Ans to the qno - 2(1)

User Story 1: As a user, I want to log in securely so that I can access my account.

Task:

1) create a login Page ui design
2) Develop a backend authentication system
3) Implement password encryption and hashing
4) Set up automated security testing

User Story 2: As a user, I want to search for Products by catagory to find items easily.

Tasks:

1) Design the ui for the search and catagory filter components
2) Create a database schema for product catagories
3) Develop an API endpoint for fetching Products by catagory

(2)

4) Implement a sorting and pagination feature.

(11)

During the sprint planning meeting, the development team and Product owners can prioritize these user stories based on two key factors.

1) value to the customer:

* High Priority: user story1: logging in securely is critical for all users to access their accounts.
* Searching for Products improves usability but does not block core functionality like login.

(11) Technical Fesibility

(1) evulate the effort required for each user story using story points.

(1) Break each story into smaller increment

that can be completed within a single sprint.

(IV)

Board Layout:

To Do	In Progress	Done
create a login page ui design	Develop backend authentication	conduct end to end testing
Design ui for category Search	Implement Password encryption	Write unit tests for login
Develop api endpoint for Search	Integrate third-party libraries	

(4)

(2)

Agile methodology is the most suitable for a high-risk project with evolving requirements because it emphasized frequent iterations, early delivery of value, and close collaboration with the client. It adapts quickly to changing needs, mitigates risks through incremental progress, and ensures alignment with the client's vison. For very high technique, Agile can incorporate extreme Programming (XP) Practices like test-driven development and pair Programming to enhance quality.

→ Project A (well-defind reavirements, strict deadline): use waterfall for its structured predicatable approach that ensures timely delivery with clear milestones.

→ Project B (evolving reavirements, uncertain timeline): use agile for its flexibitily, adaptabilily to chang, and focus on continuous customer collaboration. If ~~technicar~~ technical tusks are high, integnate extreme Programming (XP) Practices for avalily assurance.

Sowtwane engineering ethics emphasize public interest, quality, honesty confidentiality, competence and accountability. Professionals must prioritize safety, respect privat, and avoid conflicts of interest.

The ACM/IEE code of ethics guides ethical decision-making by promoting principles such as prioritizing public welfare, maintaining competence, being transparent, protecting privacy and ensuring fairness. It helps engineers address issues like sowtwane risks, data privacy, and professional responsibility while making decisions that align with societal and stake-holder needs.

Functional Requirements:

1. Flight booking and cancellation core functionality for managing reservations.
2. Search for flights: Allows users to find flights based on preferences
3. User Authentication: Secures accounts and protects use data.
4. Payment Integration: Enables seamless transcrip
5. Real-Time Seat Availability: ensures accurate and updated booking optims.

Non-functional requirements

(1) System Performance: supports 10000+ concurrent users.

(2) Response Time: Displays results within 3 seconds

(3) Data Security: Encrypts user data to ensure

(8)

Compliance and safety

(4) Scalability: Handles increased traffic during peak times.

