Answer to the Q: 1

Scrum-based software development project:

- Product Backlog Breakdown:

1. "As user, I want to log in securely so that I can access my account."

→ Implement login page UI.
→ Set up authentication mechanism (e.g. email/password, social login)
→ Development password encryption and storage
→ Add multi-factor authentication (MFA).
→ Conduct security testing

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

→ Design the search UI.
→ Implement category filters.
→ Developm backend search functionality
→ optimize database queries for search efficiency.
→ Conduct usability testing.

• Prioritization in sprint planning:

The team can prioritize based on the buisness value (e.g. login is crucial for user access) and ~~the~~ have, should-have, could-have, coonit have) the team can catagorize ~~tasks~~ to ensure the most critical ones are completed first. Dependencies and team capacity should also be considered.

• Scrum Board Tracking:

Takes will be tracked on the scrum board with colum such as.

- To Do: Tasks not yet started (e.g, design UI, backend development)
- In progress: Tasks being actively worked on (e.g implementing authentication).
- Done: Completed and tested tasks ready for development.

=o=

Ans to the 9:2

Considering the project's high risk and evolving requirements, Agile methodology would be the most suitable choice Here's a comparison of methodologies:

Spiral model:

- Strengths: Focuses on risk management through iterative prototyping.
- Weakness: Can be costly and time-consuming due to repeated cycles.

Agile methodology:

- Strengths: Provides flexibility to adapt to changing requirements, ensures frequent feedback from the client, and delivers value incrementally.
- Weakness: Requires strong collaboration and discipline within the team

Extreme programming (XP):

- Strengths: Promotes continous feedback, test-driven development, and rapid releases.
- Weakness: May not suitable for large

teams or complex projects requiring detailed planning.

Conclusion: Agile is the best fit due to its adaptability, iterative approch, and cost effictiveness in managing evolving requirement while delivering functional increments quickly.

=o=

Answer the 9: 3

Comparing and constrasting waterfall, Agile, Extrem and Spiral models for project A and Project B:

- Project A (Well-defined requirement, strict deadline):

- Best Fit: Waterfall model
  - Predictibility: High, as all requirements are well-defined and planned in advance
  - Customer Collabration: Limited, since the project follows a sequential flow.
  - Risk Management: Minimal flexibility changes are costly and difficult to incroporate.

- Alternative Fit: V-model (Verification and validation) - Emphasizes testing alongside development.

- Project B (Evolving requirements, uncertain timeline, continuous feedback):

- **Best Fit: Agile Model**
  - **Predictibility:** Lower, but adaptable to change and evolving requirements.
  - **Custome collabration:** High, with frequent interactions and feedback loops.
  - **Risk management:** Incremental delivery reduces risks by addressing change iteratively.

- **Alternative Fit: Spiral Model** – Combines iterative development with strong risk management assessment at each phase.

Ans to the Q: 4

Principles of software Engineering Ethics and ACM/IEEE code to Ethics:

- Principles of software Ethics:
  - public interest and welfare.
  - Integrity and honesty in software practices.
  - Confidentiality of sensitive information
  - Continous professional development.
  - Respect for intellectual property rights.
- ACM/IEEE Code of Ethics Guides:
  - Promotes fair decision-making considering user safety, data privecy, and professinal conduct.
  - Encourages honesty in reporting project limitations and ensuring software reliability.
  - Emphasizes respect for public and clint's interests.

Ans to the Q: 5

functional and Non-Functional requirement for the Airport Reservation system:

Functional Requirements:

1) User registration and login.
2) Flight search and booking.
3) Payment processing system.
4) Seat selection and reservation management.
5) Notification system for flight status updates.

No - functional Requirements:

1) System performance (response time under 2 seconds)
2) Security (data encription and secure transactions).

3) Scalibility (handling large numbers of users):

4) Usability (intutive UI for diverse users)

5) Maintainability (easy to update and modify)

=oz

Ans to the Q: 6

V-model of testing phase Explantion:

The V-model (Verification and Validation model) is software development methodology that emphasizes rigorous testing by aligning development stages with corresponding testing phases.

Key Phases of the V-model:

1) Requirement Analysis → Acceptance Testing: Ensures the system meets buisness needs.

2) System Design → System Testing: Validates complete system functionality.

3) Architectural Design → Integration Testing: Tests interactions between modules.

4) Module Design → Unit Testing: Verifies individual components.

CamScanner

Advantage of the V-model:

- Clearly defined relationships between development and testing activities.
- Early defect detection due to the parallel planning.
- Suitable for projects with well defined requirements.

Disadvantages:

- Less flexibility for change once development starts.
- Costly if changes occur late in the process.

=o=

Ans to the Q: 7

Prototype Development process in Software Engineering:

Key stage in Prototype Development:

1) Requirement Gathering: Identifying initial software needs.

2) Quick Design: Creating a basic model to capture user requirements.

3) Prototype Construction: Developing a working model with limited features.

4) User Evaluation: Gathering feedback for improvements.

5) Refinement: Enhancing the prototype based on feedback until requirements are satisfied.

Benefits of Phototyping:

1) Improved User feedback: Users can

visualization and intraction with the prototype early.

2) Risk Reduction: Identifies potential issues before full-scale development

3) Iterative Development: Allows flexiblity in refining system functionality.

challenges of prototypeing:

- Soope creep due to continous refinement
- Higher initial development costs for building multiple versions.

= o =

Ans to the Q:8

Process improvement cycle:

The process improvement cycle software engeneering focuses on continously improving software development processes to increase efficiency, enhance quality, and reduce errors. It follows an iterative approch consisting of four key stages:

1) Plan:- In this stage, software teams identify areas of improvement by analyzing existing processes. They set clear goals and define ~~stages~~. strategies for enhancements.

2) Do:- The proposed improvements are implemented in small-scale trails. Changes may include new methodologies, tools, or coading practices.

3) Check:- The outcomes of the implemented change are assessed using various metrics to determine thier effectiveness.

4) Act :- If the changes yield positive results they are standardized and incorporated into the software development lifecycle. If not, fourther refinements are made.

Commonly used process Metrics:

1) Defect Density :- Measure the number of defects per unit of code (e.g. per 1000 lines of code). A lower defect density indicates better software quality.

2) Cycle time :- The total time required to complete a development cycle from start to finish. It helps in identifying delays and inefficiencies.

