous improvement)

The organization focuses on continuous improvement by optimizing process based on data and feedback.

How each level contributes to improving software development.

- Level 1 (initial)
- Level 2 (managed)
- Level 3 (defined)
- Level 4 (Quantatively managed)
- Level 5 (optimizing)

Question - 10

Answer:

Core Principle of Agile software Development

1. Customer collaboration over contract Negotiation:

Agile emphasizes working closely with customers throughout the development process to ensure the product meets their needs

2. Responding to change over following a plan: Agile values flexibility, allowing the project to adapt to changes in requirements or market conditions.

3. Delivering working software frequently: Agile encourages frequent delivery of small, functional increments of the product. 1-4 weeks.

Benefits of Agile:

1. Flexibility: Agile allows teams to adapt to changing requirements and feedback quickly.

2. Faster delivery: Regular, incremental releases keep the project moving forward and allow early testing.

3. customer satisfaction: continuous collabor with the customer leads to better alignment with their needs.

2. Requires experienced

3. Overhead in coordination

4. Not Ideal for all projects.

Question-11

Answer:

The extreme programming (XP) release cycle focuses on frequent, small releases, ensuring that the software is always in a deployable state.

deliverables.

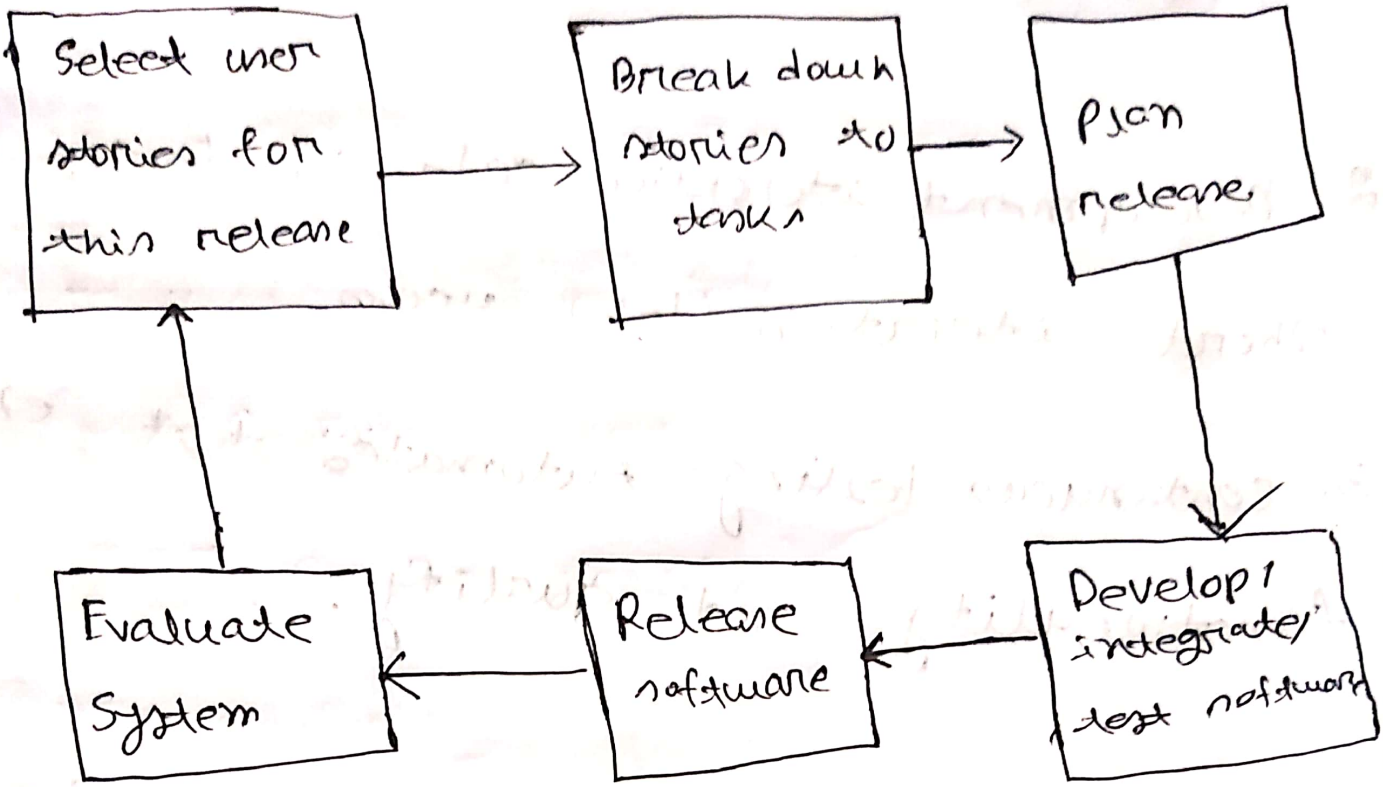
2. Development iteration: code is developed in short iteration (1-2) weeks

