Subject / Theme:

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

Software Engineering
ICT - 3209
Assignment

Questions:

1. In a Scrum-based software development project, the Product Owner has defined the following user stories for an e-commerce application:

- As a user, I want to log in securely so that I can access my account.
- As a user, I want to search for products by category to items easily.

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

II. Describe how the development team can prioritize these user stories during a sprint planning meeting, considering value to the customer and technical feasibility.

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

Subject / Theme:

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

Answer: (i)

User Story 1: Log in securely

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

Tasks:

1. Design the login page UI (username/password input fields, buttons).
2. Implement client-side validation for login inputs.
3. Develop a backend API for user authentication.
4. Hash and store user passwords securely in the database.
5. Integrate the login feature with session management.
6. Create error messages for invalid credentials and locked accounts.
7. Write unit tests for the authentication API.
8. Perform security tests for potential vulnerabilities
9. Conduct end-to-end testing of the login feature.

Subject / Theme:

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

User story 2: Search for products by category

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

Tasks:

1. Design the search UI with dropdowns or filters for categories.
2. Implement category-based filters on the frontend.
3. Create a backend API to fetch products by category.
4. Optimize database queries for retrieving category data efficiently.
5. Implement pagination for search results to enhance performance.
6. Connect the search API with the frontend.
7. Write unit tests for search functionality.
8. Conduct usability testing to ensure an intuitive user experience
9. Conduct performance testing to handle large datasets.

ii. How to Prioritize User Stories During Sprint planning :

During a Sprint Planning meeting, the Product Owner, Scrum Master, and Development Team Prioritize tasks by considering :

1. Value to the customer :

- User Story 1 (Secure Login) : High priority since user authentication is fundamental to the app. Without this, users cannot access personialized features.
- User story 2 (Search by category) : Moderate priority. While it enhances users experience, it becomes valuable only after secure access to accounts is ensured.

2. Technical Feasibility :

- Assess dependencies :
  - Secure login depends on the databases setup and user management system.
  - Search by Category relies on having an existing product catalog in the database.
- Break larger tasks into smaller, achievable

tasks and consider the team's capacity for the sprint.

3. Sprint Goals :

- First Sprint : Deliver the complete secure login feature, ensuring users can log in safely.
- Second Sprint : Start developing and delivering the search functionality after user login is established.

(iii)

A Scrum board is devided into three main columns To Do, In progress, and Done. Each task moves through these stages as the team works on them.

Example Scrum Board :

To Do	In Progress	Done
Design login page UI	Implement backend authentication.	Design login page UI
Develop authentication API	Hash and store passwords securely	Write unit tests for login API

Subject / Theme:

Date: / /

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

Implement session management.	Perform client-side input validation.	
Design search page UI		
Create backend API for category search		

Explanation of columns :

- To Do : Contains all tasks not yet started.
- In Progress : Tasks currently being worked on by team members.
- Done : Completed tasks that have been verified and accepted.

Workflow in the Scrum Board :

1. Tasks start in the To Do column based on priority.
2. Team members pick tasks to work on, moving them to In Progress.
3. Once tasks are completed and verified,

Subject / Theme:

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

They are moved to the Done column.

This approach ensures visibility and progress tracking throughout the sprint.

Would you like me to create a visual representation of this Scrum board or further elaborate on sprint planning details?

Question : 2

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

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

Subject / Theme:

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

Solve:
How the Spinal, Agile, and Extreme Programming Methodologies Address Risk Management and Adaptability:

1. Spinal Model: The Spinal Model is highly effective in managing risks, especially in project with significant novelty and uncertainty.

• Risk Management: This model explicitly focuses on identifying, analyzing, and mitigating risks at each iteration or "spinal".
At the start of each cycle, risks are assessed, and solutions are developed to address high priority risks before proceeding to the next phase.

• Adaptability: The iterative nature of the spinal Model allows for frequent reassessment of goals and alignment with client needs.

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

Subject / Theme:

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

2. Agile Methodology:

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

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

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

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

Subject / Theme:

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

3. Extreme Programming (XP):

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

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

- Adaptability: Like Agile, XP embraces evolving requirements, with customer feedback incorporated after each iteration. Simple design and refactoring ensure the product can adapt to changing needs without become overly complex or expensive.

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

Subject / Theme:

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

Recommended Methodology : Agile

Given the high risks and evolving requirements, Agile methodology is the most suitable for this project.

Reasons :

1. Adaptability to change : Agile is designed to handle evolving client requirements through incremental development and regular feedback. It ensures that each sprint delivers value and aligns with updated client priorities.

2. Risk Mitigation Through Iteration : Frequent testing and integration reduce technical risks.

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

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

Subject / Theme:

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

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

Agile provides the best balance of risk management and adaptability, enabling the team to deliver a high-quality product in an evolving and uncertain environment.