3) Customer Satisfaction Score (CSAT) :- Measure user feedback on software performance, usability, and reliability

4) Mean Time to Failure (MTTF) :- The average time before a system fails

reflacting the software's stability and reliability.

5) Productivity Metrics: Includes factors like lines of code written per hour or completed user stories per sprint, helping in evaluating developer efficency.

Thes matrics help software team identify weakness make data drive decision, and enhance the overall software development process.

z o z

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Question : 9

Explain the software Engineering Institute Capability Maturity Model (SEI CMM) and its five levels of Cabability and maturity. Analyze how each level contributes to improving the software development process and organizational Performance.

Answer :

The Capability Maturity Model (CMM), developed by the software Engineering Institute (SEI), is a structured approach to evaluating and improving software development processes. By categorizing process maturity into five levels, it helps organizations move chaotic and inconsistent practices to streamlined, efficient, and continuously improving operations.

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

CamScanner

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

The Five levels of CMM and their impact :

1. Level 1 : Initial (Unpredictable)
   - Processes are ad hoc, dependent on individuals.
   - Outcomes are inconsistent, highlighting the need for structure.

2. Level 2 : Repeatable (Basic Control)
   - Basic project management practices are established.
   - Ensures repeatable success and reduces project risks.

3. Level 3 : Defined (Standandized)
   - Processes are documented and standardized across the organization.
   - Improves collaboration and consistency.

4. Level 4 : Managed (Measured)
   - Metrics are used to monitor and control processes
   - Enables predictable performance through data-driven decisions.

CamScanner

Date: / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme:

⑤ Level 5: Optimizing (Continuous Improvement)

- Focus on innovation, defect prevention, and adaptability.
- Processes evolve to stay competitive and effecient.

How each level Drives process and Performance Improvements:

1. Starting point:- At level 1, organizations recognize the need for structured processes, without formal practices, success depends on individual skills, making outcomes unreliable

2. Building foundations: Level 2 introduces process discipline.

3. Scaling Efficieney: At level 3, Standardization removes inconsistencies,

4. Achieving Precision: Level 4, focuses on measurement,

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

5. Sustaining Excellence : At level 5, innovation takes center stage.

Why CMM Matters for Organizations :

i. Predictability :- Higher maturity levels make project outcomes more reliable and consistent.

ii. Quality Enhancement : Defect prevention and process control lead to superior software products.

iii. Operational Efficiency : Standardized and optimized processes reduce waste and improve productivity.

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

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

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme :

Question 10:

Describe the core principles of agile software development methods. Analyze how these principles are applied in different software development environments, and assess the benefits and challanges of using agile methods in various project types and organizational settings.

Answer:

An Agile software process is designed to handle the unpredictability inherent in most software projects. It recognizes that requirements and customer priorities can change rapidly, and it is difficulty to predict the necessary design before construction begins.

Agile software development is based on the Agile Manifesto, which outlines key principles to enhance collaboration, adaptability, and customemer satisfaction.

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

The principles focus on delivering high-quality software in a flexible, iterative, and team-oriented manner. Below are the core principle :

(1) Customer collaboration Over Contract Negotiation : Engage customers throughout the development process to ensure the product meets their needs.

(2) Working Software over Comprehensive Documentation : Prioritize delivering functional software over extensive intial documentation.

(3) Responsing to change Over following a plan : Embrace change, even late in development, to improvement the product

(4) Individuals and Interactions over Processes and Tools : Foster collaboration, communication and teamwork as the foundation of success

(5) Frequent Delivery of valee : Deliver working software in small, freqeent increments.

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme :

⑥ Sustainable Development : Maintain a consistent work pace to avoid burnout.

⑦ Continuous Feedback and Improvement : Use feedback loops to refine both the product and the development process.

⑧ Technical Excellence and Good Design : Focus on high-quality code and maintainability.

⑨ Self-Organizing Teams : Empower teams to take ownership of tasks and make decisions collaboratory.

⑩ Simplicity : Maximize work done by keeping processes and solutions straightforward.

CamScanner

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Benefits of Agile :

i. Increased Customer satisfaction
ii. Improved Quality
iii. Increased flexibility
iv. Improved Team Morale
v. Reduced Time-to-Market

Challanges of Agile :

i. Difficulty in estimating project timelines
ii. Resistance to change
iii. Maintaining focus.
iv. Finding and retaining skilled personel.

In conclusion, agile methodologies offer a flexible and effective approach to software development.

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme :

Question 11 :

Draw the release cycle of Extreme Programming (XP) and explain the influential programming practices :

Answer :

The Extreme Programming (XP) release cycle focus on frequent, small releases, ensuring that the software is always in a deployable state and customer feedback is continuously integrated.

Here's how the release cycle typically works :

Extreme programming (XP) Release Cycle :

i. Planning : Initial and iteration planning define features and deliverables.

ii. Development Iteration : Code is developed in short iterations (1-2 weeks)

iii. Continuous Testing : Automated tests ensure functionality and quality.

iv. Customer Feedback : Customer reviews each iteration and provides input.