3. Continuous Testing: Automated test ensure functionality and quality.

4. Customer feedback: customer reviews each iteration and provides input.

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

6. Repeat: The process repeats in cycle, refining the software with each iteration.



Influential xp programming practices

1. Pair programming
2. Test Driven Development
3. continuous Integration
4. Refactoring
5. Collective code ownership
6. Simple Design
7. Customer involvement.

Question 12

Answer

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 all book.

Members

- Attributes

- Member ID (Primary key): Unique identifier for each number.
- Name : Name of the member
- Contact details : contact info
- Address : Physical addres of the member

- BorrowingID (Primary key)
- BookID (Foreign key)

memberID (Foreign key)

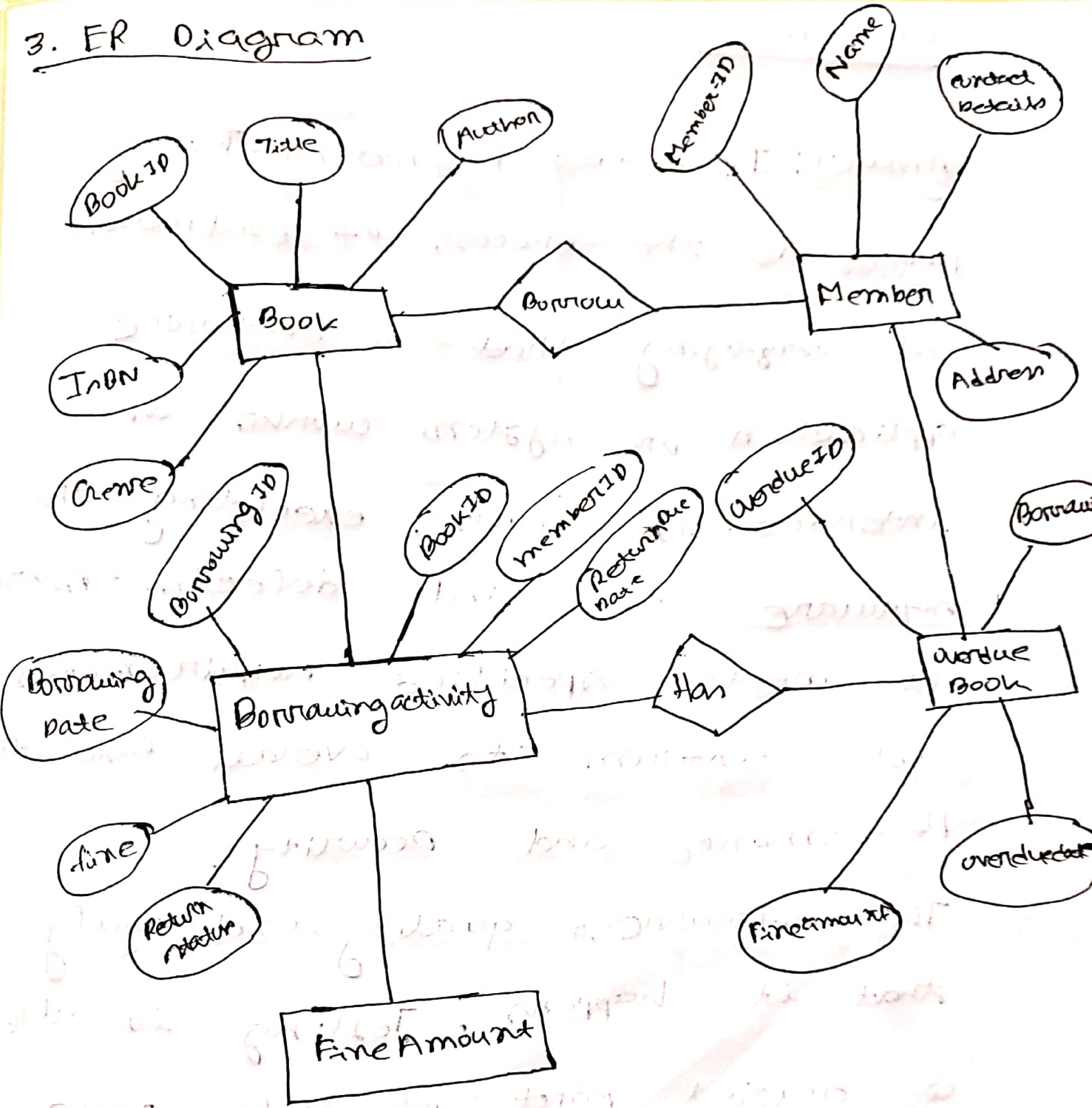
- Borrowed Date:
- ReturnDate
- Return status

