Name: Khadija Arefin Meem
ID : IT21059

Question-01

Ans: I. 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 team 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 :
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 Log out 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 as 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 frontend 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 performan

Task 7 : Test the search 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 as 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 frontend 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 performanc

Task 7 : Test the search functionality.

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

Value to the customer:

User story1 (Login): High priority. secure log in is crucial for user trust and account access, enabling core functionalities like order, history, shared items and personalized recommendations.

User story2 (category search): High priority. Effective product discovery is essential for a successful ecommerce platform.

Technical feasibility:

User story1 (log in): Moderate complexity. Requires careful implementation of security measures and integration with authentication services.

User story2 (category search): Moderate to high complexity depending on the complexity of the product catalog.

III. The scrum board will track the progress of tasks during the sprint. The board consis of columns to visualize the workflow such 'To Do', 'In progress' and 'Done'.

Columns:

1. To Do: Tasks that have not yet started and are in the planning stage.

2. To 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.

Question 02:

Ans: Risk Management and Adaptability in spiral Agile and Extreme Methodologies:

Spiral Methodology: focuses on risk management through frequent cycles of planning, risk analysis, and refinement. It's ideal for high-risk projects, because it proactively identifies and mitigates risks.

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

Extreme programming (XP): Emphasizes continuous integration and test driven development, ensuring high quality code and minimizing defects. It's highly adaptable with close client collaboration and frequent releases.

Given that the project in question has high risks components and there is uncertainty regarding future client needs, Agile would likely be the most suitable methodology. Here's why:

1. Continuous Risk Mitigation:

Agile's frequent feedback loops allow the team to identify and address risks early. Risks are managed through short iterations, with each sprint serving as an opportunity to adjust the approach based on client feedback.

2. Flexibility and Adaptability:

Agile's iterative process allows for flexibility making it easy to adapt to new information or changes in client needs. The client is involved throughput the process and requirements can be adjusted regularly based on real world feedback.

3. Cost Effective Evolution:

Agile helps in delivering incremental improvements. This iterative approach ensures that the project envolves in a manageable, cost effective way, allowing for changes to be incorporated without major disruptions.

Why not spiral or XP?

➤ Spiral is a great risk management approach but it can be more formal and document heavy than Agile; which might slow down adaptation, especially in a fast changing.

➤ XP is highly effective in ensuring high quality code, but it may be overkill for a project where the client's envolving needs are the primary concern. XP emphasizes technical practices, which are excellent but might not fully address the need for iterative feature adjustments.

Agile is the most suitable methodology for project with significant risks and envolving requirements.

Comparison of Waterfall, Agile, Extreme and Spiral Development Models:

Waterfall Methodology:

Waterfall is a linear and sequential development model. It involves distinct phases: requirements gathering, design, implementation, testing, deployment and maintenance.

Agile Methodology:

Agile is an iterative and incremental approach. Development is split into short cycles, and feedback is continuously incorporated into the process.

Extreme programming (XP):

XP is an Agile methodology that emphasizes technical excellence, continuous integration, collaboration and frequent releases.

Spiral Methodology:

Spiral is a risk driven model that combines iterative development with regular risk assessment. It involves continuous planning, prototyping and risk analysis throughout the project.

Question - 3:

Ans: Project A (well defined requirements, strict deadline):

Best Methodology: Waterfall

Why: Since project A has well defined requirements and a strict deadline, waterfall is the most suitable methodology. Its predictability and structured approach make it ideal for projects where the requirements are clear from the start and

changes are minimal.

How it addresses the project:

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

Customer collaboration: Minimal, as requirements are set at the beginning.

Project B (Envolving requirements, uncertain timeline, continuous feedback):

Best Methodology: Agile or spiral

Why: ProjectB requires continuous customer feedback and has envolving requirements, making both Agile and spiral suitable. However, Agile is slightly better suited because of its flexibility and adaptability.

How it addresses the project:

predictability: Agile is less predictable, but this is acceptable given the certain timeline and ~~evo~~ envolving nature of the project.

customer collaboration: Agile emphasizes close collaboration with the customer, allowing them to provide feedback at the end of each sprint.