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

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

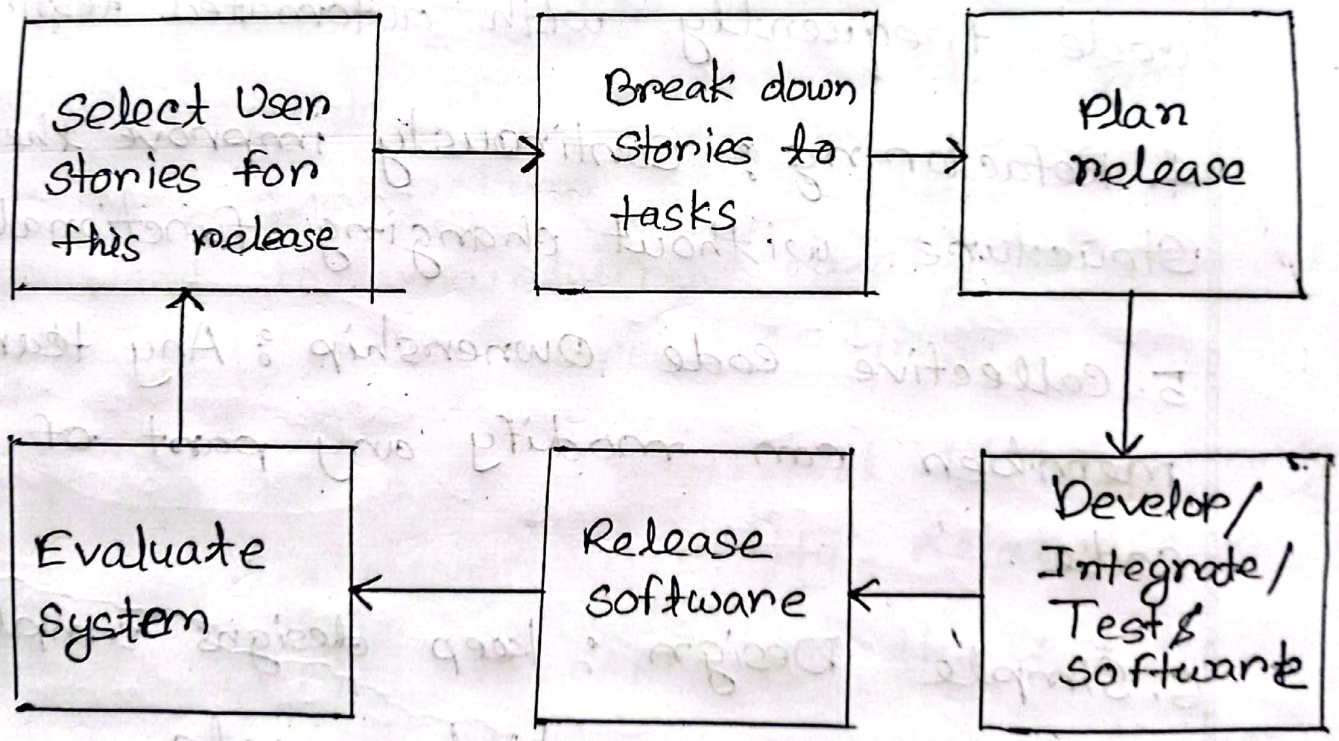


Diagram of release cycle of XP

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme :

Influential XP Programming Practices :

1. Pair Programming : Two developers collaborate on the same code to improve quality.

2. Test-Driven Development (TDD) : Write tests before code to ensure correctness.

3. Continuous Integration (CI) : Integrate code frequently with automated testing.

4. Refactoring : Continously improve the code structure without changing functionality.

5. Collective code Ownership : Any team member can modify any part of the code.

6. Simple Design : keep designs simple, focusing on immediate needs.

7. Customer Involvement : Customers provide continuous feedback throughout development.

Subject / Theme :

Date : / /

☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Question - 12

A local Library wants to create a digital system to manage its operations. The system will track books, members and borrowing activities. Each book has attributes like title, author, TSBN and genre. Members have attributes such as name, membership ID and contact details. When a member borrows a book, the system records the borrowing date, return due date and return states. The library also want to maintain a catalog of overdue books and their respective fines.

Using the seenario of a digital library management system, design an Entity - Relationship Diagram (ERD) to represent the entities (e.g. books, members, borrowing activities) and their relationships. clearly explain the attributes of each entity and how they are interconnected.

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Answer :

To design an Entity-Relationship Diagram (ERD) 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 :

1. Entities and Attributes :

i. Book : Track books with attributes like Book_ID , Title , Author, ISBN, Genre and Availability_States.

ii. Member : Records members with Member_ID, Name, Contact_Details, and Membership_status.

iii. Borrowing Activity : Manages borrowing details, including Borrow-ID (PK), Borrowing_Date, Return_Due_Date, Return_states, and Fine_Account.

iv. Overdue Book : Logs overdue books with Overdue_ID (PK), Book_ID (FK), Member_ID (FK),

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Fine_Amount, and Overdue_Days.

2. Relationships :

i. Books ↔ Borrowing Activity : One book can have many borrowing activities (1 to many).

ii. Members ↔ Borrowing Activity : One member can borrow multiple books (1-to-many).

iii. Borrowing Activity ↔ Overdue Book : A borrowing activity may lead to one overdue record (1-to-1, optional)

3. key features :

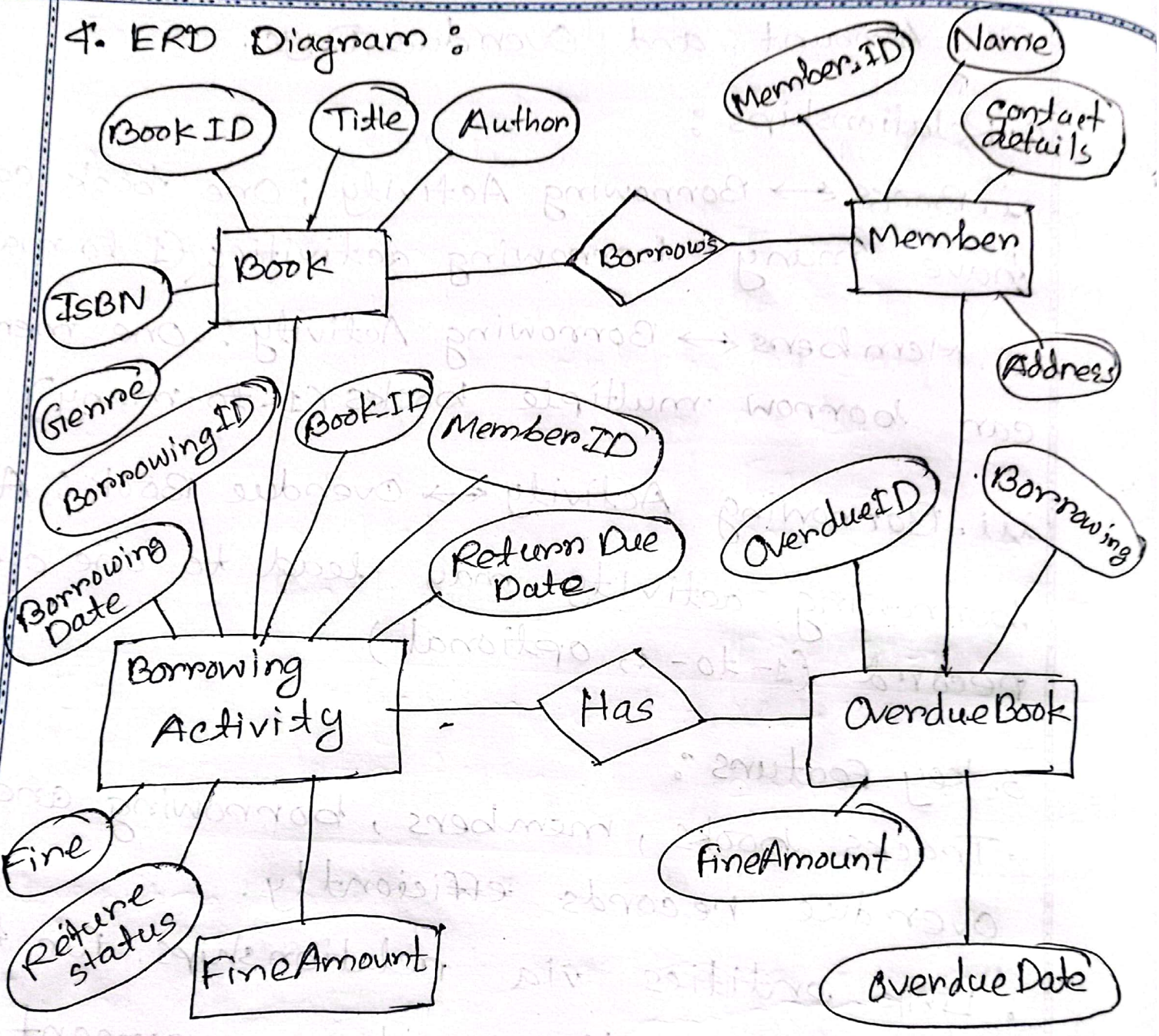
. Tracks books, members, borrowing and overdue records efficiently.

