Bindu

## Answer to the Question No:-1

*Scrum-Based E-commerce Project:-

A Product backlog (Breaking user stories into tasks)

user story:1 :- "As a user, I want to log in securely so that I can access my account"

- Tasks:-

1) Design the log in page.
2) Set up the database to store dataset.
3) Develop the backend for the login System.
4) Encrypt passwords.
5) valid inputs (Check if username and password are correct)
6) Add a forget password option.
7) Tests all features and fix bugs.

user story:-2 :- "As a user, I want to search for Products by Category to find items easily"

- Tasks:-

1. Design the Searchbar.
2. Create a database to store Product information

Bindu

3) Develop the backend for the Search feature.

4) Implement category-based filtering.

5) Optimise search for large database.

6) Test the feature.

7) Conduct user testing for the feature.

B :- Sprint planning (setting Priorities)

- Login Feature (user story 1) :-
  → Highest priority because users need secure access to their accounts.
- Search Feature (user story : 2) :-
  → Second priority because users can manually browse Products if the Search feature is not ready.

C) :- Scrum Board (For Task Tracking)

| To Do | In Progress | Done |
|---|---|---|
| Design login page | Develop backend API | Database setup completed. |
| Encrypted password | validate inputs | |
| Design Search bar | | |

Bindu

D:- How the Scrum board works?

- To Do:- Task that haven't started yet.
- In Progress:- Tasks that are currently being worked on.
- Done:- Tasks that are completed.

Tasks move step by step from "To Do" → "In Progress" → "Done"

The Scrum master monitors Progress daily and helps solve any issues.

## Answer to the Question no:- 2

For a Project with high risks and changing requirements, Choosing the right methodology is important to manage risks and adapt to change effectively. Let's look at how each methodology helps.

1) Spiral methodology:-

- Risk management:- The spiral model focusse

on managing risk by constantly reviewing them throughout the Project. It allows for quick identification of risks and fixeds them early, with testing and prototyping in each phase.

- Adaptability:- The spiral model is flexible and iterative, allowing changes based on new informations on requirements. This is helpful when the Project is complex on has many unknowns.

2) Agile methodology:-

- Risk management:- Agile works in small steps called Sprints, where the team checks and deals with risks regularly. This means risks can be addressed early and issues are found quickly.
- Adaptability:- Agile design to adaptable. The team can easily adjust to changes in client needs, as it involves constant

feedback and changes after each Sprint.

3) Extreme Programming (XP):-

- Risk manggement :- XP uses continuous testing and regular release, which helps identify Problems immediately. The team also work in pairs. which helps reduce errors and risks in code.
- Adaptability:- XP is highly adaptable, with short cycle and constant client feedback, So, it can quickly respond to changing needs.

* For this Project, Agile is the best choice.

why Agile ?:-

- Risk management:- Agile's small, regular updates helps the team manage risk more easily.
- Adaptability:- Agile's flexibility means the Product can evolve as the client's

needs changes.

- Cost effectiveness:-
  Agile allows the ~~to~~ ~~the~~ team to focus on the most important features first saving time and money.

## Answer to the Question no:-3

To choose the best development methodology for each Project, we need to consider the characteristics of both Projects.

- ProjectA:- well-defined requirement and a ~~Strict~~ deadline.
- ProjectB:- Evolving requirements, uncertain timeline and continuous customen feedback.

Lets compare the methodologies based on these characteristics.

1) waterfall model:-

- predictability:- The waterfall model is

linear and Sequential. It works Project will well-defined requirements.

- Customer Collaborations:- Minimal Collaboration after the initial requirement phase, customer feedback is given only after the final product is delivered.
- Risk management:- It is less flexible to change. If risks arise later, it becomes difficult to address them because changes require going back to earlier stages.

2) Agile model:-

- Predictability:- Agile is iterative. Delivering small pieces of the Product regularly. It is less Predictable but more adaptable to changes.
- Customer collaboration:- continuous customer involvement and feedback is key in Agile. The product evolves based on customer needs throughout the process.

Bindu

Risk management:- Agile helps in managing risk by working in small sprints. Risk are identified early and addressed immediately, making it highly flexible.

3) Extreme Programming (XP):-

- Predictability:- similar to Agile, xp focused on short cycles and continuous delivery, which can make it ~~not~~ less predictable.
- Customer collaboration:- xp emphasizes close collaboration with customers, continuous feedback, and regular adjustment to the product are ~~the~~ ~~reg~~ center of xp.
- Risk management:- xp uses practices like pair Programming, continuous testing, and frequent release to manage risk effectively and improve quality. This helps mitigate technical risks and ensure a high-quality product.

Bindu

(4) Spiral model:-

- predictability:- The spiral model combines iterative development with risk management. It is flexible but allows for better planning and refinement over time.
- Customer collaboration:- It involves customer in regular reviews and the Product evolves with their feedback.

Risk management:- The spiral model's key strength its focus on risk management at every stage. Risks are assessed and addressed at each iteration, which helps in high-risk Projects.

*Which methodology is best for each project:-

For Project-A:-

Best methodology:- waterfall on spiral:-

- waterfall works well for clear requirement and strict deadline. However, if there are potential risks or changes during

Bindu

the Process, the Spiral model would be better as it includes risk management and allows adjustments along the way.

For Project-B

- Best methodology:- Agile on xp
- Since the requirement are evolving and there's continuous customer feedback, Agile would be the most suitable methodology. xp could also be considered if technical excellence, ~~frequently~~ frequent testing, and customer collaboration are prioritized.

Answer to the Question.

Q8-4

Principle of Software Engineering ethics.

1) Public Interest :- Always prioritize the well-being of society, users and the

Bindu

public over personal or compani benefits.

2) Quality of works:- Deliver high quality software that meets user needs, work effecientfy and avoid harm.

3) Honesty:- Be truthful about what software can and can't do. Don't mislead clients or users.

4) Privacy and confidentiality:- Protect user data and ensure confidentiality. Do not misuse information.

5) Competence:- Only take up work you are qualified for and continuously improve your skills to maintain Professional standards.

Professional Responsibility Issues:-

1) Accountability:- Software engineer are responsible for the impact of their work, both good and bad.

2) Ethical Dilemmas:- Sometimes engineers face tough choices, like balancing busines

Bindu

goals with user safety.

3) Plagiarism and Integrity :- Using others code and ideas without acknowledgement violates ethical standards.

Role of Acm/IEEE code of Ethics :-

The Acm/IEEE code of Ethics ~~provides~~ provides guidelines to help Software engineers make ethical decision.