Spiral Alternative: while spiral also allows for continuous feedback and risk management, it may be more formal and documentation heavy compared to Agile, making it less adaptable in fast-paced environments like project B.

Summary of Methodology suitability:

Project A: waterfall is the most suitable methodology ~~data~~ due to its predictability, structure and ability to meet deadlines with well defined requirements

project B: Agile is the best fit for its flexibility, customer collaboration and ability to adopt to envolving requirements. Spiral could also be an option, but Agile's speed and focus on iterative delivery make it more appropiated for a project with a high degree of uncertainty.

Qusestion 4:

Ans: Principles of software Engineering Ethics:

Software engineering ethics emphasizes the responsibility of engineers to ensure their work priorites the safety, well-being and privacy of users, as well as the broader society impact.

key principles include:

Public welfare: Software engineers must prioritize the public's safety and interests, ensuring their work does not cause harm or unethical consequences.

Professional competence: Engineers should work within their areas of expertise, continuously improving their skills to deliver high quality work.

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

Issues Related to professional Responsibility:

Accountability: Engineers must be accountable for their actions and decisions, ensuring the software is reliable and safe for users.

Conflict of Interest: Engineers should avoid situations where personal or financial interests interfere with their professional duties.

Privacy and security: protecting user data and ensuring compliance with privacy laws is a critical responsibility.

ACM/IEE code of Ethics and Ethical Decision making:

The ACM/IEE code of Ethics provides a frame work for ethical decision making by offering principles that guide software engineers in making responsible choices. It emphasizes:

Public interest: Engineers are encouraged to act in the public's best interest, ensuring their work is safe, secure and ethically sound.

Competence and Quality: Engineers must produce high quality, well tested and reliable software, constantly improving their knowledge.

Question 5:

Ans: The Airpact Reservation system (ARS) is designed to manage flight bookings, passenger reservations, and other related services in an airport or airline. Below are five functional and five non-functional requirements for such a system, explaining how much each contributes to the system's performance, usability and security:

Functional Requirements:

User Registration and Login: Allow user to create and manage accounts, ensuring secure access to booking and personal details.

Flight Search and Booking: Enables users to search for flights and book tickets based on various criteria, providing the core functionality of the system.

Payment processing: secures financial transactions for bookings, ensuring trust and security in Payments.

Notification system: sends update on booking, cancellations and flight status ensuring users and are informed.

Non functional Requirements:

performance: The system should process requests quickly to ensure a smooth user experience, especially during pack times.

Scalability: The system must handle increased users and traffic without compromising performance as the airpact on airline grows.

Availability: Ensures the system is accessible 24/7 with minimal downtime, offering reliability to users.

Security: protects sensitive user data with encryption and secure authentication to maintain privacy and trust.

Maintainability: Allows for easy updates and bug fixes, ensuring long-term system reliability and adaptability.

Question 6:

Ans: The V model is a sequential software development life cycle model that emphasize the synchronization of development and testing activities. It's named after V shaped formed by the development and testing phase.

Key Relationships:

Verification (Left sides):

- Requirement analysis
- system design

- Architectural design
- Module design

Validation (Right side):

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

How it works:

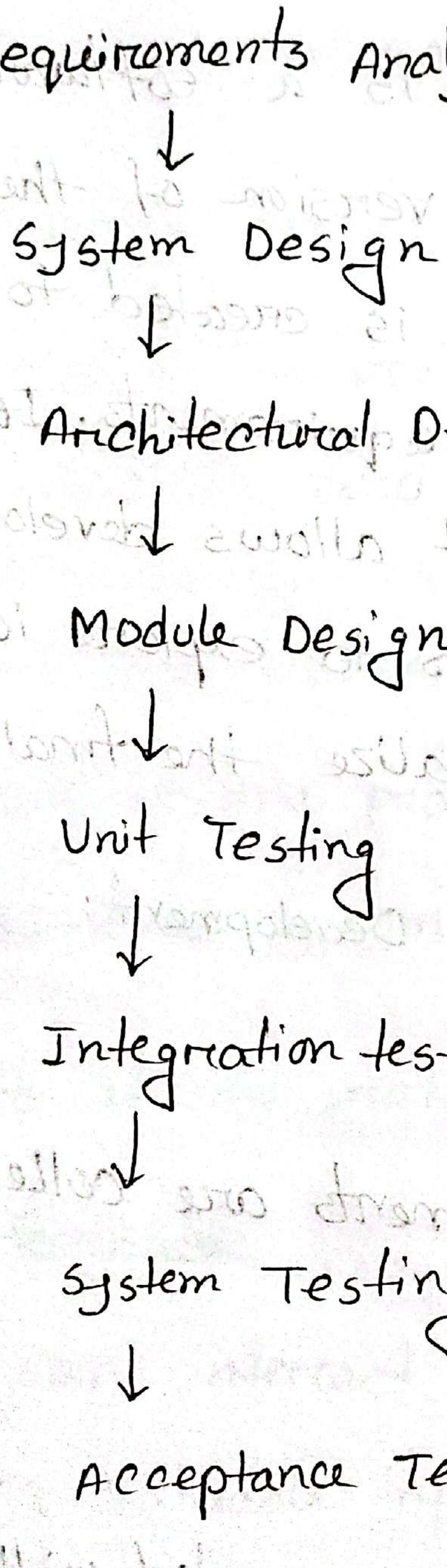
Development: starts with requirements analysis and progress through design phases.

Testing: corresponding testing phases are planned and executed in parallel with development.

verification: Ensures the product is built correctly.

validation: Ensures the product is built right:

Visual Representation:



Question 7:

Ans: Prototype Development is a software development model where a basic version of the system known as a prototype is created to gather feedback and refine requirements. It's an iterative process that allows developers, stakeholders, and users to explore ideas, test functionalities and visualize the final product.

Key stages of prototype Development:

Requirements Gathering:

Basic, high-level requirements are collected to build the prototype.

prototype Development:

A simple, working model is created with key features for initial testing.

User feedback:

users interact with the prototype and provide feedback on functionality and design.

Refinement:

The prototype is revised based on feedback, adding features and improving functionality.

final Product Development:

once refined, the prototype becomes the foundation for the final system.

Benefits of the prototyping model:

User feedback:

Users can interact with the prototype de early, providing valuable input to improve the product.

Risk Reduction:

Identifying issues early reduces the risk of building a product that doesn't meet user needs.

Question 8:

Ans: The process improvement cycle focuses on continuously enhancing software development process to improve quality, efficiency and performance. It typically follows a structured approach like PDCA (Plan-Do-check-Act):

Plan: Identify areas for improvement and define goals or process to the enhance.

Do: Implement the changes or improvements in a controlled environment.

check: Measure the impact of the changes using process metrics.

key stages in process Improvement:

Assesment: Evaluate the current process to identify inefficiencies or bottlenecks.

Analysis: Investigate the root cause of problems and determine where improvements are needed.

Implementation: Apply changes or improvements to the process.

Evaluation: Measure the results and impact of the changes.

standardization: If successful, standardize the improvements and make them part of the regular process.

In summary, the process improvement cycle is about continuously improving software processes by assessing, implementing and measuring changes. process metrics help monitor progress and ensure improvements leads to better quality and efficiency.

Question 9:

Ans: The software Engineering Institute capability Maturity Model (SEI CMM) is a frame work used to access and improve software developm processes within an organization. It helps organizations evaluate their current process, identify areas for improvement and provide roadmap for achieving higher level of proces maturity and capability.

Five Levels of the SEI CMM:

Level 1 - Initial (Ad hoc):

At this level, processes are often chaotic and unorganized. Software development is typically reactive with no standardized procedures of or consistency.

Level 2 - Managed (Repeatable):

At this level, basic project management processes are established. The organization starts to repeat successful practices, focusing on ensuring projects are completed on time and within budget.

Level 3 - Defined (Proactive):

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

This level leads to more proactive management of processes. with a focus on consistency accross all projects. It reduces variability and enhances coordination within teams, leading to higher quality and more predictable outcomes.

Question 10:

Ans: Core principles of Agile software Development:

Negotiation:

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

Responding to change over following a plan:

Agile values flexibility, allowing the project to adopt to changes in requirements or market conditions.

Delivering working software frequently:

Agile encourages frequent delivery of small, functional increments of the product, often every 1-4 weeks.