Overdue Book

-Attribute

- OverdueID (Primary key)
- Borrowing activity

3. ER Diagram


Book ID
Title
Author
Book
ISBN
Genre
Borrow
Member
Member ID
Name
Contact details
Address
Borrowing ID
Book ID
Member ID
Return date
Overdue ID
Borrowing date
Borrowing activity
Has
Overdue Book
Fine
Return status
Fine Amount
Fine amount

Question-13

Answer: 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 its overall functionality, Performance and security.

The software's quality and verify that it happens. Testing is also a crucial part of life cycle (SDLC).

Difference between Verification and 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
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
The goal of verification is application and software architecture	The goal of validation is an actual product
It comes before validation	It comes after verification
Verification is for prevention of errors	Validation is for detection of errors
Verification finds about 50-60% of the defects	validation finds about 20-30% of the defects

Question 14

Answer:

Layered Architecture model for an Online Judge System

1. Presentation Layer (UI)

- The front-end where users submit code, view results and check rankings
- Example: A website or mobile app using HTML, CSS and javascript.

2. Application Layer

- Handle user request, authentication
- Backend framework like Django, Node.js

3. Business logic Layer (Judge system)

- Runs the submitted code in a secure environment.

4. Data layer (Database)

- Store user submissions, problem, test cases
- Example: MySQL, Firebase etc

Benefits of this Architecture

- Scalability: can handle many users
- Maintainability: Each layer is independent making updates.
- Performance: uses caching and background processing to make the system fast and efficient.

Question 15

Answer: context level (0 level) DFD of Hospital Management system.