1) Act in Public interest :- Always prioritize safety, fairness and social good in Software development.

2) Be honest and Trustworthy :- Avoid misleading stakeholders or users about Software capabilities or limitations.

3) Respect Privacy :- Protect user data and ensure it's used responsibly

4) Continuous learning :- Stay updated

with the latest knowledge and best practices.

5) Work with integrity :- Avoid bias, unfair treatment or harm in software processes.

## Answer to the Question No:- 5

### Functional Requirement:-

1) Flight Booking :- users must be able to search for flight, select one and book tickets.
   - contribution :- Provides the core functionality of the system and enhances user satisfaction.

2) User registration and login :- The system should allow user to create accounts and log in securely.
   - contribution :- Ensures personalized services and Protects user information.

3) Payment processing :- The system should

Bindu

Support multiple payment method like credit cards, debit cards and digital wallets.

- Contribution:- Simplifies the transaction process, improving the user experience.

4) Flight Rescheduling on Cacellation:- User must be able to reschedule or cancel bookings.

- Contribution:- adds flexibility, making the system more user-friendly.

5) Real-Time Updates:- The System should provide real-time flight status updates.

- Contribution:- keeps users informed improving reliability and trust.

Non-Functional Requirement:-

1) performance:- The System must handle up to 10,000 users concurrently without slowing down.

- Contribution:- Ensures smoth operation even during peak times.

2) Usabilityes;- The System should have a simple and intuitive interface to easy navigation.

• Contribution:- Improves the user experience making the System accessible to all.

3) Security:- All transactions and user data must be encrypted.

• contribution:- Protects sensitive information boosting user trust.

4) Scalability:- The System should support additional features on handle increased user traffic in the future,

• Contribution:- Ensures long-term maintainability.

5) Availability:- The System should be operational 99.9% of the time to minimize down time,

contribution:- Improve reliability and ensures

Bindu

continuous services for users.

## Answer to the Question NO:-6

**Definition:-** The V-model (verification and validation model) is a software development model that emphasizes the relationship between development phase and ~~corresponding~~ corresponding testing phases. The 'V' shape visually represents the sequence of activities, where the left side ~~involves~~ involves testing.

* **Key Features of the V-model:-**

1) **Development phase:-** (Left side of the V)

- **Requirement Analysis:-** Understanding what the customer wants.
- **Test:-** Acceptance Testing ensures the final Product meets user needs.
- **System Design:-** Planning how the

Bindu

Software will work as a whole.

- Test:- System Testing validates the integration of # all components.

- Architectural Design:- Designing the modules/components of the software.
  Test:- Integration Testing checks how modules work together.
- Module Design:- Detailed design of each module.
- Test:- Unit Testing verifies individual modules work correctly.

2) Testing phases (Right side of the V)

- Each development phase has a ~~Corres~~ corresponding test phase to validate it
- Testing starts early with plans and progresses alongside development.

## Relationships Between Development and testing

1) Requirement Analysis ↔ Acceptance Testing.
   - Ensures the software satisfies user needs.
2) System Design ↔ System testing
   - Verifies the complete system's behavior and performance.
3) Architecture Design ↔ Integration Testing.
   - Checks interactions between different modules.
4) Module Design ↔ Unit Testing,
   - Ensures each small part (module) work correctly.

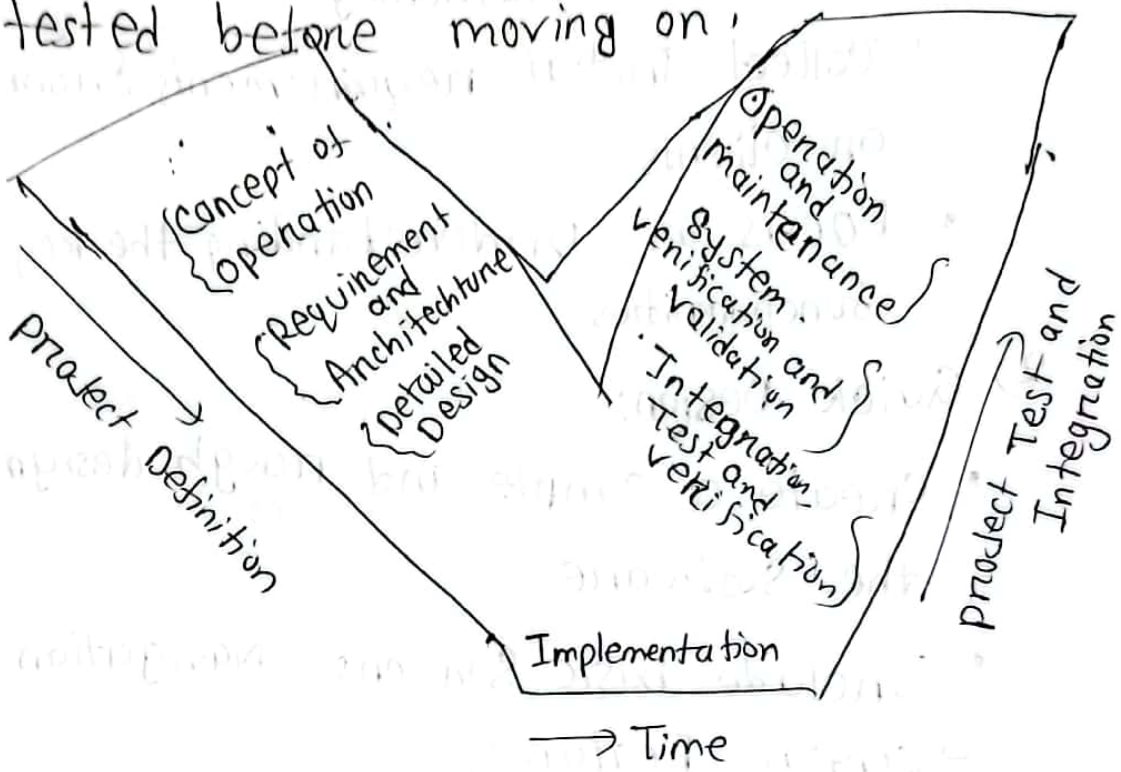
### Advantage of the V-Model:-

- Early detection of Errors:- Testing is planned alongside development.
- Clean Structure:- Easy to understand and

Bindu

manage.

* Improve Quality:- Each phase is thoroughly tested before moving on.




Answer to the Question No:- 7

Prototype development is a process used in Software engineering to create a working model of the Software before full-scale development begins. It helps refine requirement and ensures that the final Products meets user expectations.