Subject / Theme:

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

Question 3:

A company is working on two different projects. Project A has well defined requirements and a strict deadline, while Project B has evolving requirements with an uncertain timeline and continuous customer feedback. Both projects involve high stakes, and the team must decide which development methodology to would best suit each? Support your answer with a detailed analysis of how each methodology would address the specific needs of the projects, considering factors such as predictability, customer collaboration, and risk management.

Solution:

Project A: Well-Defined Requirements, Strict Deadline so ∴ recommended methodology is Waterfall Model.

Why waterfall?

i. Predictability: With well-defined requirements the sequential nature of Waterfall ensures

Subject / Theme:

Date: / /

☐Sat ☐Sun ☐Mon ☐Tue ☐Wed ☐Thu ☐Fri

a structured and timely delivery.

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

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

Project B:- Evolving Requirements, Uncertain Timeline, Continuous Customer Feedback:
So this is Agile Model.

Why Agile:-

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

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

Subject / Theme :

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

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

Detailed Analysis of the Needs :

Factor	Project A (Waterfall)	Project B (Agile)
Requirements	Fixed and well-defined.	Evolving and subject to change
Timeline	Strict and predictable	Uncertain ; needs flexibility
Risk Management	Minimal risk once requirements are clear.	High risk due to uncertainty; mitigated through iterations
Customer Collaboration	Minimal Once requirements are finalized.	Continuous involvement to provide feedback
Flexibility	Low	High
Adaptability	Hard to adapt to changes	designed for adaptability.

Subject / Theme:

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

Conclusion:-

Project A is best suited for the Waterfall Model, as its fixed requirements and strict deadlines align well with Waterfall's sequential nature.

Project B is best suited for the Agile Model, given its ability to handle evolving requirements and continuous customer feedback.

Question 4

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

Subject / Theme:

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

Answer:

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

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

Professional Responsibility Issues:
Software engineers face numerous ethical challanges:

i. Data Privacy.

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

ii. Algorithmic Bias
iii. Cybersecurity
iv. Software Reliability
v. Social Justice

ACM/IEEE Code of Ethics:

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

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

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

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

4. Judgement: Software engineers shall maintain integrity and independence in their

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

Subject / Theme:

profession. consistent with the public interest

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

6. Profession.

7. Colleagues

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

The code of Ethics provides a valuable resource for software engineers to navigate ethical dilemmas and make responsibly.

Subject / Theme:

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

Question 5

Given the story of the Airport Reservation System, identify at least five functional and five non-~~five~~ functional requirements for the system. In your answer, explain how each requirement contributes to the overall performance, usability, and security of the system. Consider factors such as performance, user experience, and system maintenance in your discussion.

Answer :-

Certainly, Here are five functional and five non-functional requirements for an Airport Reservation System, along with their contributions :

Functional Requirements :

1. User Registration and Login;
2. Flight Search and Booking
3. ~~Pg~~ Payment Processing

Subject / Theme:

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

4. Reservation Management
5. Notifications and Alerts

Non-functional Requirements:

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

How these Requirements contribute:

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

Subject / Theme:

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

Question :-6:

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

Answer:

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

Left Side (Verification Phase): This side focuses on ensuring the system is built correctly.

- Requirement Analysis;
- System Design;
- Architectural Design
- Module Design

Subject / Theme:

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

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

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

ii. Module Integration : Corresponds to Integration Testing.

iii. System Integration : Corresponds to System Testing.

iv. System Validation : Corresponds to Acceptance Testing.

The V-Model ensures thorough testing at each stage, aiming for early defect detection. However, it can be less flexible for projects with changing requirements.

Subject / Theme:

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

Question. 7

Explain the process of prototype development in software engineering. Discuss the key stages involved in creating a prototype and how it helps in refining software requirements. Analyze the benefits of using the prototyping model, particularly in terms of user feedback, risk reduction, and iterative development.

Answer:

Prototype development is an iterative process in software engineering where a preliminary version of a software application is created to understand user requirements, test concepts, and explore design solutions. The goal is to refine requirements and improve software quality by engaging users and stakeholders early in the development process.

Subject / Theme :

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

Key stages in Creating a Prototype :

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

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

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

4. User Evaluation :
- Share the prototype with stakeholders and end-users for feedback.

5. Iterative Refinement
- Based on feedback, refine and improve the prototype.

Subject / Theme:

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

How prototype Helps Refine Software Requirements:-

i. Early Validation of Requirements.

ii. Clear communication

iii. Discovery of Hidden Needs.

Benefits of Using the Prototype Model:

1. User feedback: Prototypes provides a tangible product for user to test and critique

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

3. Iterative Development:- Support an iterative apporoch where changes can be made incre-mentally.

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

5. Cost and Time Efficiency:- Saves time and resources by avoiding late-stage changes.

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

Subject / Theme:

Q.8 ~~Explain the Software Engineering Institute~~

Explain the process improvement cycle in software engineering and describe its key stages. Name and explain some commonly used process metrics, highlighting how they help in monitoring and improving software processes

Answer:

The process improvement cycle in software engineering focuses on refining and optimizing software development processes to improve quality, efficiency, and reliability. It follows an iterative approach to identify inefficiencies, implement changes, and measure the effectiveness of those changes.