Individuals and Interactions over processes and Tools:

Agile prioritizes communication and collaboration

within the team rather than relying on rigid processes and Tools.

simplicity and focus:

Agile promotes a simplicity in design and functionality focusing only on what's necessary for the project.

self Organization Teams:

Agile relies on teams to organize and manage themselves, empowering # team members to make decisions and Collaborate effectively.

Agile methods bring flexibility, faster delivery and customer satisfaction, making them great for projects with envolving requirements or those that need quick iterations. However, they can be challenging in environments with rigid structures of where team experience and coordination are lacking.

Question 11:

Ans: The Extreme programming (XP) release cycle focuses 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:

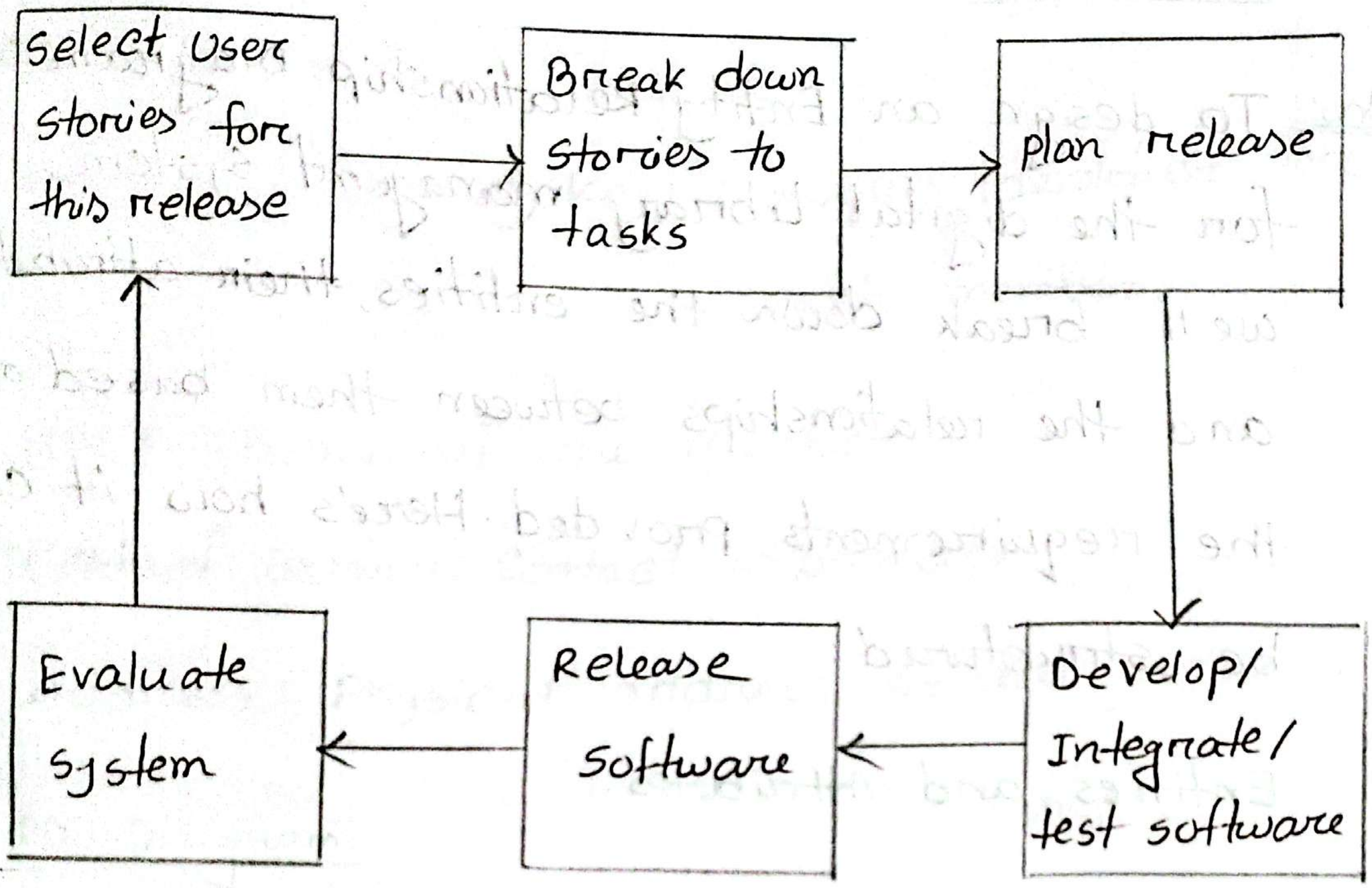
Extreme programming (XP) Release cycle:

Planning: Initial and iteration planning define features and deliverables.