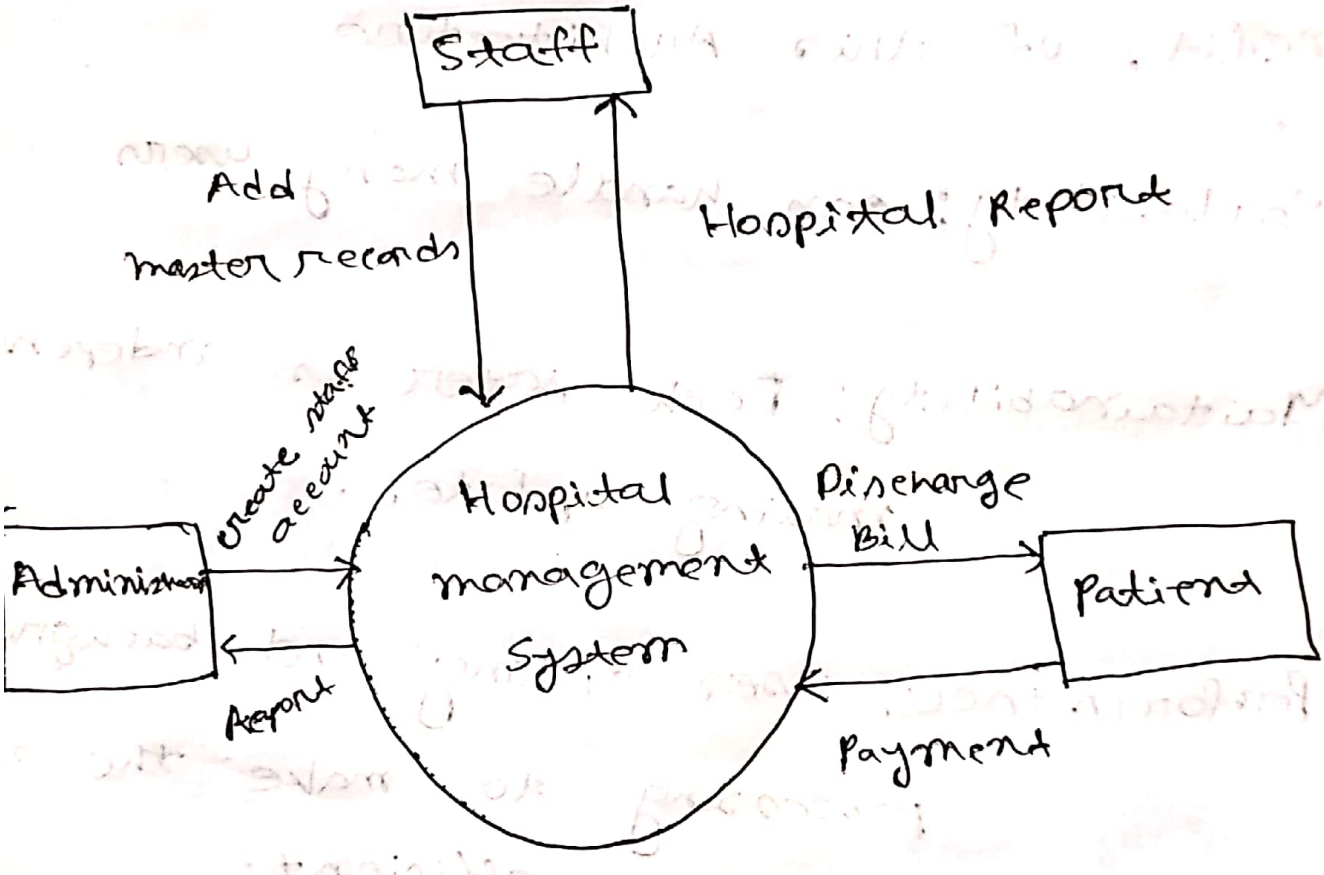