Key Stages of Prototype Development:

1) Requirement Gathering:-
   - Collect Initial requirements from users or clients.
   - Focus on understanding the key ~~function~~ functionalities.

2) Quick Design:-
   - Create a simple and rough design of the software,
   - Include basic Screens, Navigation and essential Feature,

3) Prototype Building:-
   - Develop the Prototype with the necessary features,
   - Ensure it is functional but not the final Product.

4) User Evaluation:-
   - Share the Prototype with users for feedback,

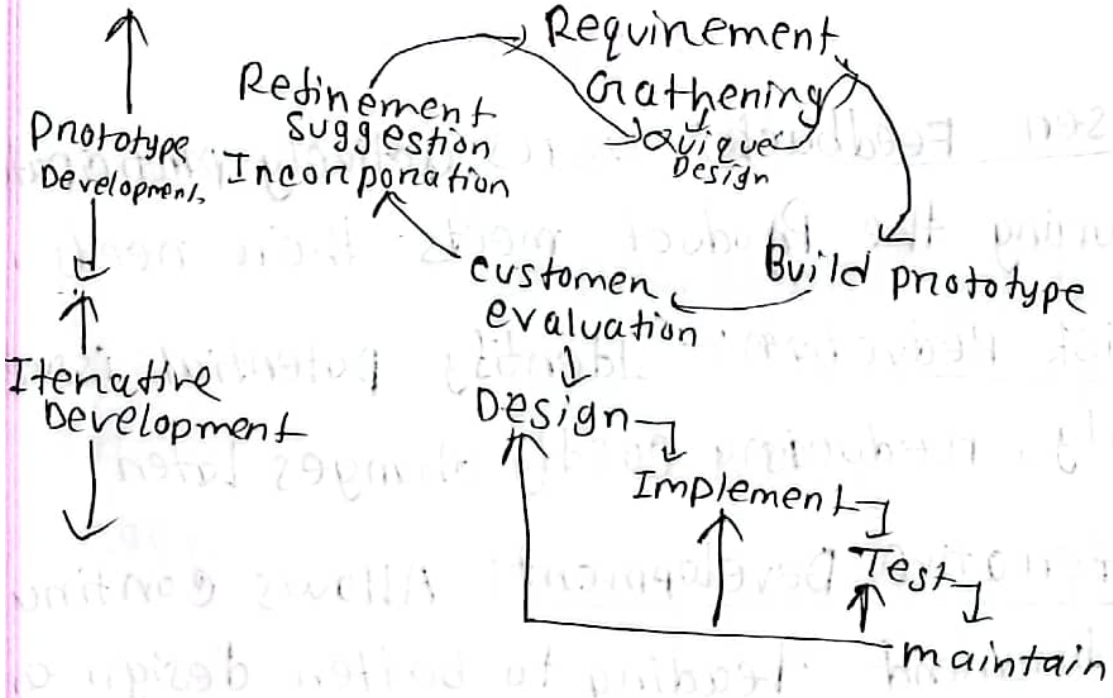
- understand what they like on want to improve,

5) Refinement:-

- Incorporate user feedback to improve the Prototype,
- Repeat the process until requirements are clear.

6) Final Product Design:-

- Once the Prototype is finalized, it becomes the basic for the actual software design.



How Prototyping helps Refine Requirement :-

- Clear understanding :- It shows how the system will work, helping identify gaps on unclear requirements.
- Early Feedback :- users can see and test the Prototype, Providing feedback before full development.
- Better communication :- Improve communication between developers and users by offering a visual and functional reference.

Benefit of the Prototype model :-

1) User Feedback :- users actively participate ensuring the Product meets their needs.
2) Risk Reduction :- Identify potential issues early, reducing costly changes later.
3) Iterative Development :- Allows continuous refinement leading to better design and functionality.

4) Time Saving:- Helps avoid misunderstanding and reduces rework by clarifying requirements early.

When to use Prototyping:

1) when you don't fully understand the requirement from the start.
2) when you need frequent feedback from users,
3) when the system is complex or the user experience is important.

Answer to the question no:- 8

The Process improvement cycle helps make Software development better by finding problems, fixing them, and making the process smoother and more efficient.

Key Stages of the Process Improvement Cycle:-

1) measure the Process:-
   - Collect data about how tasks are being done.

- Example:- Check how long it takes to finish coding on how many bugs are found.

2) Analyze the data:-

- Look at the data to find problems.
- Example:- If coding takes too long, figure out why.

3) Make changes:-

- Fix the Problems by improving the process.
- Example:- Use better tools, teach the team new skills on simplify tasks

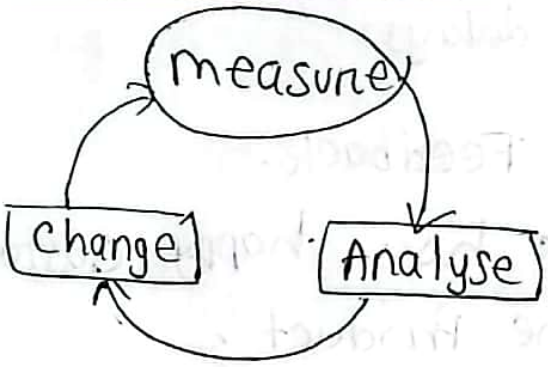
4) Check Results:-

- See if the changes worked by comparing the new performance with the old one.
- Example:- Did coding become faster on have fewer bugs?

5) Repeat:-

- keep improving the process over time.

Bindu




Common matrics(measurement) in Software proces

1) Time Taken:-

- measures how long tasks take, like coding on testing,
- Helps find slow areas.

2) Defect count:-

- Tracks the number of bugs in the software,
- Show the quality of the code.

3) Team Productivity:-

- measures how much work is completed in a specific time;
- Shows how efficient the team is.

4) Cycle Time:-

- Time to complete a task, like adding a feature on fixing a bug.

- Helps find delays.

5) Customen Feedback:-

- Measures how happy customers are with the Product.

Why these matrics are useful?:-

- metrics show where things are slow or not working well.
- Compare result before and after making changes.
- Helps teams decide what to fix or improve.
- ~~Ensure the Software gets better~~

Answer to the Question No: 9

SEI/CMM:- The Software Institute Capability Maturity Model (SEI-CMM) is a framework used to improve the Software development Process in an organization. It ~~help~~ helps companies gradually become better

at managing and delivering Software product capacity and

Five level of maturity:

1) Level-1 (Initial)(unorganized):-

- Process are unpredictable, chaotic and reactive.
- Success depends on individual effort not the Process.
- Example:- Team fix Problem as they occur without any Proper plan.
- Impact:- Low consistency; Projects often fail on miss deadline.