Development Iteration: Code is developed in short iterations.

Continuous Testing: Automated tests ensure functionality and quality.

Customer feedback: Customer reviews each iteration and provides input.



Influential XP programming practices:

Pair programming:

Two developers collaborate on the same code to improve quality.

Test Driven Development:

Write tests before code to ensure correctness.

Continuous Integration:

Integrate code frequently with automated testing.

Question 12:

Ans: To design an Entity Relationship Diagram for the digital Library management System, we'll break down the entities, their attributes and the relationships between them based on the requirements provided. Here's how it can be structured:

Entities and Attributes:

Book:

Attributes:

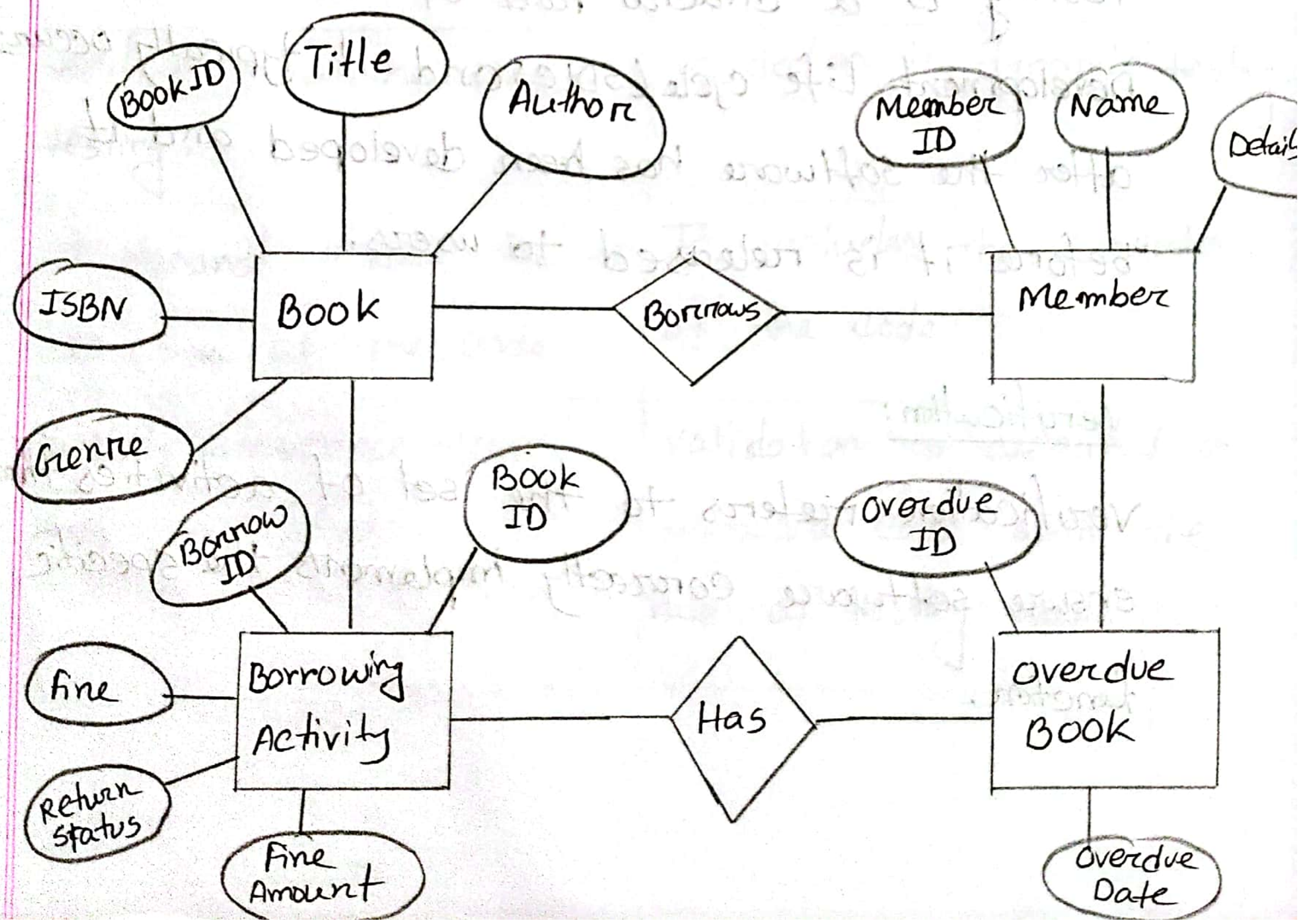
- BookID (Primary key): Unique identifier for each book.
- Title: Title of the book
- Author: Author of the book.
- ISBN: International standard Book Number.
- Genre: Genre of the book