. Link entities via relationships for better data organization and management.

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

4. ERD Diagram :



Summary :

The ERD captures the relationships and key attributes needed to manage the digital library's operations. It effectively tracks books, and other essential components.

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Question : 13

What is called Testing ? Differentiate between validation an verification.

Answer :

Testing in software engineering is the process of evaluating a software system or component to find defects or errors. It's a critical step in the software development lifecycle to ensure the quality, reliability, and functionality of the final product.

"Testing" refers to the process of systematically evaluating a product or system to ensure it functions as intended and meets specified requirements, while "validation" checks if the product meets the actual needs of the user, wheres "verification" confirms if the product is built according to the defined specifications, essentially ensuring its "built right" rather than "right for the user."

Date: / /
☐Sat ☐Sun ☐Mon ☐Tue ☐Wed ☐Thu ☐Fri

bject / Theme:

Difference between Validation and Verification.

Aspect	Validation	Verification
Definition	Ensures the product meets the user's needs and expectations.	Ensures the product is built according to the specified requirements.
Focus	Focuses on the end product and its functionality.	Focuses on the processes and intermediate work products.
Type of Testing	Verification is the static testing.	Validation is dynamic testing.
Execution	It does not include the execution of the code.	It includes the execution of the code.
Timing	It comes before validation.	It comes after verification.
Performance	Verification finds about 50 to 60% of the defects	Validation finds about 20 to 30% of the defects.

CamScanner

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Question : 14

Design a layered architecture model for an online judge system, identifying the key layers (e.g., presentation, application, business logic, and data). Explain the responsibilities of each layer and analyze how this architecture ensures scalability, maintainabity, and efficient performance.

Answer :

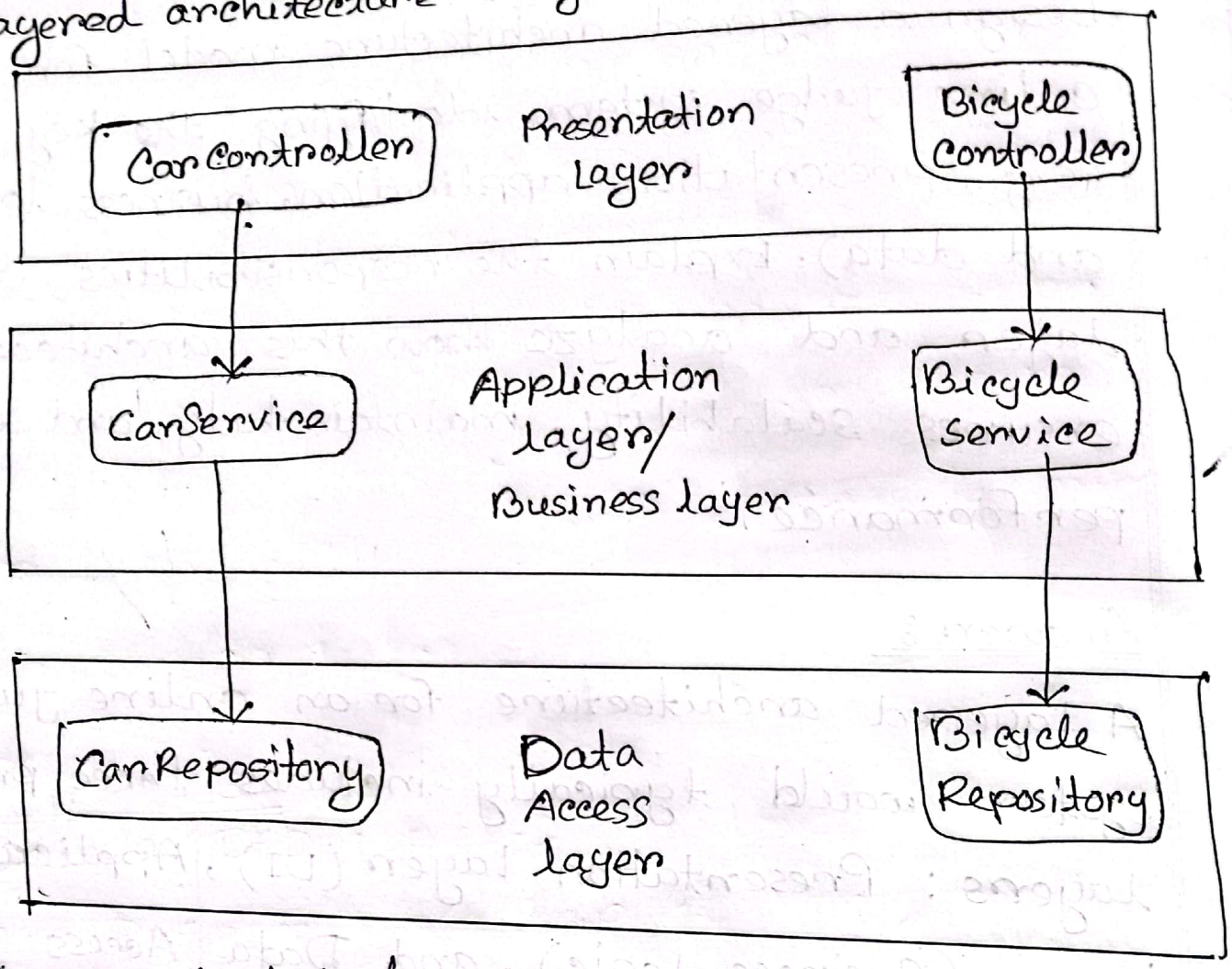
A layered architecture for an online judge system would typically includes three primary layers : Presentation Layer (UI), Application Layer (Business Logic), and Data Access Layer (Database Interaction), with each layer having well-defined responsibilities and interaction only with the layer directly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