2) Level-2:- Repeatable (Basic management):-

- Process are more organized and repeatable for Similar Projects.
- Basic Project management practice ~~are~~ are in place, like timeline and budgets.
- Example:- Teams use checklists and track Project Progress.

- Impact:- Reduces risk of failure and allows for consistent outcomes.

3) Level-3:- Defined (Standardized process):-

- Processes are documented, Standardized and followed across the organization.
- Team work with defined guidelines and clean structure,
- Example:- Every Project follows the same development framework.
- Impact:- Ensures consistancy and quality accross all Project.

4) managed (measured and controlled)

- Processes are managed and monitored using matrix.
- Teams use data to Predict and control Project outcomes.
- Example:- Tracking defect rate and time taken for tasks.
- Impact:- Improve effeciency and helps in making better decision.

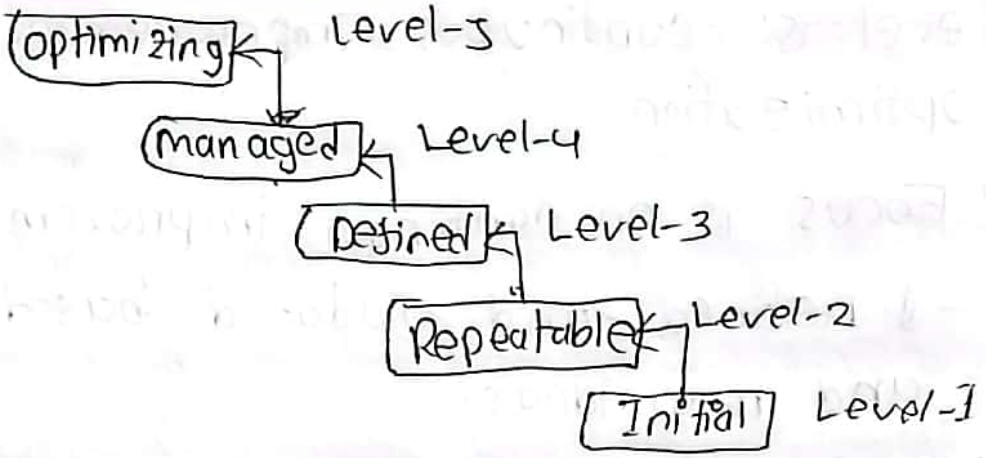
5) Level-5: (Continuous Improvement):- optimization:

- Focus is on continous improvement, refined and updated based on feedback and new ideas.
- Example:- Regular improving tools, techniques and workflows.

Impact:- Ensures high-quality Products and adaptability to change.

How Each helps improve Processes:-

- Level-1:- Highlight the need for structured process to avoid chaos,
- Level-2:- Introduces basic Project control to reduce risks and improve consistency,
- Level-3:- Standarized Prosess to ensure same quality accross team.
- Level-4:- Use matrics to monitor perdormance and solve Problems eanly,
- Level-5:- Focuses on growth and innovation ensuring long term success.




## Answer to the Question No:-10

**Agile:-** Agile Software development is a flexible and iterative approach that focuses on delivering valve to customers quickly and adapting to change.

**Core Principle of Agile:-**

1) **Customer collaboration:-**
   - Work closely with customers for feedback and improvement.
   - **Example:** Regularly show working software to clients.

2) **Responding to change:-**
   - Adapt plans quickly when requirements

change.

- Example: Adjust priorities if the clients needs evolve.

3) Frequent Delivery:-

- Deliver small, working parts of the software regularly.
- Example: Release updates every 1-2 weeks

4) Teamwork and communication:-

- Teams and Stakeholders work together daily.
- Example:- Developers and testers share ideas in daily meetings.

5) Simplicity:-

- Focus on doing only what is necessary. Avoid extra work.
- Example:- Build only the features the customer needs now.

## Application in Different Environments:-

1) Startup:-

- Agile fits well because ·ill allows quick changes based on customer feedback.
- Example:-Launching a new app and making updates based on user reviews.

2) Large Organizations:-

- Teams can break big Projects into smaller tasks and work collaboratively.
- Example:- A bank creating a mobile app

3) Complex Project:-

- Agile ensures frequent testing and delivery reducing risk,
- Example:- Building a healthcare System that needs continuous adjustments.

## Benefits of Agile methods :-

1) Faster Delivery :-

- Small releases ensure customers get usable features quickly.

2) Improved quality :-

- Continuous testing and feedback improve Software quality.

3) Flexibility :-

- Agile adopts to changing customen needs.

4) Higher customen satisfaction :-

- Customers see Progress regularly and get exactly what they need.

## Challenges of Agile methods

1) Frequent Changes :-

- Too many changes can delay progress.

2) Team ~~collabor~~ coordination :-

- Requires strong communications and collaborations.

3) Not Suitable for all Projects:-

- For Projects with fixed requirements, Agile might not be the best choice.

## Answer to the Question NO:-11

The release cycle in Extreme programming (XP) is all about delivering small, working part of the software quickly and improving it through regular feedback.

Release Cycle of XP:

1) planning:-

- meet with the customer to gather user stories (simple requirements)
- Prioritize story based on importance.

2) Iteration planning:-

- Divide the work into small tasks (iterations)

that can be completed in 1-2 weeks.

3) Coding and Testing:

- Develop code for the tasks and test it immediately.
- use Test Driven Development (TDD) & write test before writing the code.

4) Release:

- Deliver small working parts of the software to the customer.
- Gather feedback for the next iteration.

5) maintainance:

- use feedback to improve the software in the next iteration.

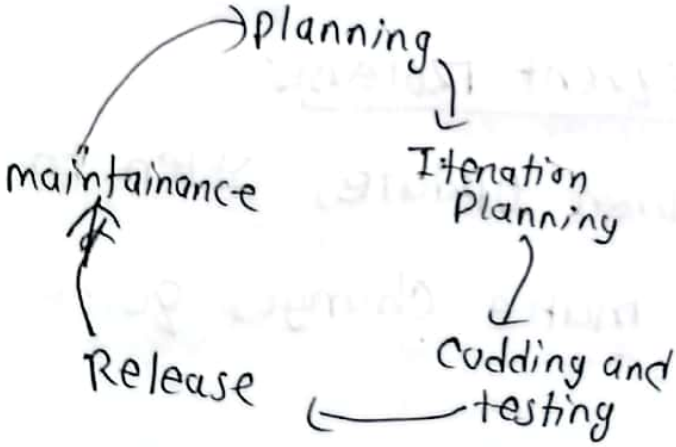



Fig: Release cycle of XP

Influential Programming Practice in XP :-

1) pair Programming :-
- two developers work together on the same code to improve quality.