Member

Attributes

- Member ID (Primary key) : Unique identifier for each member.
- Name : Name of the member.
- Contact Details : Contact information
- Address : Physical address of the member.

ERD Diagram:



Question 13:

Ans: 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 goal is to identify bugs, ensure the software's quality, and verify that it performs as expected under various conditions.

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 users.

Verification:

verification refers to the set of activities that ensure software correctly implements the specific function.

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.
It includes checking documents, designs, codes and programs.	It includes testing and validating the actual product.
verification is the static testing.	validation is dynamic testing.
It doesnot include the execution of the code.	It includes the execution of the code.
Quality assurance team does verification.	validation is executed on software code with the help of testing team.

Question 14

Ans: A layered Architecture organizes the system into layers, each with specific responsibilities. Here's how the Online Judge system can be divided:-

1) Presentation Layer:

- Handles user interactions like login, problem selection and result display.
- provides a user friendly interface.

Example: A webpage where users submit code or see result

11) Application Layer:

- processes request from the presentation Layer.
- Coordinates communication between the user interface and the business Logic.

Example: Handles requests like 'submit code' or 'retrieve problems'.

Business Logic Layer:

- Implement the core functionalities: code compilation, execution and result evaluation.
- Ensures the code is tested against predefined test cases securely and efficiently.

Example: The system evaluates the submitted code and checks if it passes all test cases.

Data Layer:

- stores and retrieves data, such as user accounts problems, test cases and result.
- Ensures secure and efficient data management.

Example: A database storing all problems and user submitted solution.

How the Architecture Ensure Efficiency?

Scalability:

- Each Layer can be scaled independently.

Maintainability:

- Change in one Layer do not affect others, making update easier.

Performance:

- clear separation of responsibility enslures smooth data flow and optimized system operation.

Question 17:

Ans: Quality Assurance (QA) and quality control (QC) are both important for ensuring software quality but they focus on different aspects.

what is QA?

- QA is process oriented.
- It ensures that proper methods and followed to prevent defects.
- It focuses on the entire development.
- process rather than just the final product.

Example: creating code, testing guidelines and regular audits to ensure quality.

what is QC?

- QC is product oriented.
- It checks the final software to find.
- It involves testing, reviewing and inspecting.

Example: Running test cases on an application to find and fix bugs before release.

Difference between QA and QC:

Aspect	QA (Quality Assurance)	QC (Quality Control)
Focus	Process on development.	Final product quality.
Goal	prevent defects.	Detect and fix defects.
Approach	Proactive	Reactive
Method	Entire Reviews, audit, Process improvements.	Testing, inspections, bug, fixing.

Impediments (challenges) to QA and QC:

for QA:

- Resistance to process changes.
- Lack of proper documentation.

- Time constrainst for process improvements.

For Qc:

- Testing may not find all defects.
- fixing issues late increases costs.
- Limited resources for testing.

Question 18:

Ans: Is the Goal of QA just to find Bug?

No, the Goal of Quality Assurance (QA) is much more than just finding bugs. QA ensures that the process used to build software are correct, efficient and result o in high quality products. while finding and fixing bugs early is a part of QA, the main focus is on preventing defects and ensuring that the software meets user needs.