CamScanner

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme :

Layered architecture diagram :



Layered Architecture for online Judge system :

1. Presentation Layer :
   i. Handles users interactions.
   ii. Technologies : React, Angular, on mobile interfaces.

Subject / Theme:

Date: / /

☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

2. Application Layer:

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

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

3. Data Layer:

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

ii. Technologies: SQL and caching

Benefits:

i. Scalability: Independent scaling of layers and load balancing

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

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

CamScanner

Date: / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme:

Conclusion:

This layered architecture organizes an online Judge system into distinct, independent layers Each layer handles specific responsibilities, enabling the system to process user requests efficiently, scale to accomodate growing traffic, and be easily maintained or upgraded.

Question: 15

Draw a DFD (Level-0 and Level-1) and UML case Diagram for a Hospital Management System.

A hospital management system is a large system that includes several subsystems or modules that provide various functions. Your UML case diagram example should show actors and use cases for a hospital's reception.

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Answer :

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.

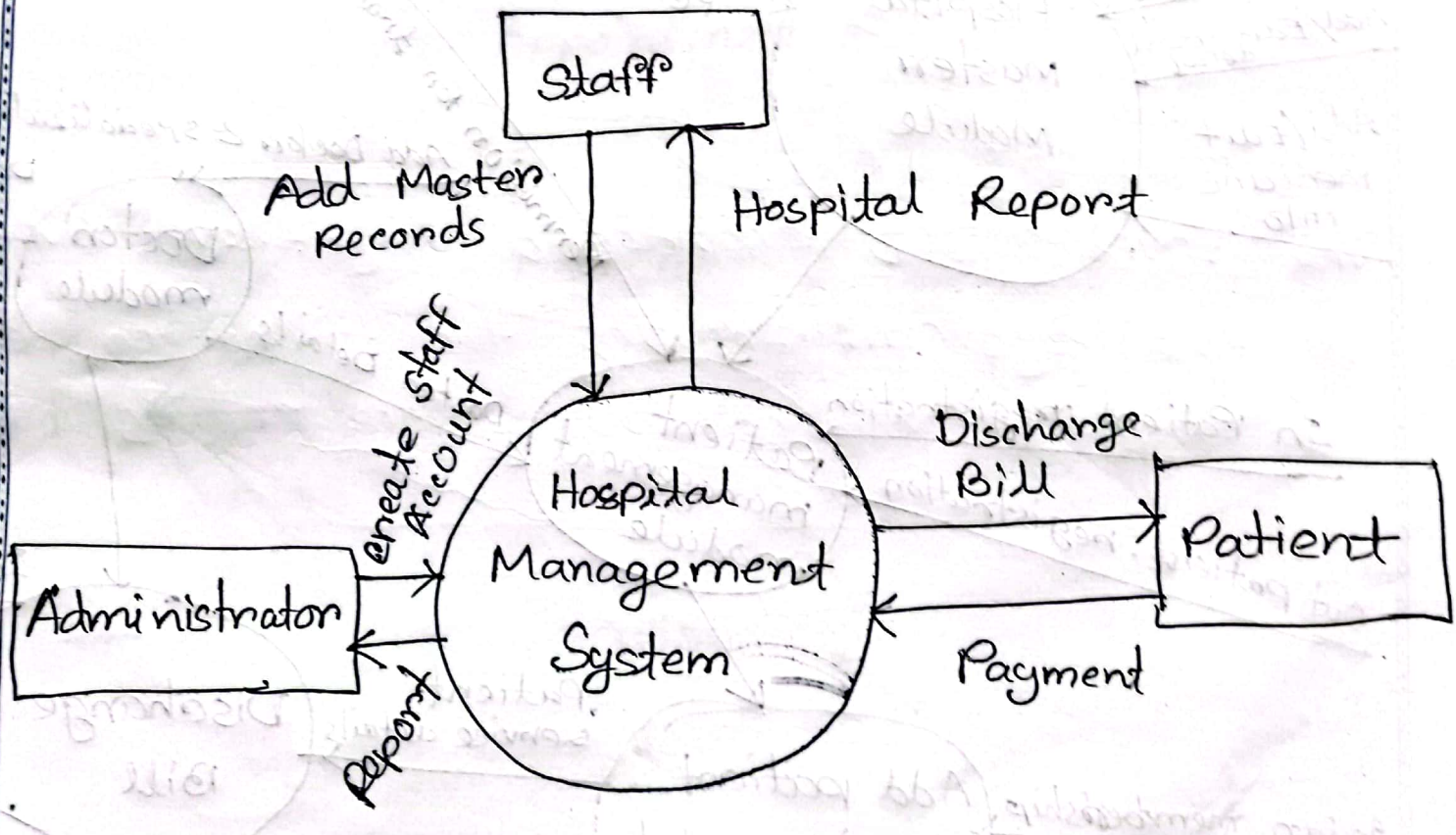


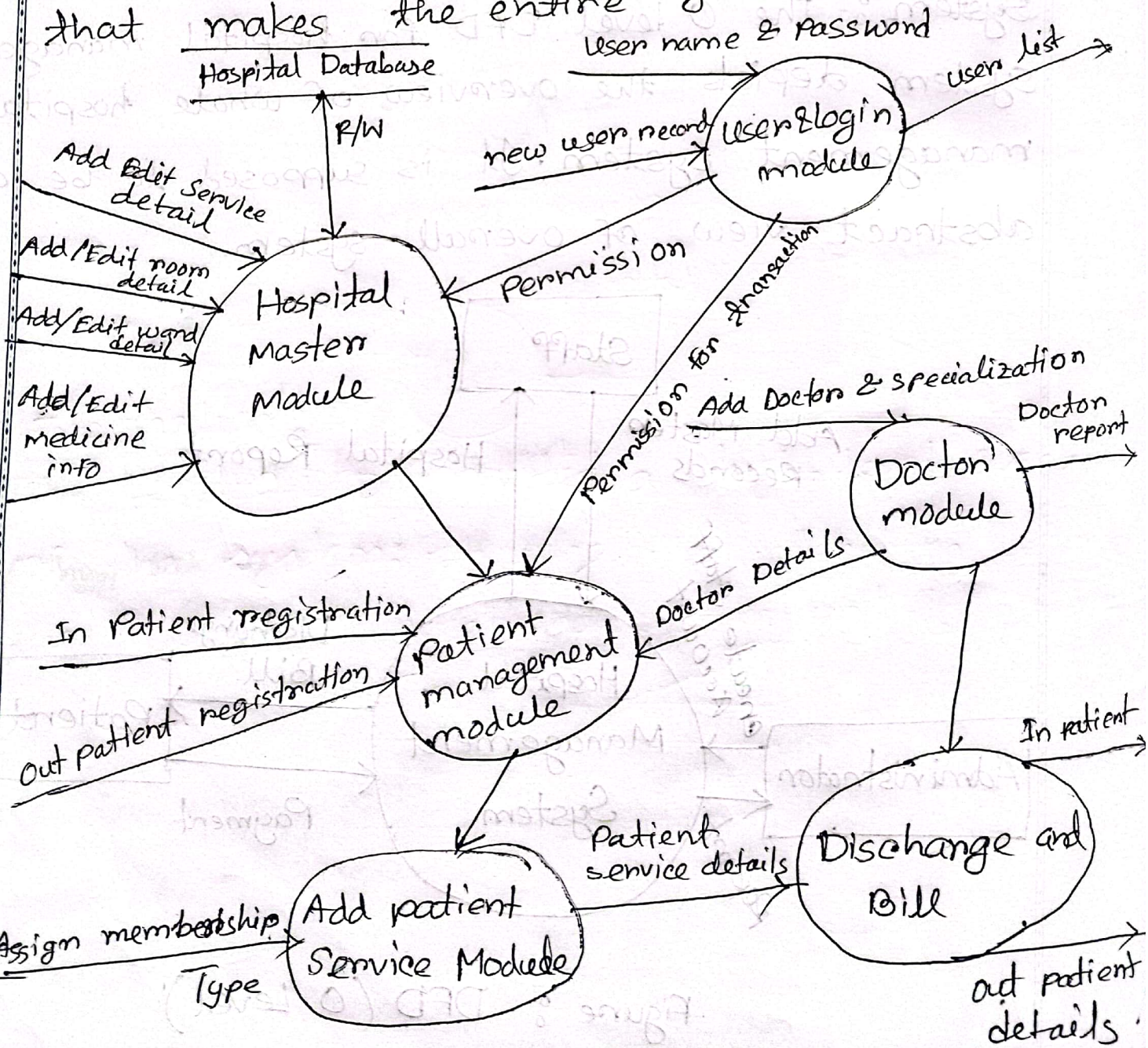
Figure : DFD (0 Level)

Date: / /
☐Sat ☐Sun ☐Mon ☐Tue ☐Wed ☐Thu ☐Fri

Subject / Theme:

First level Data flow Diagram (Level 1 DFD) of Hospital Management System:

The first level DFD (1st level) of Hospital management system shows more details of processing Level 1 DFD list all the major sub processes that makes the entire system.



Subject / Theme :