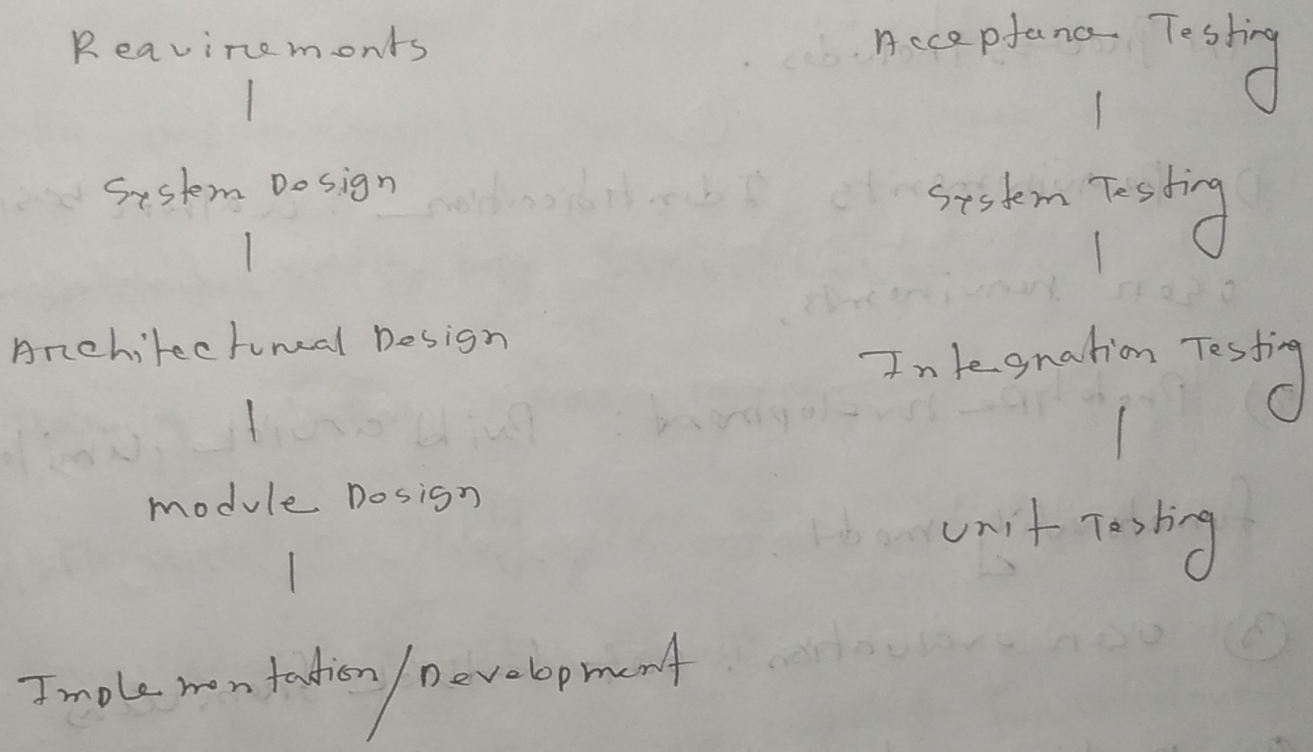
(5) System uptime: Maintains 99.9% availability with minimal downtime.

Both sets of requirements ensure the system is functional, reliable, and user-friendly.

(6)

V-model of Testing Phases: The V-model (verification and validation model) is an extension of the waterfall model where testing phases are explicitly mapped to corresponding development stages. It emphasizes a

Parallel relationship between development and testing activities, ensuring that every development activity is verified and validated early in the process.



(10)

(7)

The Prototyping model is software engineering involves creating an early, working version of the software to gather user feedback and refine requirements.

The Process includes.

1) Requirements Identification: Gather basic user requirements,

2) Prototype Development: Build a quick, limited functionality model.

3) User evaluation: Get feedback from users.

4) Refinement: Adjust requirements based on feedback

Benefits

(i) User feedback improves requirement accuracy

(ii) Risk Reduction: Identifies issues early on

minimizes ~~teaviement~~ misunderstandings.

- Iterative Development allows flexibility and faster evolution of the Product.

(8)

Process improvement cycle in software engineering

1. Assessment - Identify Process Weknesses
2. Planning - ~~Defina~~ Define improvement goals.
3. Implementation - Apply Processenhancements
4. Monitoring - Track key metrics.
5. Optimization - refine and repeat for continuous improvement.

Common Process metrics

→ Defect Density - measures code quality

→ cycle Time - Tracks development Speed

→ lead time - Assess Process efficiency

(12)

Productivity - measure effort per feature.

The cycle enhances Software Processes using key metrics to improve quality efficiency and customer satisfaction.

(9)

The capability maturity model (CMM) developed by the Software engineering institute (SEI) is a framework that helps organizations improve their Software development processes systematically. It defines five maturity levels, each representing a stage of Process improvement.

Five level of SEL CMM and Their contributions

1. Level-1 - initial

→ Characteristics: Ad hoc, unpredicatable, reactive, and dependent on individuals, rather than defined process.

2 - Level 2 - Repeatable

- characteristics: Basic Project management Practices are established, ensuring consistency.

→ Contribution: Introduces discipline, helping teams meet Project deadlines and budgets.

3 - Level 3 - Defined

→ characteristics - Organization-wide Process standardization and documentation.

→ Contribution: ensures Process consistency improving quality and reducing defects.

(10) (15)

Core Principles of Agile →

(I) Collaboration - People over Processes

(II) Working software - Deliver value quickly

(III) customer feedback - continuous engagement

(IV) Adaptability - embrace change

Application in Different environments

(I) Startups - quick iterations for MVPs

(II) large enterprises - scaled Agile frameworks

(III) regulated industries - Hybrid Agile for compliance

Benefits →

- faster delivery
- Higher flexibility
- Improved collaboration
- Better customer satisfaction

(16)

Challenges

- Hard to scale
- Requires active engagement
- Less Predictability

(4)

Release Cycle of Extreme Programming (XP)

- The Extreme Programming (XP) release cycle follows an iterative and incremental approach, focusing on frequent release and continous customer feedback. Below,

[Planning] → [Iteration] → [Feedback] →

[Release] → [Repeat]

Key Stages in xp release cycle

1. Planning - define users stories and prioritize features.
2. Iteration - short development cycles
3. Feedback - continuous customer input to refine features.