Role of QA in each SDLC phase:-

Requirement Analysis phase:

- Ensures that requirement are clean and testable.
- Identifies any unclear or missing details in the requirements.
- Prevents future problems due to miss and requirements.

Design phase:

- Reviews the system design to ensure it meets requirements.
- checks for potential design issues before development starts.
- Helps avoid a major changes later in development.

Development phase:

- Ensure developers follow coding standards.

- Helps with unit testing and early errors detections.
- Reduces the chances of masors bugs during testing.

Testing phase:

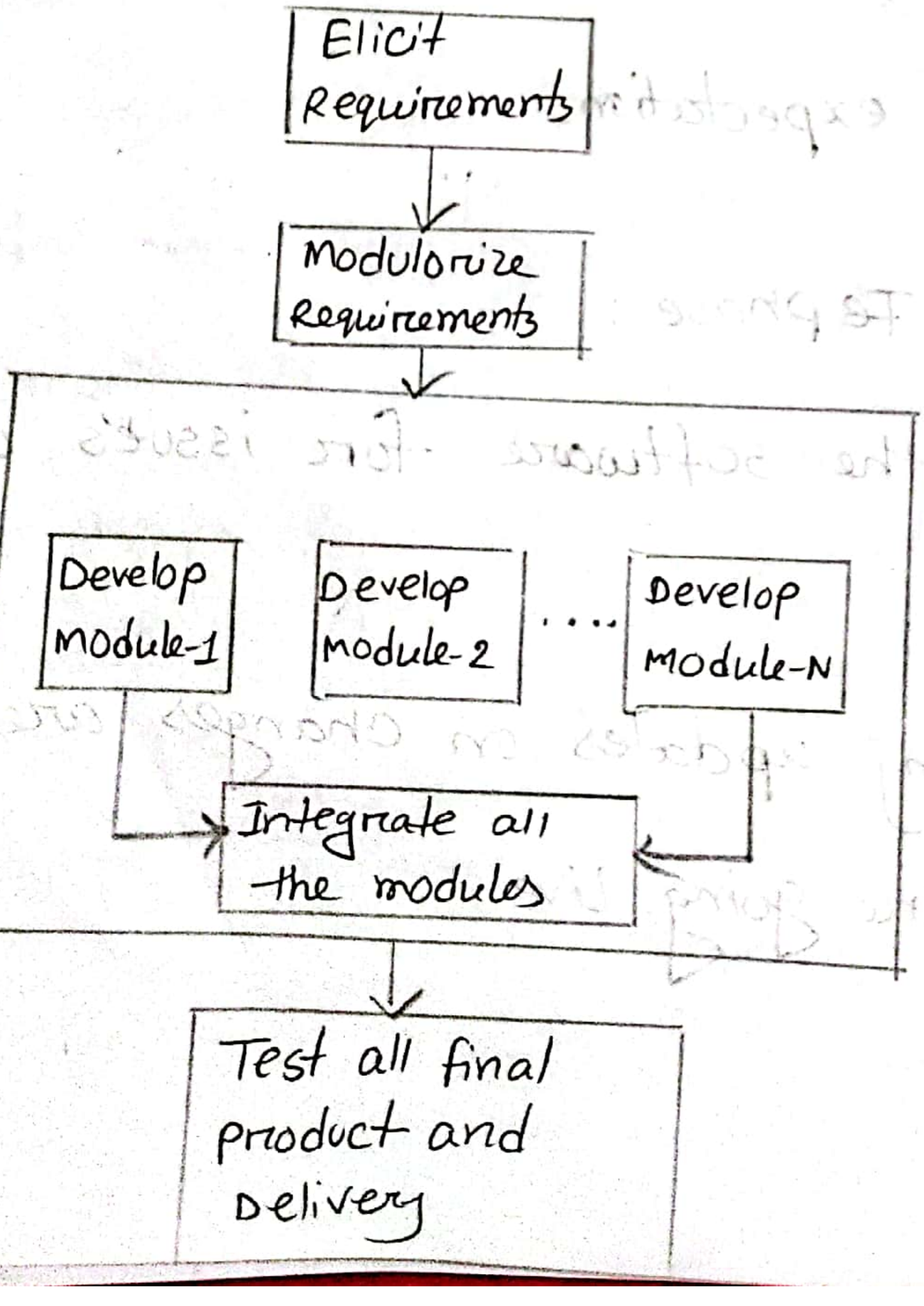
- Runs different tests like functional, Integration, and system testing.
- finds and reports bugs to be fixed.
- Ensures software works as intended and meets user expectations.