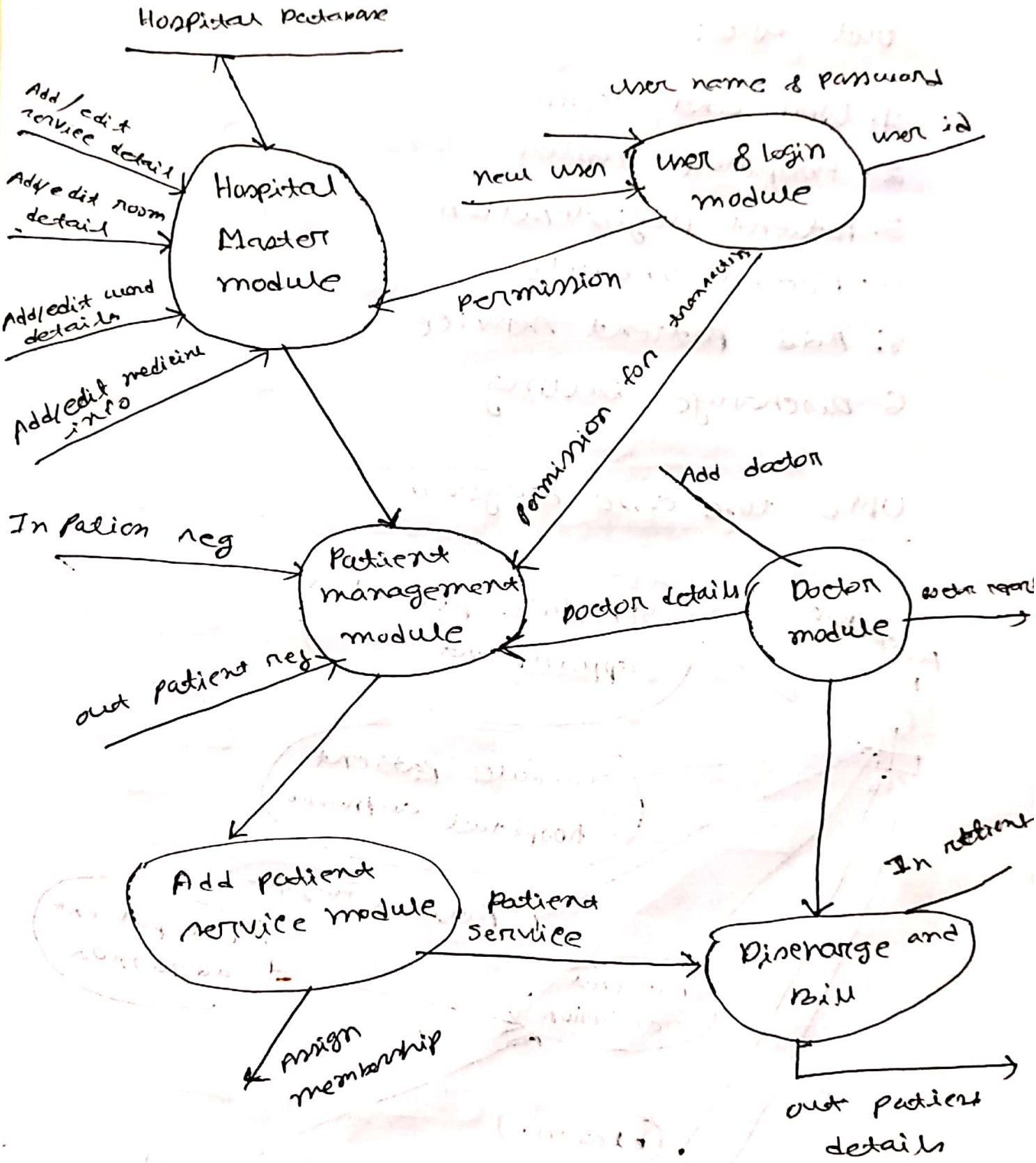