Key stages in the Process Improvement cycle:-

1. Process Assessment : Collect data on process performance using metrics like defect density cycle time, or productivity.

Subject / Theme:

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

2. Goal Setting : Define clear, measurable objectives for process improvement. Align goals with organizational priorities and project needs.

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

4. Implementation : Apply the proposed process changes incrementally.

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

6. Review and Optimization : Evaluate the success of implemented changes based on collected data.

7. Continuous Improvement : Treat process improv as an ongoing activity, repeating the cycle periodically to adapt to evolving challenges and requirements.

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

Subject / Theme:

Commonly Used process Metrics in Software Engineering:

Process metrics provide quantitative data to monitor and assess software processes. These metrics help identify trends, measure performance, and guide decision-making.

1. Productivity Metrics: Measure the output produced relative to the input effort.
Example:- Lines of code per developer hour.
Function points deliver per team.

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

3. Cycle time:- Measures the time taken to complete a specific process or task, such as resolving a bug or delivering a features.

4. Effor Variance: The difference between planned effort and actual effort expand

Subject / Theme:

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

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

6. Process Compliance Matrics: Measure adherence to prescribed processes and standard.

How process Metrics help in Monitoring and Improving Software processes:

1. Data-Driven Decision Making
2. Early problem Detection
3. Performance Benchmarking
4. Encourging Accountability
5. Continuous Feedback Loop.

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.

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.

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 value: Deliver working software in small, frequent 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.

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.

Subject / Theme :

Date : / /

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

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.

Subject / Theme:

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

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.

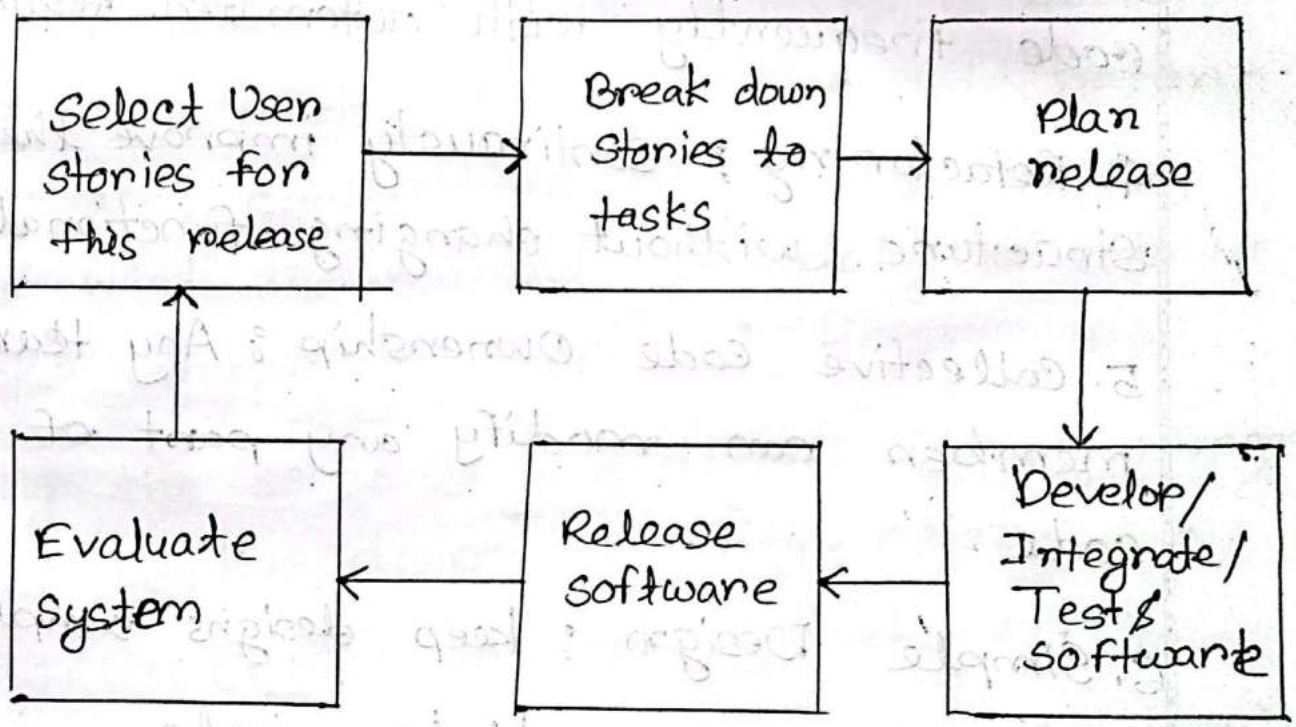


Diagram of release cycle of XP

Subject / Theme:

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

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 status. 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.