2) Test Driven- Development (TDD)
- write tests before coding to ensure the software works as expected.

3) continuous integration :-
- Regularly integrate new code with the main Project to catch bug early.

4) Simple Design :-
- keep the Design simple to focus on what's necessary.

5) Frequent Release :-
- Deliver updates often to get feedback and make changes quickly.

# Answer to the Question no. 12

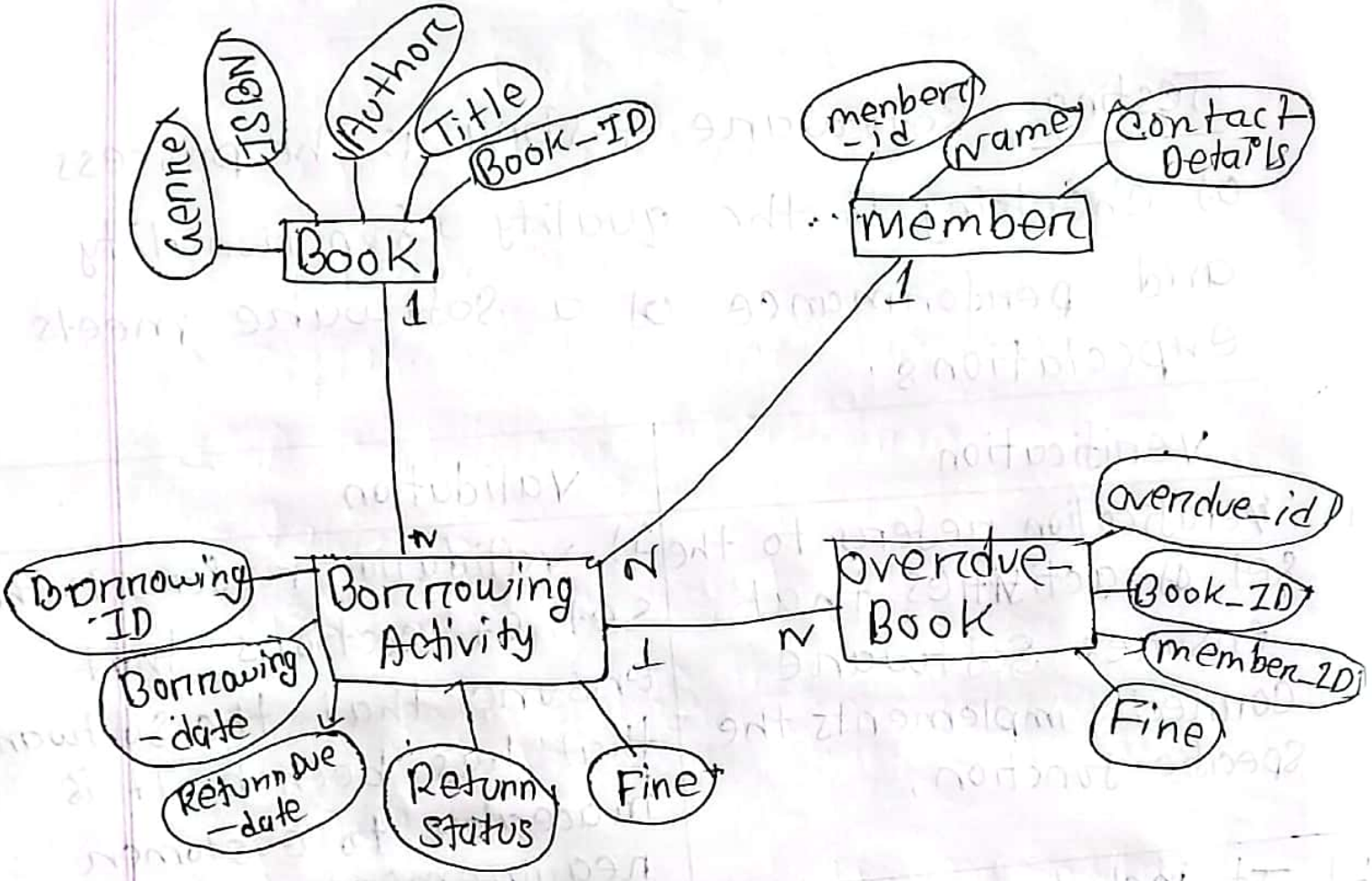



Fig:- Entity Relationship Diagram of Library management System.

## Answer to the Question No:- 13

**Testing:-** Software testing is the process of checking if the quality, functionality and performance of a software meets expectations.

| Verification | Validation |
|---|---|
| 1) Verification refers to the set of activities that ensures Software correctly implements the specific function. | 1) Validation refers to the set of actions that ensure that the Software that has been built is traceable to customer requirement. |
| 2) It includes checking documents, designs, codes and Programs. | 2) It includes the testing and validating the actual Product. |
| 3) It is static testing. | 3) It is dynamic testing. |
| 4) It does not include the execution of the code. | 4) It includes the execution of the code. |
| 5) It comes before validation. | 5) It comes after verification. |

Answer to the Question No: 14

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 users interactions like login, problem selection and result display.
- Provides a user-friendly interface (web or mobile)

Example:- A webpage where users submit code or see result.

2) Application Layer:-

Responsibilities:-

- 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".

3) Business Logic Layer:-

- Responsibilities:-
  - 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.

4) Data Layer:-

Responsibility:-

- 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 users submitted solution.

How the Architechture Ensure Efficiency

1) Scalability :-
   - Each layer can be scaled independently

2) maintainability:-
   - Change in one layer do not affect others, making updates easier.

3) Performance:-
   - Clear Separation of responsibilities ensures smooth data flow and optimized system operation,

## Answer to the Question No. 15

Data flow diagram Hospital management System is used to create an overview of Hospital management without going in too much detail

*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.