Figure: DFD (0 level)

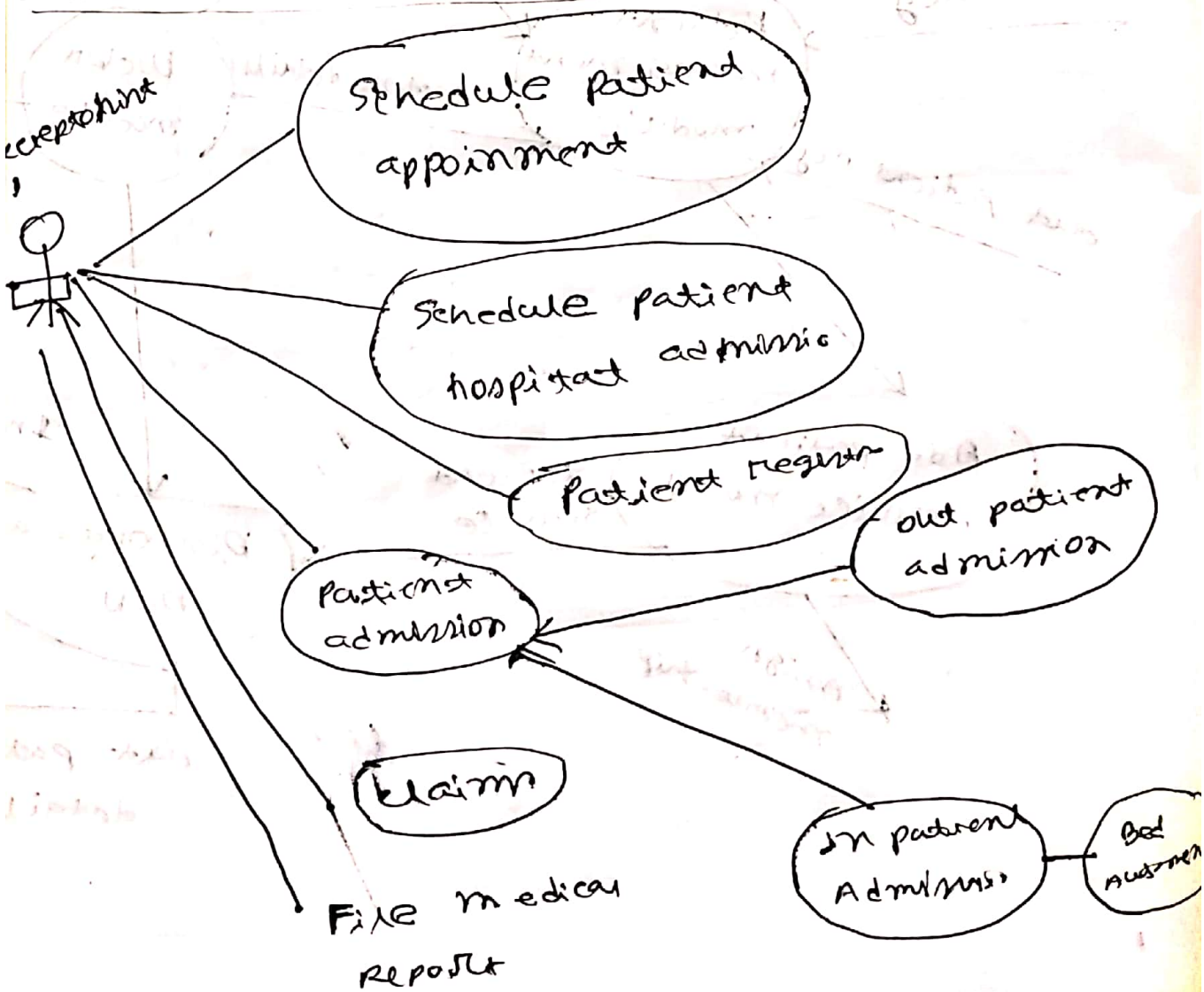
DFD (Level 1) of Hospital management system



The implemented process to be carried out are:

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

UML use case diagram:



Question 16

Answer: Based on the UML sequence Diagram 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(), viewclasslist(),

2. login Screen:
   - Methods : validateuser (user ID, password)