Maintenance Tephase:

- Monitors the software for issues after release.
- Ensures any updates on changes are proper tested before going live.

Question 10:

Ans: The Rapid Application Development (RAD) model is a software development methodology that emphasizes speed and flexibility. It focuses on quick development prototypes and delivering functional components in short cycles. RAD aims to meet changing user needs and deliver quality software faster.



Key phases of RAD model

Requirements planning:

- Involves gathering and understanding user requirements.
- Stakeholders collaborate to define the product scope and goals.

outcome: A clear plan for what needs to be developed.

construction:

- Development of the functional system based on user feedback.
- Uses iterative coding, testing and integration.

outcome: A near complete version of the software.

Cutover

- final testing, user training and system deployment.

Principle of RAD model:

User involvement: Continuous feedback users throughout development.

Iterative Development: Software is build increamently with regular updates.

Prototyping: Early version of the system a created for review and feedback.

Component Reuse: Pre built components are used to speed up development.

Advantage of RAD model:

Faster Delivery: Quick itercations and prototyping allow rapid development.

Flexibility: Easily adapts to changing user requirements.

High user satisfaction: Users are actively involved, ensuring the system meets their needs.

Reduced Risk: Problem are identified early through continuous feedback.

How RAD supports faster Delivery and maintains quality:-

Prototyping: Early and frequent prototype allow users to identify issues and suggest changes.

Iterative process: Small, incremental builds help focus on specific features, ensuring faster delivery.

User feedback: Direct involvement of users ensures the software aligns with their needs.

Question 20:

Ans: White box testing focuses on internal structur and Logic. We will apply code coverage and data coverage methods to enswre all possible execution paths and tested.

Production Code (Java implementation):-

This Java program read two integers x, y checks for different conditions and prints appropiate outputs.

```
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.nextInt();
    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,;i++){
  if(i%y==0){
  system.out.println(i);
}
}
}
```

Junit Test class

The following Junit test class implements test cases based on the decision table.

```
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 ("j is zero");
} else if (x==0){
  println ("x is zero");
}
else {
for (int i=1; i<=x; i++){
  if (i%j==0){
println (string value of (i));
}
}
}
@Test
Public void TestyIsZero(){
output.clear();
process(5,0);
assertEquals (List of ("j is zero"), output);
}
```

```
@Test
Public void TestX Is zero(){
  output.clear();
  Process(0,3);
  assertEquals(List of("x is zero"),output);
}
@Test
public void TestLoop DoesNotRun(){
  output.clear();
  process(0,2);
  assertEquals(Listof(),output);
}
```

Question 21

Ans: Here is a Junit 5 test code that demonstrates

- Exception handling
- Set up Function
- Timeout rule

Production Code (Calculator.java)

```
Public class calculator {
// simple addition function
Public int add (int a, int b) {
    return a+b;
}
// Division function (can throw exception)
public int divide (int a, int b) {
    if (b==0) {
    throw new Arithmatic Exception ("/by zero");
}
```

```
        return a/b;
    }

    // simulating a long running operation
    Public void LongRunningOperation(){
    try{
      Thread.sleep(500);
    }
    catch(InterruptedException e){
      e.printStackTrace();
    }
    }
}
```

Junit 4 Test code:

```
import org.junit.Before
import org.junit.Rule
import org.junit.Test
```

```
import org.junit.rules.ExpectedException
import static org.junit.Assert.*;

Public class CalculatorTest{
  Private Calculator calculator;
  // step 1: set up function, run before each test
@before
Public void setup(){
  Calculator = new Calculator();
@Rule
 public ExpectedException exceptionRule = Expected
Expected.none();

@Test
Public void testDivisionByZero(){
  exceptionRule.expect(ArithmeticException.class);
  exceptionRule.expectMessage("/by zero");
```

```
    calculator.divide(10,0);
}

@Test(timeout=1000)

Public void testLongRunningOperation(){

    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.