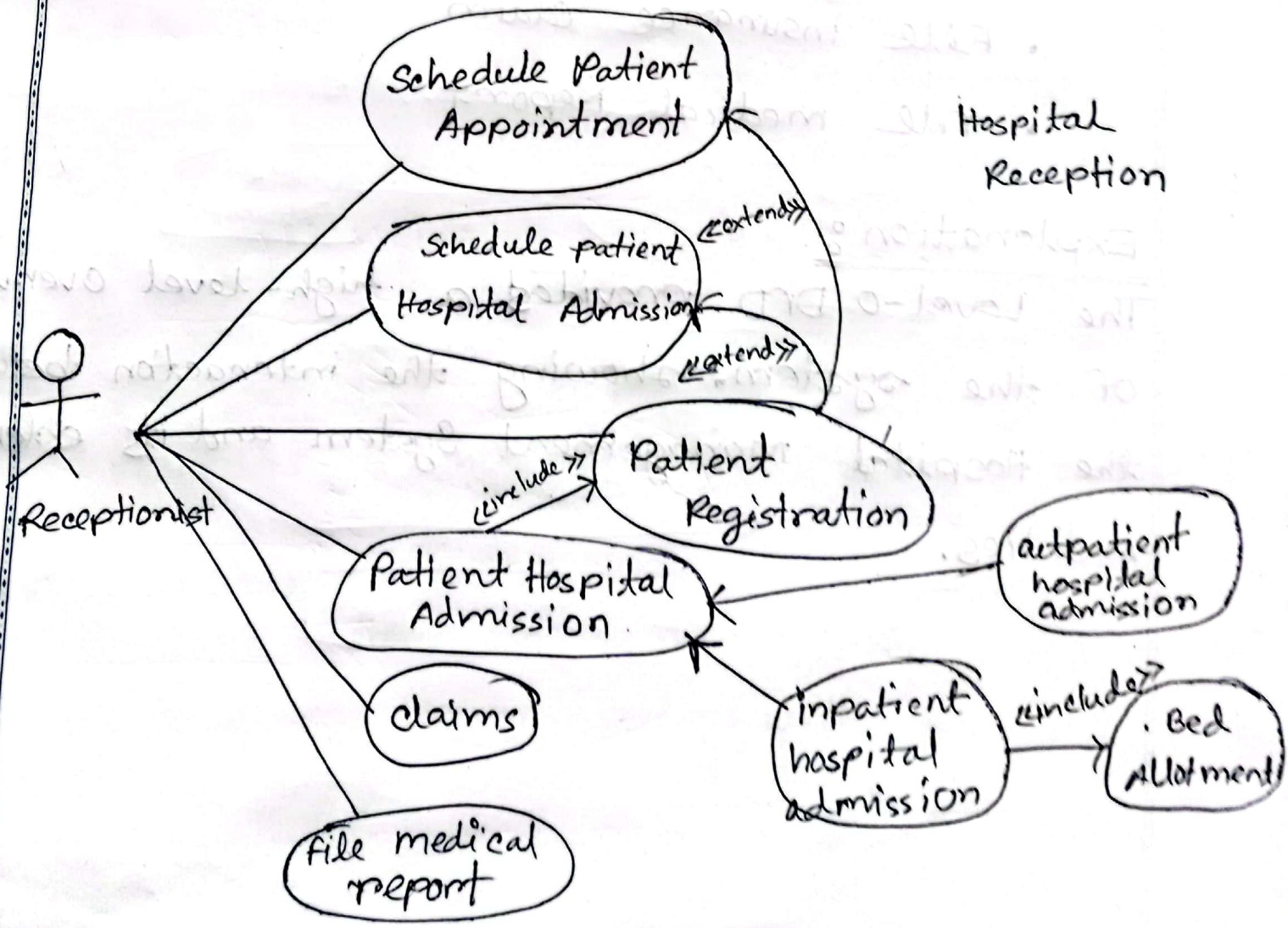
Date : / /

☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

The important process to be carried out are :

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

UML use case Diagram :



Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme :

In this diagram :

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

Explanation :

The Level-0 DFD provided a high-level overview of the system, showing the interaction between the Hospital management System and its external entities.

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Answer to the Q: No:- 16

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

Identified classes from the Sequence Diagram:

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

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

3. ValidateUser
   - Methods: checkCredentails (userID, password)

4. Database
   - Attributes: userID, password, classList
   - Methods: fetchUserDetails (), returnclassList()

Relationships:

- Student interacts with loginScreen (one to one)
- LoginScreen communicates with ValidateUser (one to one).

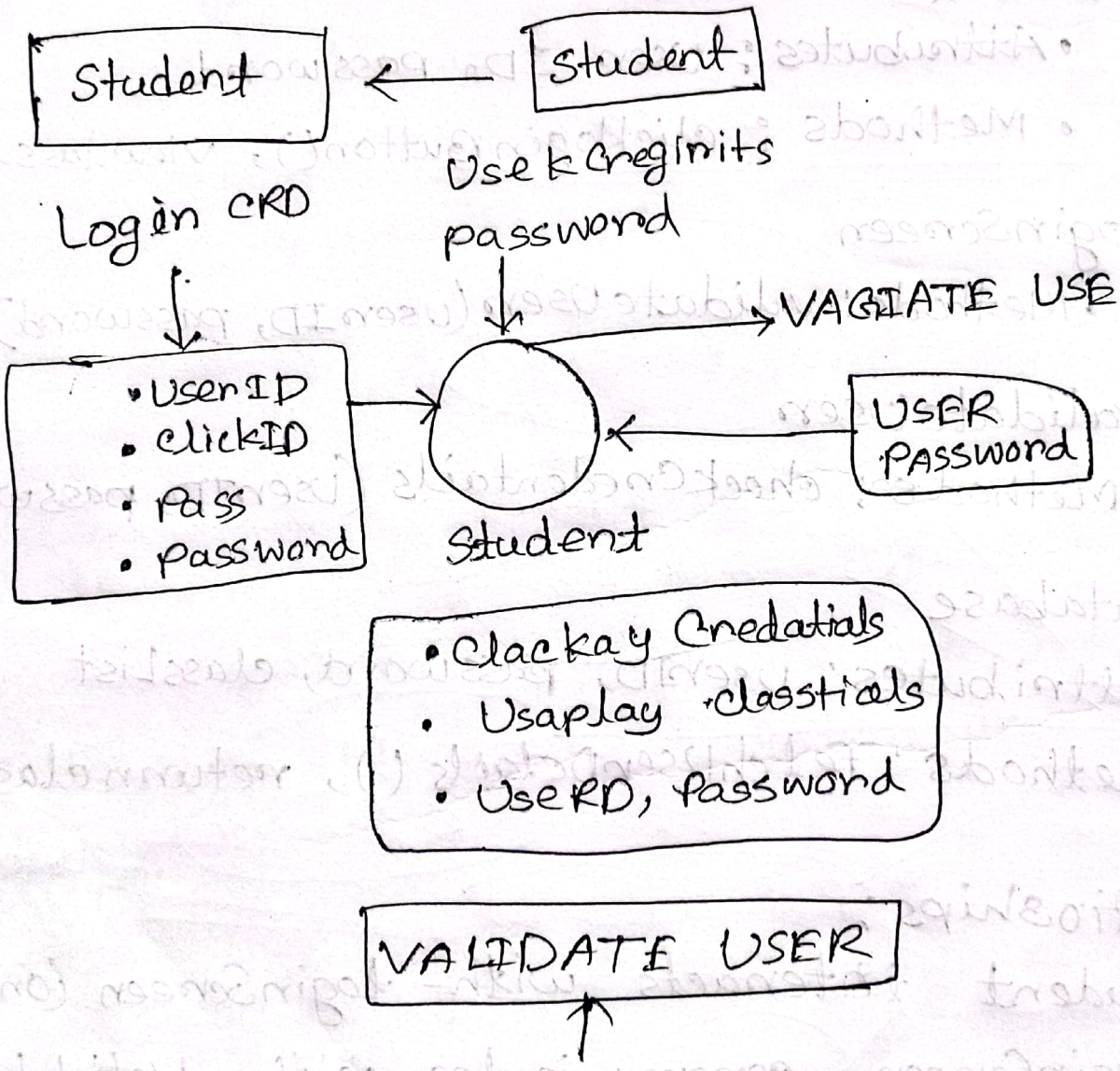
CamScanner

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme :

- ValidateUser fetches data from Database (one-to one).
- Database stores user details and class lists

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



Subject / Theme :

Date : / /

☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

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

CamScanner

Subject / Theme:

Date: / /
☐Sat ☐Sun ☐Mon ☐Tue ☐Wed ☐Thu ☐Fri

Answer to the Question No: 17

Quality Assurance (QA) vs. Quality control (Qc) - Explanation, Differences, and Impediments:

Explanatory Discription:

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

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

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

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

CamScanner

Date: / /
☐Sat ☐Sun ☐Mon ☐Tue ☐Wed ☐Thu ☐Fri

Subject / Theme:

Difference between QA and QC

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

Impediments to QA and QC:

For QA:

i. Lack of Standardized Processes
ii. Poor Management support
iii. Resistance to change
iv. Limited Resources
v. Time Constraints.