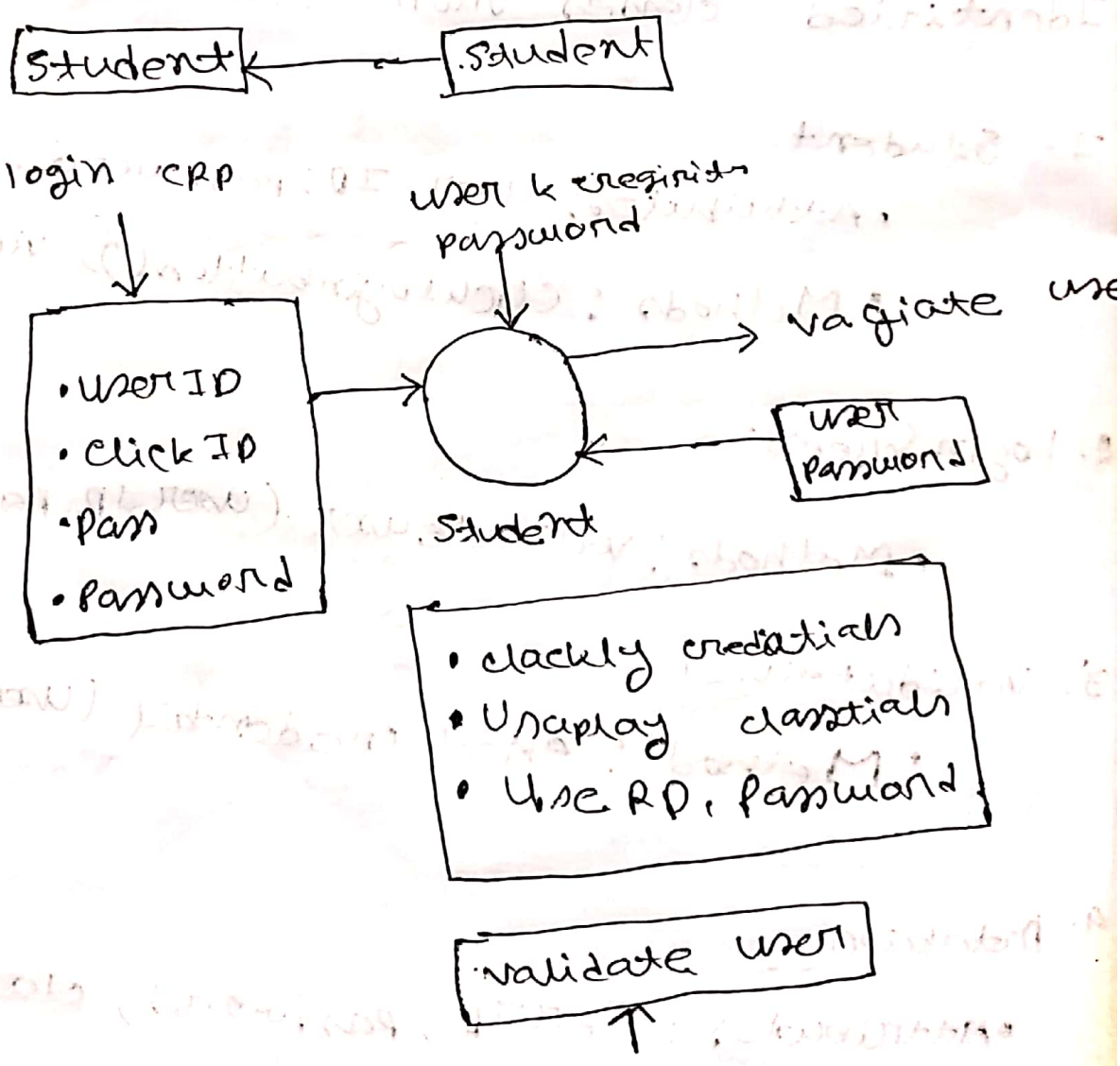
3. Validateuser
   - Method : check credential (user ID, passwor

4. Database
   - Attributes : user ID, password, classlist
   - Methods: Fetchuser(), returnclasslist()

Relationships:

- Student interacts with loginscreen (one to
- LoginScreen communicates with validateuser (one to one)

UML class diagram based on this stru



Question 17

Quality Assurance (QA) vs Quality control (QC)

explanation, Difference and impediments:

Explanatory Description:

QA and QC are both essential components of quality management, but they serve distinct purpose

- QA is proactive, process focused approach that ensures quality is built into the development process,

- QC is a reactive product focused approch that involves inspecting and testing the final product to identify and correct defect before release.