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_status, 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:

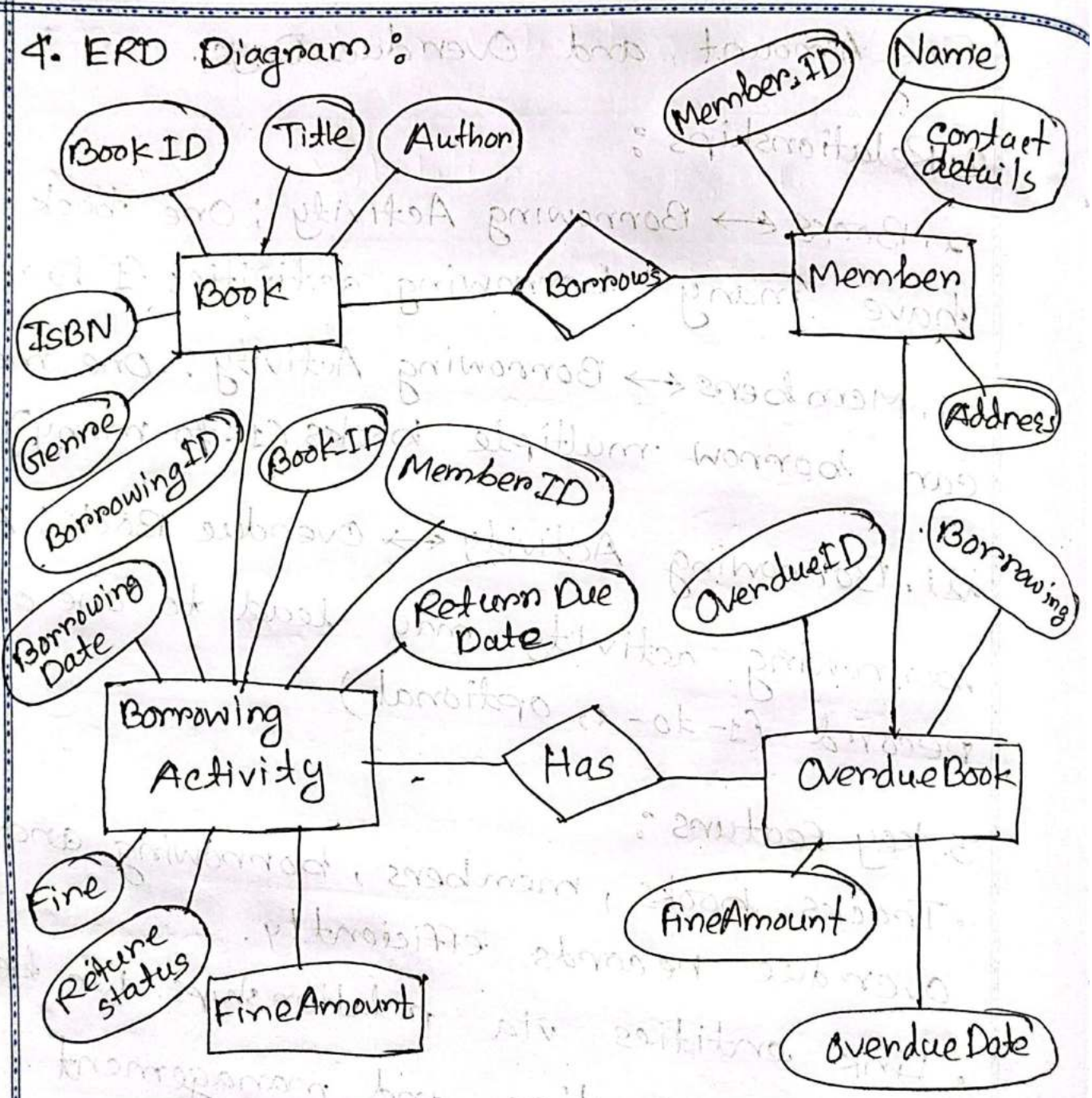
- Tracks books, members, borrowing and overdue records efficiently.
- Link entities via relationships for better data organization and management.

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."

Subject / Theme:

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

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.

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 :