CamScanner

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

For QC :

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

Conclusion :

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

CamScanner

Date: / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme:

Answer to the Question No: 18

Role of Quality Assurance (QA) in Software Development Life Cycle (SDLC):

The goal of Quality Assurance (QA) is not just to find bugs early but to ensure that quality is built into the software development process from the beginning. It focuses on preventing defects, improving processes, and ensuring that the final product meets customer expectations.

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

CamScanner

Subject / Theme :

Date : / /

☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

QA Role at Each Phase of SDLC

SDLC Phase	Role of Quality Assurance (QA)
1. Requirement Analysis	- Ensure clarity, completeness, and feasibility of requirements. - Conduct requirement reviews to prevent ambiguity. - Define acceptance criteria and quality standards early.
2. Planning	- Identify risks and mitigation strategies. - Define QA objectives, scope and test strategy - Plan resources, timelines, and test environments.
3. design	- Review system architecture and design for quality aspects like security, scalability and performance. - Identify potential failure points early.

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme :

In conclusion, QA is not just about finding and fixing bugs - it ensures the entire process is structured to produce a high-quality product. By embedding QA at every phase of SDLC, organizations can reduce costs, enhance reliability, and deliver better user experiences.

Answer to the Question No : (19)

Rapid Application Development (RAD) Model in Software Engineering :

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

models like the Waterfall model.

key phases of the RAD Model :

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

Principle of the RAD Model :

i. Prototyping Over Planning
ii. Active User Involvement
iii. Iterative and Incremental Development
iv. Component - Based Development
v. Quick Delivery.

Advantages of the RAD Model :

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

CamScanner

Date: / /
☐Sat ☐Sun ☐Mon ☐Tue ☐Wed ☐Thu ☐Fri

Subject / Theme:

How the RAD Model Ensures Faster Delivery while Maintaining Quality and User satisfaction:

i. speed Through Prototyping
ii. Continuous User feedback
iii. Iterative Testing Ensures Quality
iv. Focus on Reusable Components.

The RAD model accelerates software development without compromising quality by emphasizing prototyping, user feedback, and reusability. It is ideal for projects that require fast iterations and evolving requirements, ensuring that the final product is not only delivered quickly but also meets high-quality standards and user satisfaction.

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Answer to the Question NO: 20

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

Test Cases Table

Decision Statement	x Input	y Input	Expected Output
If (y == 0)	5	0	"y is zero"
else if (x == 0)	0	3	"x is zero"
For loop does not run	0	2	"" (No output)
For loop prints numbers divisible by y	4	2	"2", "4"
For loop prints numbers divisible by y	4	3	"3"
For loop with negative y	5	-2	"2", "4"
Negative x, loop does not execute	-3	2	"" (No output)

Date: / /
☐Sat ☐Sun ☐Mon ☐Tue ☐Wed ☐Thu ☐Fri

Subject / Theme:

JUnit Test class in Java

```
import static org.junit.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import java.util.ArrayList;
import java.util.List;

class DecisionTest {
    private List<String> output = new ArrayList<>();
    private void println(String message){
    output.add(message);
    }
private void process(int x, int y){
    if (y==0) {
      println("y is zero").
    } else if (x==0){
      println("x is zero");
    } else {
      for (int i=1; i<=x; i++){
        if (i%y==0){
          println(string.valueOf(i));
    } }
} }
```

Subject / Theme :

Date : / /

☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

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

@Test
void testLoopDoesNotRun() {
    output.clear();
    process(0, 2);
    assertTrue(output.isEmpty());
}

@Test
void testEdgeCaseXNegative() {
    output.clear();
    process(-3, 2);
    assertTrue(output.isEmpty());
}
}
```

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme :

Explanation of the JUnit Test class :

i. Simulating

ii. Implementing the original logic

iii. JUnit Test Methods

In conclusion, This JUnit test class effectively tests all possible branches in the given Java program. White box testing principles are applied to ensure complete code coverage and data coverage. This implementation is structured similarly to the python version, ensuring easy verification of results.

CamScanner

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Question 21 :

Black Box Unit testing is earlier and more precise than Black Box System testing - it can find errors very easily, even before the entire first version is finished. Now, consider the production codes that need function testing. Suppose you have JUnit 4 API in your IDE and you are said to develop test codes for these production codes showing the application of Excep-tion, Setup function and Timeout Rule How do you solve it ?

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme :

Answer to the Question NO: 21

JUnit 4 Black Box Unit Testing Approach :

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

To test exception handling, setup function, and time out rule, we develop JUnit 4 test cases for the following production code.

Production code (Calculator.java)

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

```
public double sqrt (double number){
if (number <0) throw new IllegalArgumentException
        ("Negative number");
try { Thread.sleep (500); } catch (InterruptedException e) { }
}
public void storeInMemory (int value) {
    this.memory = value;
}
public int getMemory () {
    return this.memory;
}
}
```

In JUNit 4, we can apply :

i. Exception Handling

ii. Setup function

iii. Timeout Rule

Date: / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

Subject / Theme:

JUnit 4 Test class for Function Testing :

```
import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.Timeout;

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

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

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

Subject / Theme :

Date : / /
☐ Sat ☐ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri

```
@Test
public void testsqrtComputationTime()
{
    double result = calculator.sqrt(16);
    assertEquals(4.0, result, 0.01);
}

@Test
public void testMemoryFunction(){
    assertEquals(10, calculator.getMemory();
}
}
```

Explanation of the JUnit 4 test class:

i. Setup function

ii. Exception Handling Test

iii. Timeout Rule

iv. Function Execution Test

v. Setup verification

Date: / /
☐Sat ☐Sun ☐Mon ☐Tue ☐Wed ☐Thu ☐Fri

Subject / Theme:

In conclusion,

This JUnit 4 test class effectively applies exception testing, setup functions, and timeouts. The black box approach ensures function correctness without accessing internal logic. These techniques help detect errors early before full system testing. This method ensures early error detection and efficient function validation.