Difference between QA an QC

Aspect	Quality Assurance	Quality contra
Focus	Process-oriented	Product-oriented
Objective	Prevent- Defects	Detect and correct defect
Approach	Proactive	Reactive
Responsibility	Everyone in the process	Dedicated testi team
Timing	Applied throughout development	Applied afte development

Impediments to QA and QC

For QA

(i) Lack of standardized process

(ii) Poor management support

(iii) Resistance to change

(iv) Limited Resource

(v) Time constraints.

For 9c

1. Late detect
2. Inadequate Testing
3. Time pressure
4. Limited Automation

Question 18

Answer

Role of QA in SDLC: The goal of Quality Assurance is not just to find bugs but to ensure that quality is built into the software development process from the beginning. It focuses on preventing defects, improving process

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

QA Role at each phase of SDLC

SDLC Phase	Role of quality Assurance
1. Requirement Analysis	- Ensure clarity, completeness and feasibility of requirements - Define acceptance criteria, and standards early
2. Planning	- Identify risks and mitigation strategies. - Define QA objectives, scope and test strategy. - Plan resource, timelines and test environment
3. design	- Identify system architecture and do potential failure points early

Question 5

Answers:

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.

Key phases of RAD

- Business Modeling
- Data Modeling
- Process Modeling
- Application generating and testing

Principle of the RAD Model:

- prototyping over planning
- Active user Involvement.
- Iterative and incremental development.
- Quick Delivery.

- Higher user satisfaction
- flexibility
- Reduced Risk
- Reusable components

*How the RAD Model ensures faster Delivery while maintaining quality and user satisfaction.

- Speed through prototyping
- continuous user feedback
- iterative testing
- focus on reusable components.

Question 20:

Answer:

Production code (Java)

```
import java.util.scanner
public class numberprocesson{
 public static void main(string[] args){
      scanner c = new scanner (system.in)
      int x, y;
      x = c.nextInt( );
      y = c.next Int ( );
       process (x,y);
    c.close ( );
}
Public static void process (int x, int y){
  if (y == 0){
    system.out.println ("y is zero");
}
else if (x == 0){
        system.out.println("x' is zero");
}
```

```
else {
  for (int i=1; i<=x; ++){
    if (i%y==0){
      System.out.println(i);
    }
  }
}
```

Junit test

```
import java.util.ArrayList
import java.util.List
import org.junit.Before
import org.junit.Test;

public class NumberProcessorTest{

    private List<String> output
```

@Before

```
  Public void setup(){
      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");
}
  println("x is zero");
}
else {
  for (int i=1; i<=x; i++){
    if (i%y==0){
    println(return value of (i));
  }
}
}
```

④ test

```
public void testBZero(){
        output.clear();
        process(5, 0);
    assertEquals(List of("y is zero"), output);
}
```

(a) Test

```
public void Test B zero(){
    output.clear();
    process(0, 3);
    assertiveequals(list of("A in zero"), output
}
```

(a) Test

```
public void TestLoop DoesNotRun(){
    output.clear();
    process(0,2);
    assertiveEach(list of(), output);
}
```

Question 21

Answer:

Junit code demonstrate

- exception handling
- Set up function
- Timeout rule

Production code (calculator.java)

```
Public class calculator {
    public int add (int a, int b) {
        return a+b;
    }
public int divide (int a, int b){
    if (b==0){
    throw new Arithmetic exception ("b by zero");
}
return a/b;
```

(a) befor

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

(a) Rule

```
Public expectedexception exceptionrule = expected
expected.none();
```

(a) Test

```
Public void testDivisionbyzero(){
exceptionrule.expect (Arithmetic exceptio.cla
Exception.expection ("by zero");
calculator.divide (10,0);
}
```

(a) Test (timeout=1000)

```
public void testlongrunning operation(){
  calculator.longrunning.operation();
}
```

@Test

public void testAddition() {

assertEquals(15, calculation.add(10, 5));

Here @Before is declared with setup() function which is a build in function of Junit