Influential Programming Practices in xp

(i) Test-Driven development - write tests before code to ensure quality

(ii) Pair Programming - Two developers work together to improve design and reduce bugs.

(iii) Simple Design - keep the design as simple as possible

(18)

(12)

Entity relationship Diagram (ERD) for a Digital library management system.

Entities and Attributes

(I) Book
  (i) Title (ii) Author (iii) ISBN

(II) member
  (i) name (ii) membership (iii) contact Details

(III) Borrowing
  (i) member-ID (ii) Book-ID (iii) Borrowing Date

(IV) Overdue
  (i) Borrow-ID (ii) Fine Amount

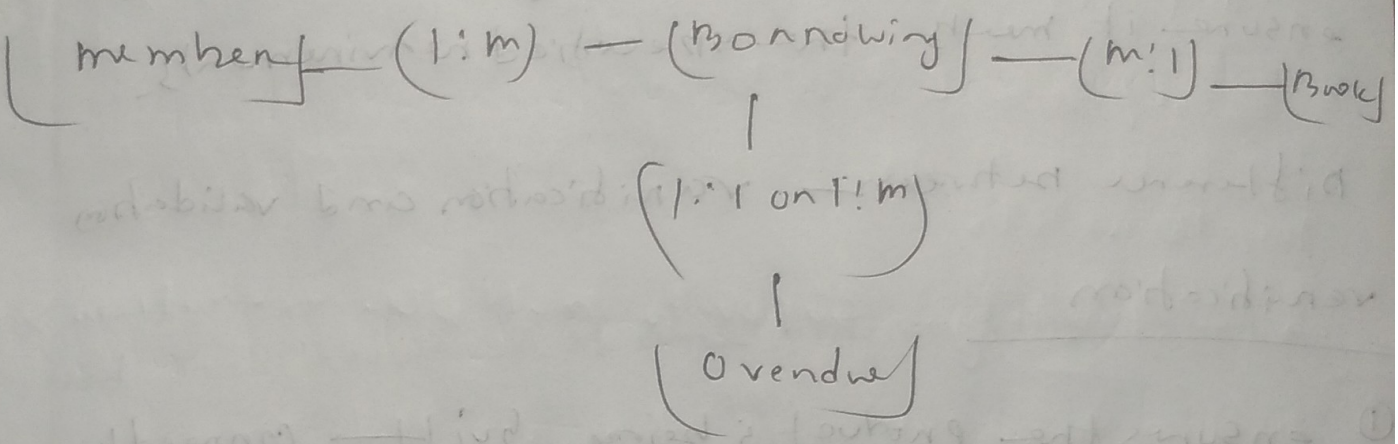
relationships:

→ A member can borrow multiple Books

→ A book can be borrowed multiple times by different members

→ A borrowing record is linked to Overdue if the book is not returned on time

ER representation:



→ Member-ID is Borrowing connects to member

→ Book ID is Borrowing connects to book

→ Borrow-ID is Overdue connects to Borrowing

(20)

(120)

Software testing is the process of evaluating a software application to identify defects and ensure it meets the specified requirements.

Difference between verification and validation

verification

① ensures the product is being built correctly by checking if it meets design and requirements

② Process-oriented

validation

① checking if a login form is correctly designed as per the specifications

validation

(i) ensures the right product is build by checking if it meets user need

(ii) Product - oriented

(iii) Testing if user can successfully login

(4)

Layered Architecture of an online judge system

1. Presentation layer (API Layer) - Manages authentication, session handling and communication between UI and backend.

i. Presentation layer (UI Layer) - Provides the user interface for submissions login, and result display

2. Application Layer (API Layer) - manages authentication, session handling and communication between UI and backend.

3. Business logic Layer (Judging & execution engine)
   - Executes, validates, and scores submissions in a secure sandbox
4. Data Layer (Database and storage) - Stores user data, Problems, test cases, and results using SQL/NoSQL data bases.

Key Benefits;

→ Scalability - Supports multiple execution servers and cloud ~~deveto~~ deployment.