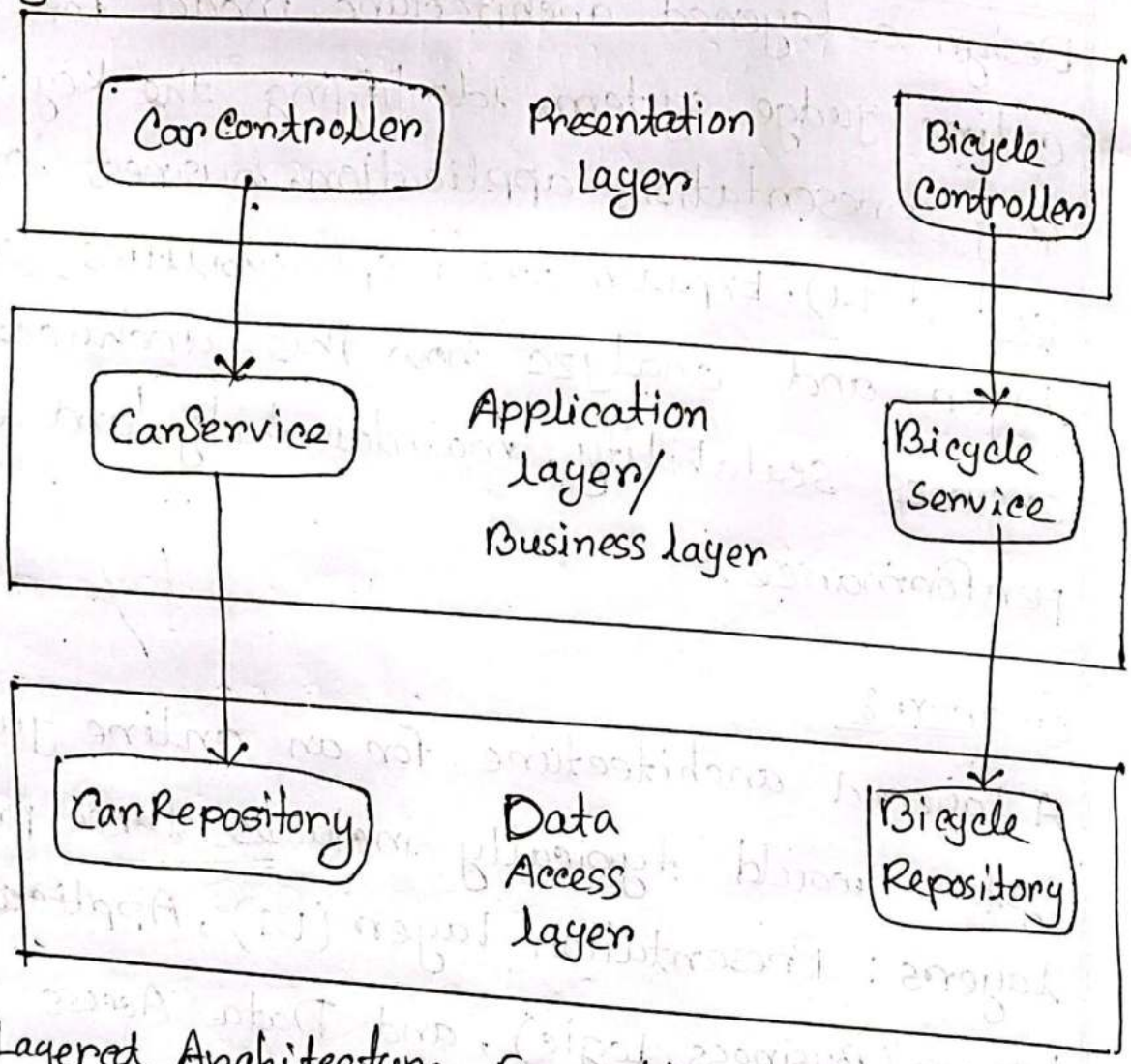
A layered architecture for an online judge system would typically includes three primary layers : Presentation Layer (UI), Application Layer (Business Logic), and Data Access Layer (Database Interaction), with each layer having well-defined responsibilities and interactions only with the layer directly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

Subject / Theme:

Date: / /

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

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.

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

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.