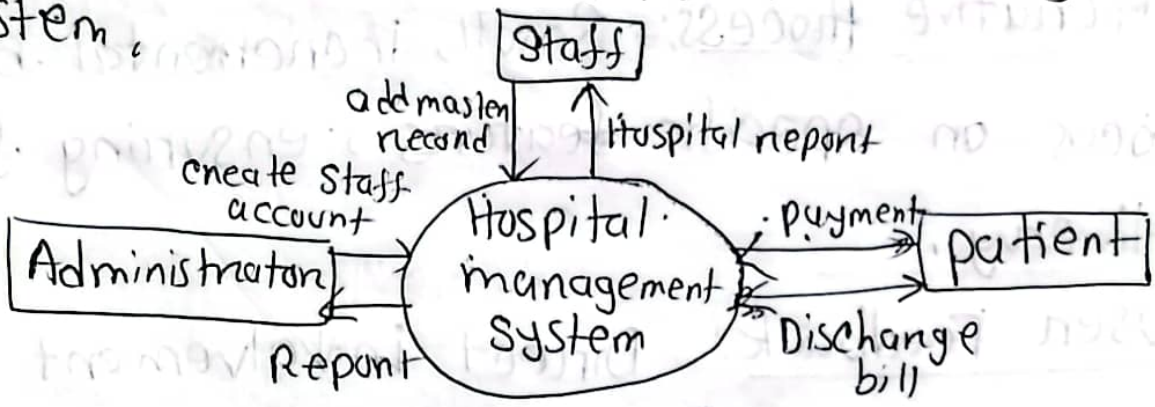
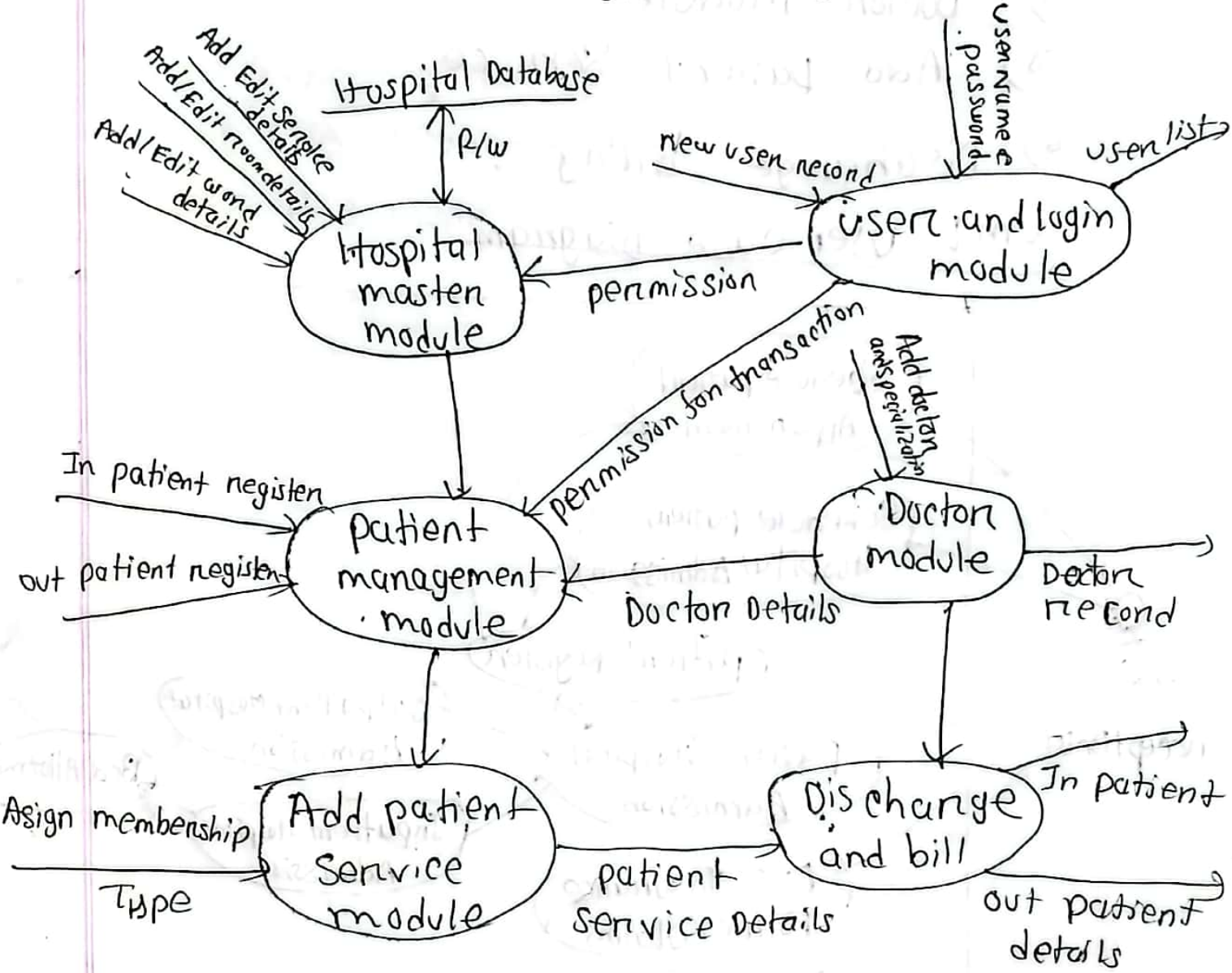



Fig:- DFD (0 Level)

* First level · Data Flow Diagram · (Level-1 DFD) on · Hospital management System.