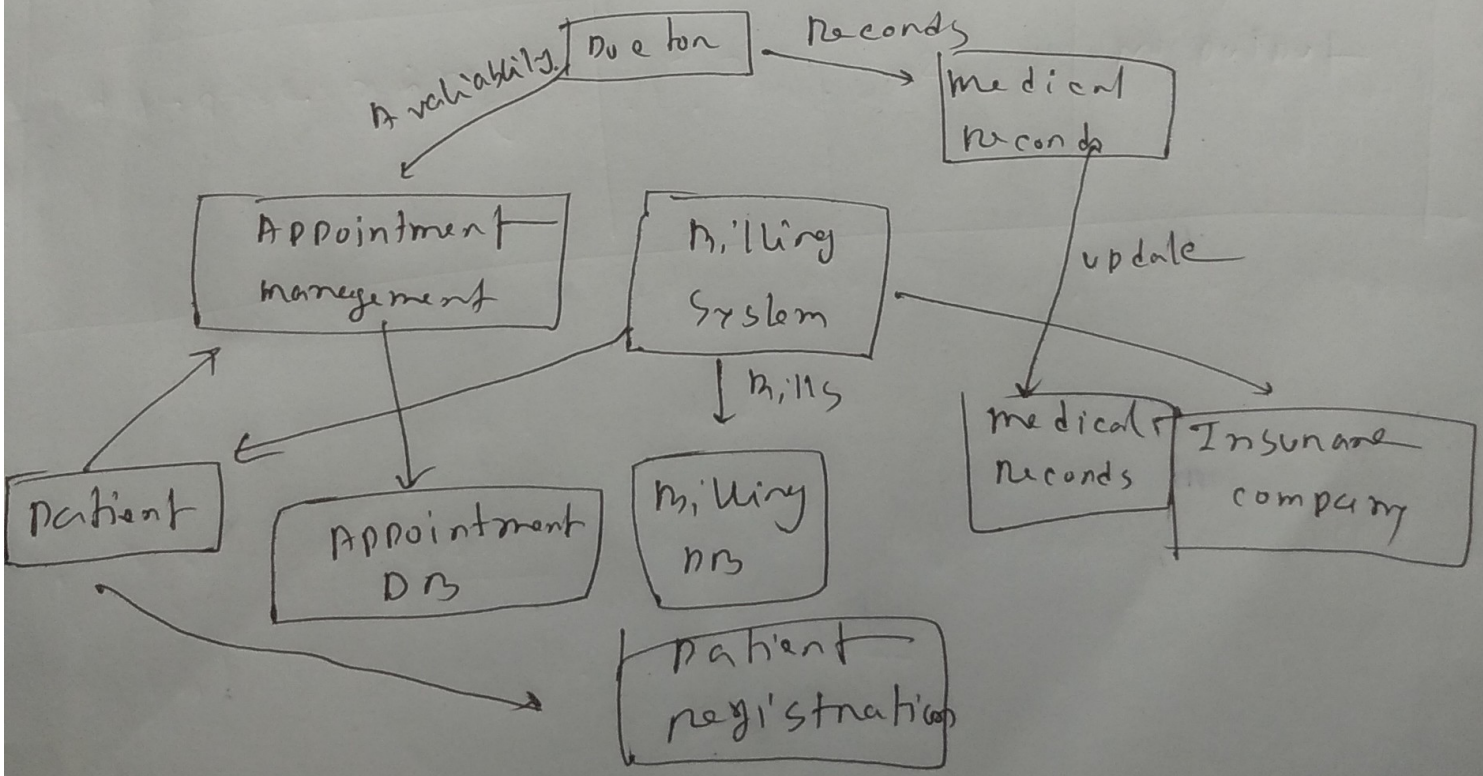
→ Maintainability - Modular Layers allow independent updates.

→ Performance - caching, parallel execution and load balancing ensure fast responses.

Itones how can break it down:

1. Data Flow Diagram (DFD)

- Level 0: Represents the entire Hospital Reception Subsystem as a single process with external entities interacting with it.
- Level 1: Showing data flow between different Processes like appointment scheduling, patient admission, billing and record management



Based on the UML Sequence diagram: we can derive a UML class diagram by identifying the key entities and their relationships. The main entities involved in this Real scenario are:

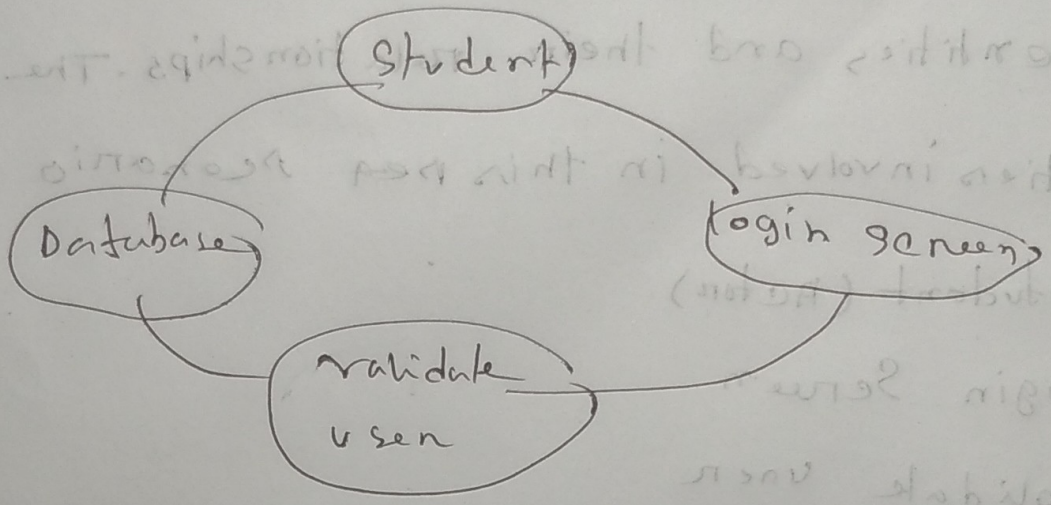
1. Student (Actor)
2. login Screen
3. validate User
4. Database

Class Diagram components

- student: Represents the User interacting with the system.
- loginScreen: Handles User inputs and communicates with the validate User class
- validate User: Process login credentials and interacts with database

Database: Stores user credentials and class list information

Class diagram relationship:



20

Here in the test table and corresponding Junit test class in Java

~~9-72~~
(17)

Differences Between QA and QC

QA

Aspect	QA	QC
Focus	Prevents defects through process improvement	Detects and fixes defects in the final product
Approach	Proactive - Ensures best practices before issues arise	Reactive - Identifies defects after development
Activities	Process audits, training, standards, compliance	Testing, inspections, defect identification

Impediments to QA and QC

QA

1. Lack of Defined Process - Poor documentation and unclear guidelines lead to inconsistencies

(14) 2. Time and cost constraints: Investing in QA takes time and resources, which some companies may not prioritize.

OR

Incomplete Testing - Tight deadlines may lead to skipped tests, increasing the risk of undetected defects.

(18)

Role of QA in SDLC

→ Requirement Analysis: ensures clear, testable requirements.

→ Design - reviews architecture of feasibility and Standards

→ Development - enforces coding best Practices, unit testing.

- Testing: conducts functional, Performance and Security tests.

→ Deployment: verifies release quality and deployment process.

→ Maintenance: monitors system performance and regression tests updates.

(19)

Rapid Application Development (RAD) model

→ RAD is an iterative user-focused approach that speeds up software development using Prototyping ~~const~~ continuous feedback

Key phase

→ Business modeling - Define objective and workflows

→ Data modeling - Structure data relationships

(26)

→ Process modeling:- convert data into functional Process

→ Application Generation: Rapid Prototyping and development

→ Testing and Deployment - Continuous feedback and refinement

Advantages

(I) Faster delivery with interative development

(IV) Higher user satisfaction through constant feedback

(III) Flexibility to adapt to changing requirement

(II) Reduced risk with early defect detection.

Test table 20

Decision	x input	y input	Expected output
y is zero	5	0	y is zero
x is zero	0	3	x is zero
loop does not run	0	2	No output
numbers divisible by y	4	2	(2, (4'
number not divisible by y	4	3	(3'
Edge case: y is neg	5	-2	(2', (4'
Edge case: x is neg	-3	2	No output

Junit 4 Test code with exception Handling, setup and Timeout Rule

Junit 4 is commonly used for unit testing in Java. Below is a test suite that demonstrates:

→ exception Handling :→ verifies if a method throws an ~~exception~~ excepted exception.

→ Setup Function - Initializes resources before each test case.

Production code

```
public class calculator {
    public int divide(int a, int b) {
        if(b == 0)
            throw new ArithmeticException("cannot
            divide by zero");
        }
        return a/b;
    }
```

Junit & Test code

```
import static org.junit.Assert.*;
import org.junit.*;
import org.junit.rules.Timeout;
Public class calculator Test {
  Public void setUp() {
    calculator = new calculator();
  }
  Public void test Divide by Zero() {
    calculator.divide (10, 0) }
  Public Timeout globalTimeout = Timeout.Seco
  Public void testLongRunningOperation() throws
  InterruptedException { assertEquals(42,
    calculator, longRunningOperation());
  }
}
```