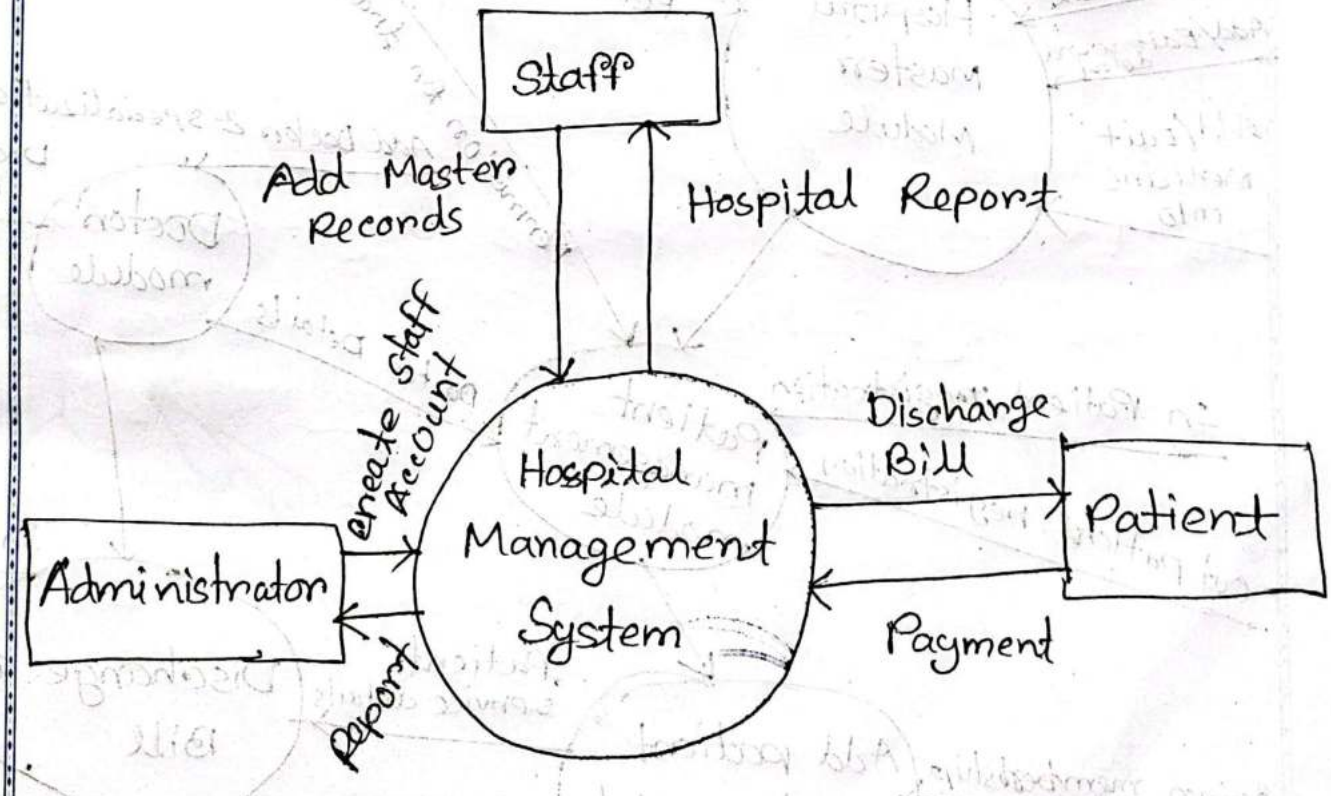


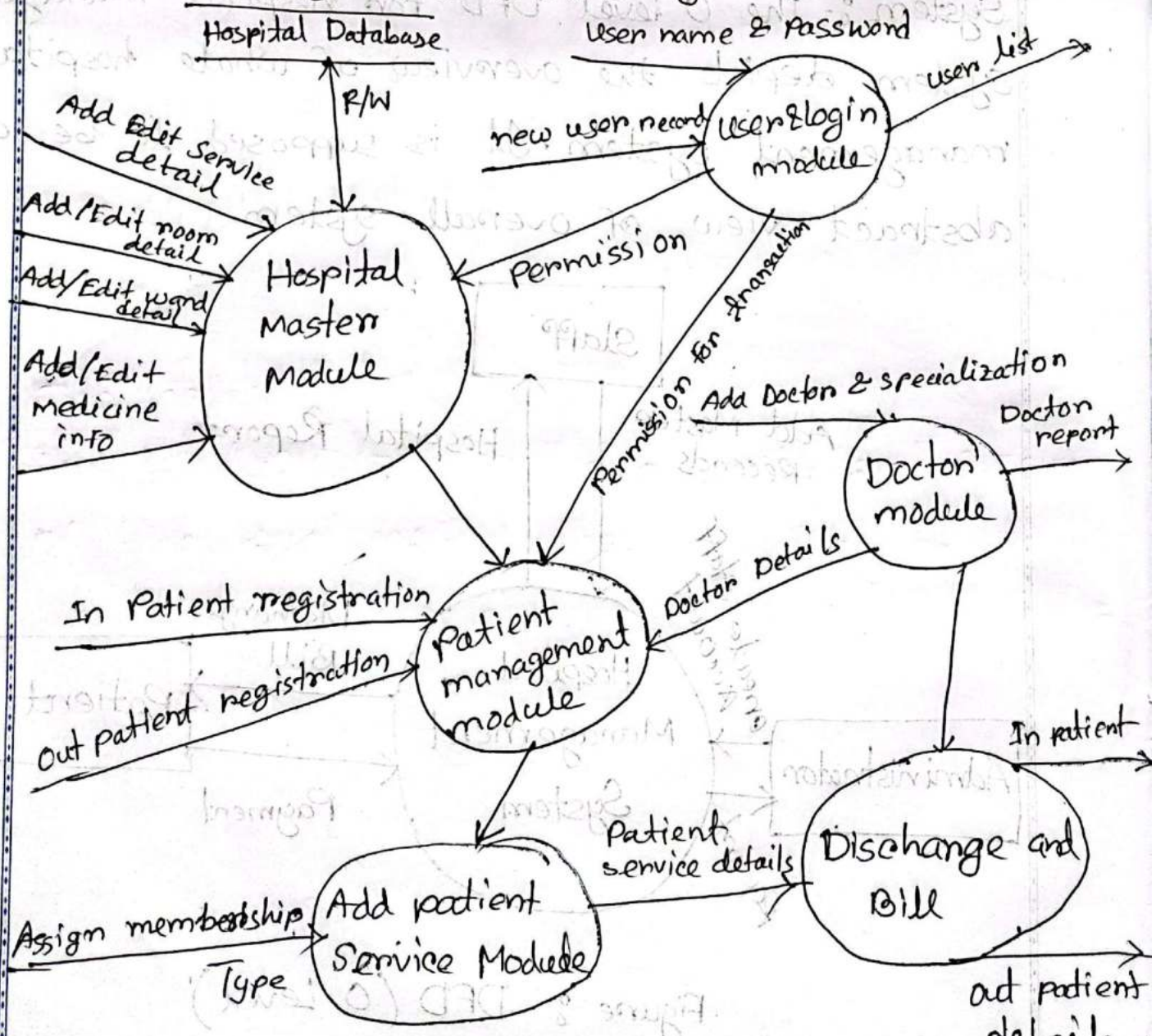
Figure : DFD (0 Level)