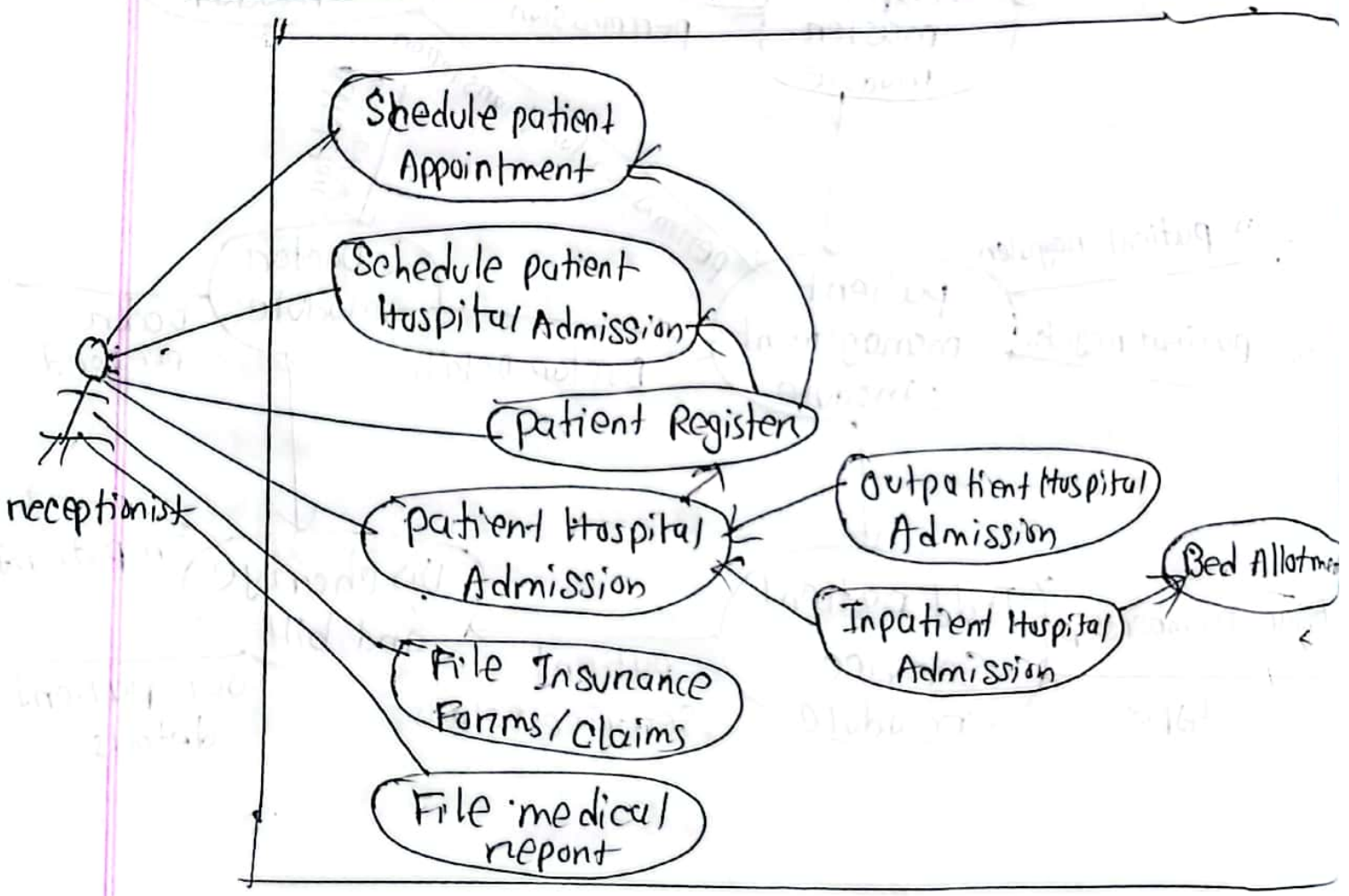
The First level DFD · (1st · level) of hospital management System Shows · more details of processing Level-1 DFD list all the major Sub process that makes the entire · System ·



The important 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 User Case Diagram



In this diagram:-

- Actor: Receptionist
- use cases:
  - Schedule Appointment.
  - Admit patient
  - Collect patient Information.
  - Allocate Bed
  - Receive payment
  - Generate Receipts
  - File insurance claim
  - File medical Report.

## Answer to the Question no:-17

Quality Assurance (QA) and Quality contro are both important for ensuring Software quality but they focus on different aspec

### what is QA?

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

Example:- creating coding standards, testi guidelines, and regular audits to ensure quality.

### What is QC?

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

the Product.

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) |
|---|---|---|
| 1) Focus | Process of development. | 1) Final Product quality. |
| 2) Goal | Prevent defects | Detect and fix defects. |
| 3) Approach | Proactive (prevention) | Reactive (correction) |
| 4) method | Reviews, audits, process improvements. | Testing, inspections, bug fixing, |
| 5) Responsibility | Entire development team. | Testing team or QA testers. |
| Example | Defining coding standards and best practices. | Running test cases to check for bugs. |
| Timing | Applied throughout the Software development lifecycle. | Applied after development during test. |

* Impediments (challenges) to QA and QC:-

1) For QA:-

- Resistance to Process changes.
- Lack of Proper documentation.
- Time constraints for Process improvements.

2) For QC:-

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

Answer to the question no: 18

Is the ~~Goo~~ Goal of QA Just to Find Bug?

Ans:-

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 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:-

1) Requirement Analysis phase:-

- Ensures that requirement are clean and

testable,

- Identifies any unclear on missing details in the requirements.
- Prevents future Problems due to misunderstood requirements.

2) Design phase:-

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

3) Development phase:-

- Ensures developers follow coding standards.
- Helps with unit testing and early error detections.
- Reduces the chances of major bugs during testing.

4) Testing phase:-

- Runs different tests like Functional, Integrati and System testing.
- Finds and reports bugs to be fixed.
- Ensures software works as intended and meets user expectations.

5) Deployment phase:-

- verifies that the software is ready to release,
- Ensures the deployment process is smoth and error free.
- Helps deliver reliable software to users.

6) maintenance phase:-

- monitors the software for issues after release,
- Ensures any updates or changes are proper tested before going live.

- keeps the software stable and functional over time,

conclusion:-

QA is important in every phase of software development. It doesn't just focus on finding bugs but ensures the entire process is effective and produces high-quality software. QA's ultimate goal is to prevent problems, improve reliability and meet user expectation.

## Answer to the question no:-19

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

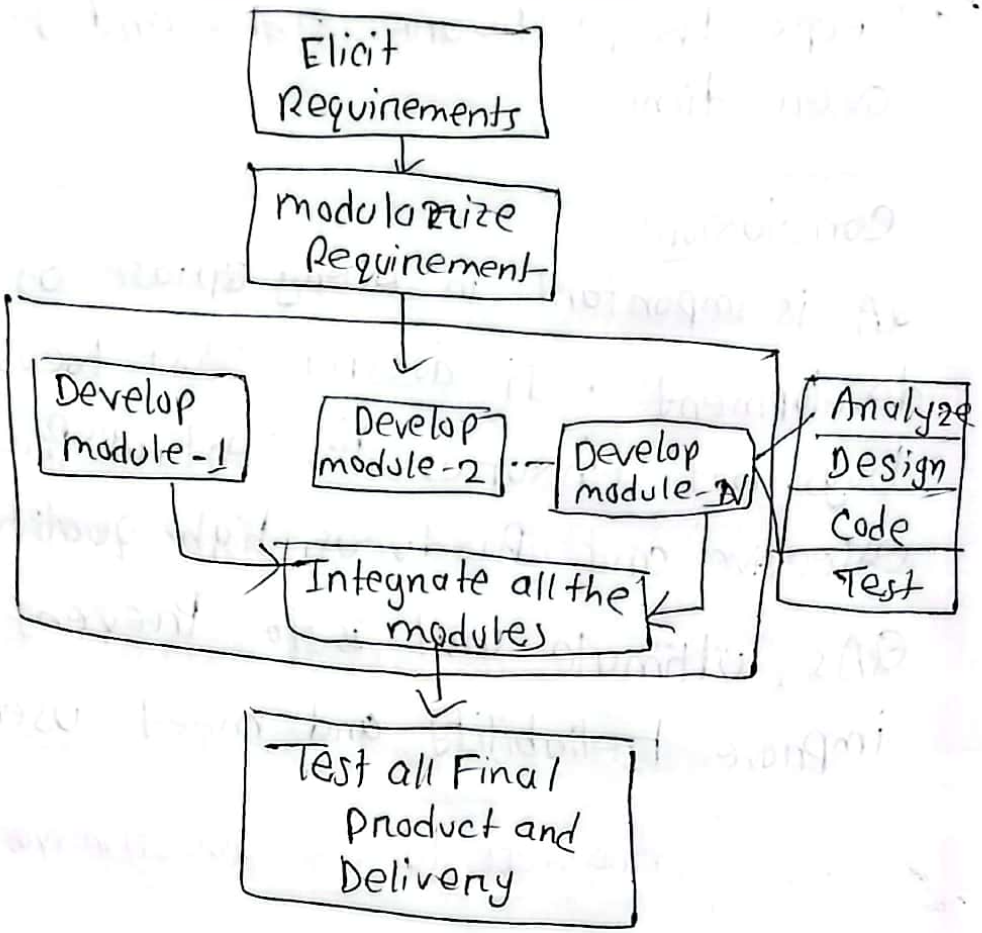



Fig:- RAD model.

key phases of RAD model

1) Requirements planning:-

- Involves gethering and underotanding user requirements
- Stakeholders collaborate to define the Project Scope and goals.

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

2) User Design :-

- A collaborative process where users and developers create prototypes.
- User provide feedback to refine the system's functionality and interface.
- outcome :- A set of working prototypes.

3) Construction :-

- Development of the functional system based on user feedback.
- Uses iterative coding, testing and integration.
- outcome :- A near-complete version of the software.

4) Cutover :-

- Final testing, user training and system deployment.

- The System is moved to Production and fine-tu for real-world use,
- outcome:- Fully functional Software delivered to users.

Principle of RAD model:-

1) User involvement:- continuous feedback from users throughout development.
2) Iterative Development:- Software is build incrementally with regular updates.
3) Prototyping:- Early version of the System are created for review and feed-back.
4) Component Reuse:- Pre-built components are used to Speed up development.

Advantage of RAD model:-

1) Fasten Delivery:- Quick iterations and Prototyping allow rapid development.

2) Flexibility:- Easily adapts to changing user requirements.

3) High user Satisfaction:- Users are actively involved, ensuring the system meets their needs.

4) Reduced Risk:- Problem are identified early through continuous feedback.

5) Better quality:- Frequent Testing ensures high-quality outcomes.

How RAD Supports Fasten 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.

Reusable component:- using existing code component accelerates developments and reduce errors.

Parallel Developments:- Team can work on different modules simultaneously.

## Answer to the question No;-20

white box testing focuses on internal structure and logic. we will apply code coverage and data coverage methods to ensure all possible execution paths are tested.

Production code (Java implementation):-

This Java Program read two integers x & y checks for different conditions and Prints appropriate 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);
}
}
}
```

Decision table:-

| Decision Condition | x input | y input | Expected output |
|---|---|---|---|
| y==0 | 5 | 0 | "y is zero" |
| x==0 | 0 | 3 | "x is zero" |
| Loop does not run (x=0) | 0 | 2 | no output |
| Number divisible by y | 4 | 2 | "2", "4" |
| Edge case; negative y | 5 | -2 | "2", "4" |
| Edge case; negative x | -3 | 2 | no output. |
| Number not divisible by y | 4 | 3 | "3" |

Each test case ensures every condition is evaluated.

## Junit Test class

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

```
import static org.junit.Assert.*;
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");
}
    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
   public void testYIsZero() {
     output.clean();
     process(5, 0);
  assertEquals(List.of("y is zero"), output);
  }

 @Test
  public void testXIsZero() {
     output.clean();
     process(0, 3);
  assertEquals(List.of("x is zero"), output);
 }
  @Test
  public void testLoopDoesNotRun() {
      output.clean();
      process(0, 2);
 assertEquals(List.of(), output);
```

```
@Test
public void testNumbersDivisibleByXY() {
    output.clear();
    process(4, 2);
    assertEquals(List.of("2", "4"), output);
}
@Test
public void testNumbersNotDivisibleByXY() {
    output.clear();
    process(4, 3);
    assertEquals(List.of("3"), output);
}
@Test
public void testEdgeCaseYNegative() {
    output.clear();
    process(5, -2);
    assertEquals(List.of("2", "4"), output);
}
@Test
public void testEdgeCaseXNegative() {
    output.clear();
    process(-3, 2);
    assertEquals(List.of(), output);
}
}
```

## Answer to the question no: 21

Here is a Junit 4 test code that demonstrates

- Exception handling
- Setup Function (@Before)
- Timeout rule (@Test(timeout = ...))

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 ArithmeticException("/by zero");
    }
    return a/b;
}
// Simulating a Long running operation.
public void LongRunningOperation(){
```

```
try{
  Thread.sleep(500); // 500ms Sleep
}
  catch(InterruptedException e){
        e.printStackTrace();
}}
```

Junit 4 Test Code (CalculatorTest.java)

```
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: setup function, run before each test.
@before.
    public void setup(){
    calculator = new Calculator();
    }
    // Step 2: Testing for exceptions (divide by zero)
```

```
@Rule
public ExpectedException exceptionRule =
                    ExpectedException.none();
@Test
  public void testDivisionByZero() {
 exceptionRule.expect(ArithmeticException.class);
  exceptionRule.expectMessage("/by zero");
   calculator.divide(10, 0);
  }
@Test(timeout = 1000)
    Public void testLongRunningOperation() {
      calculator.longRunningOperation();
    }
    @Test
     pvblic void testAddition() {
   assertEquals(15, calculator.add(10, 5));
  }}
```

* Here @Before is declared with setup() function which is a built-in function of Junit