Subject / Theme:

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

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:

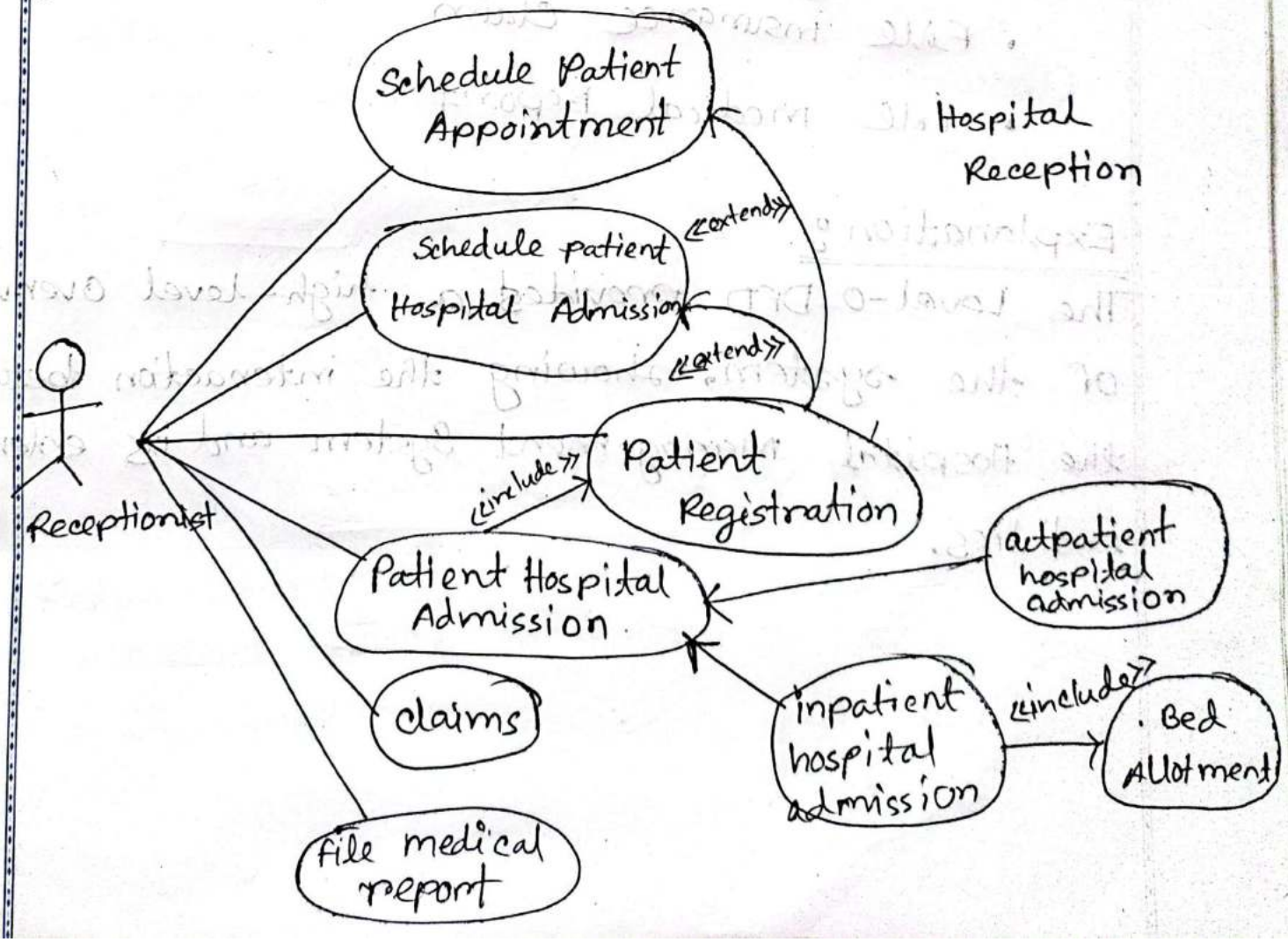
Date: / /

☐ 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:



Subject / Theme:

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

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:

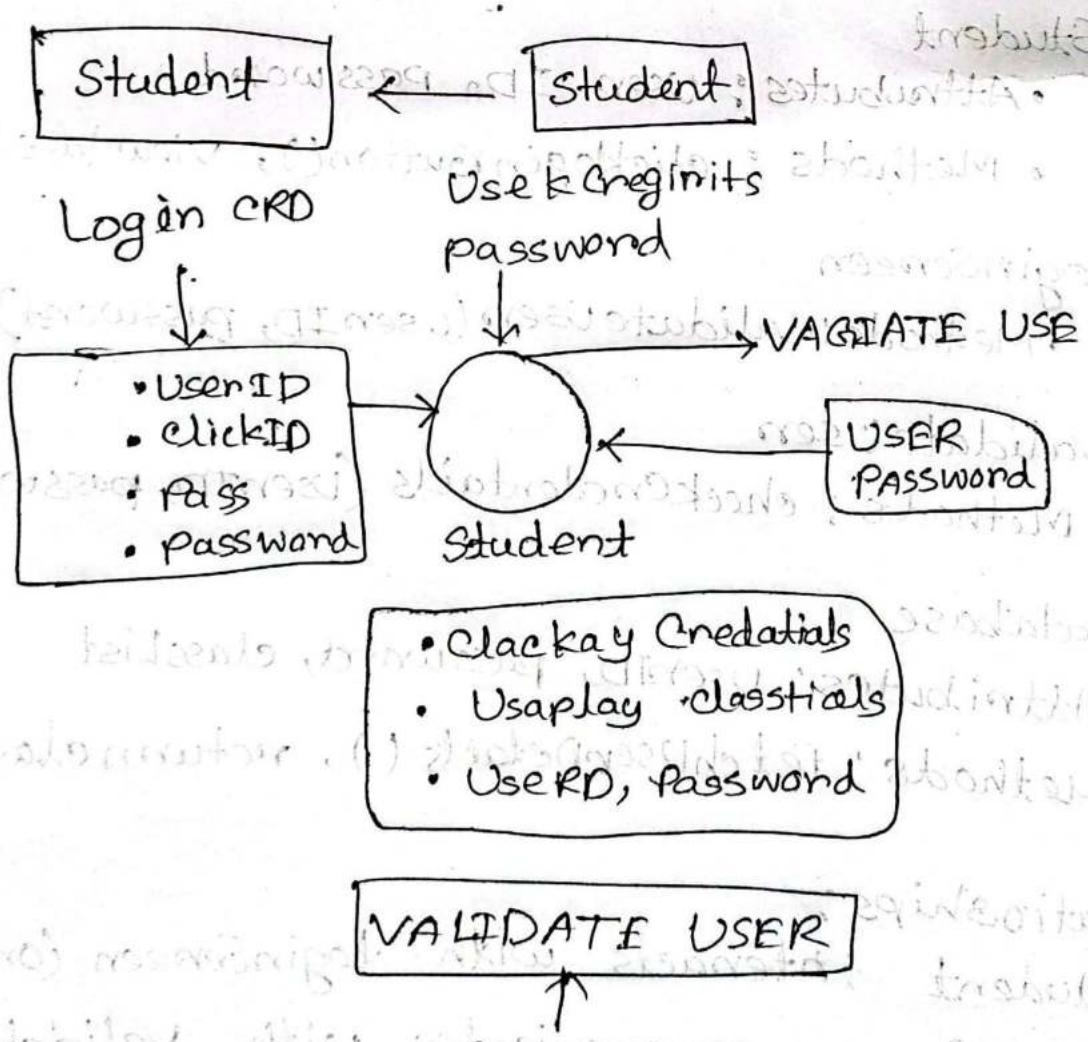
- Student interacts with LoginScreen (one to one)
- LoginScreen communicates with ValidateUser (one to one).

Subject / Theme:

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

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

Now, let me genarate 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 logine 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.

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.

Subject / Theme:

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

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.

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.

Subject / Theme:

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

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.

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.

Subject / Theme:

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

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 traditional

Subject / Theme:

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

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.

Subject / Theme:

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

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)

Subject / Theme:

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

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());

    }
}
```

Subject / Theme:

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

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.

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 Exception, Setup function and Timeout Rule. How do you solve it?

Subject / Theme :

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

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 (Interrupted
                                Exception e) { }
}
public void storeInMemory (int value) {
    this.memory = value;
}
public int getMemory (){
    return this.memory;
    }
}
```

In JUNit 4, we can apply :

i. Exception Handling

ii. Setup function

iii. Timeout Rule

Subject / Theme :

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

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);
    }
```

```
@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.