VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

OBJECT ORIENTED MODELING

Submitted by

Shlok Shivaram Iyer
(1BM22CS260)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
September-2024 to January-2025

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**OBJECT ORIENTED MODELING**" was carried out by **Shlok Shivaram Iyer (1BM22CS260),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-2025. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Modeling- (23CS5PCOOM)** work prescribed for the said degree.

Sunayana S
Assistant Professor
Department of CSE
BMSCE, Bengaluru

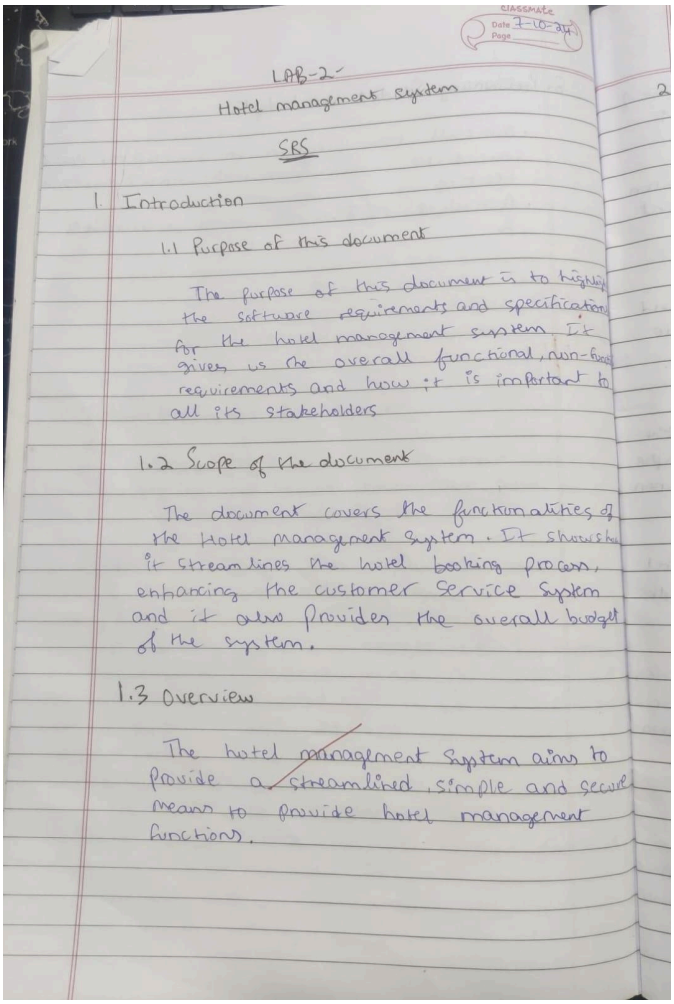
Dr. Kavitha Sooda
Professor and Head
Department of CSE
BMSCE, Bengaluru

Table of Content

https://github.com/shlokiyer123/oomd_1BM22CS260

Hotel Management System

Software Requirements Specification (SRS)



LAB-2

Hotel management system

SRS

1. Introduction

1.1 Purpose of this document

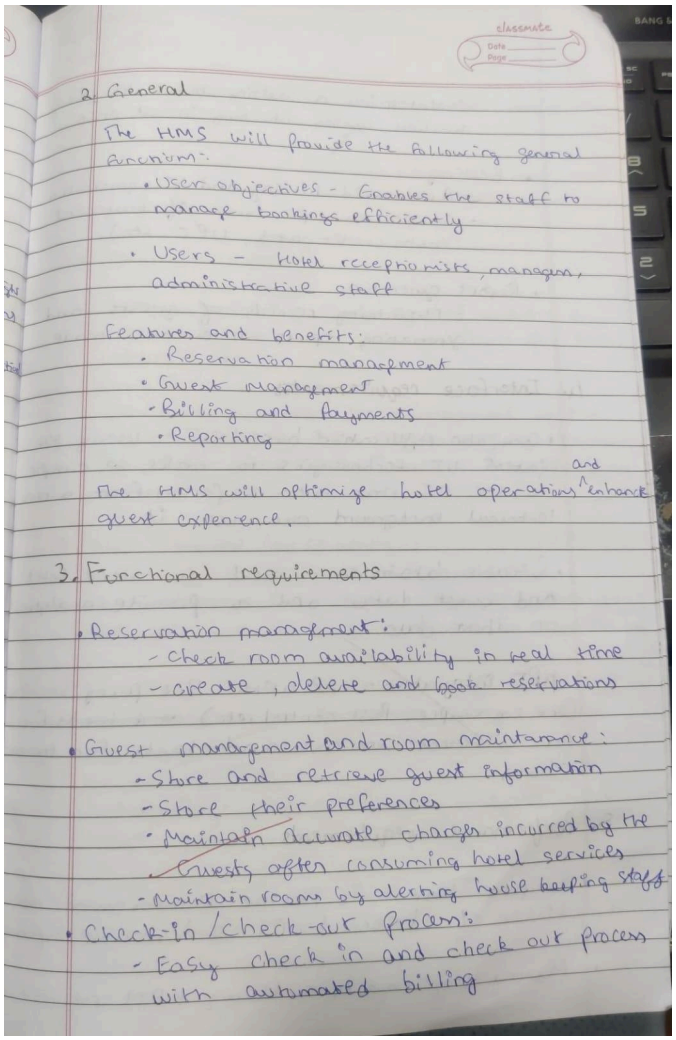
The purpose of this document is to highlight the software requirements and specifications for the hotel management system. It gives us the overall functional, non-functional requirements and how it is important to all its stakeholders

1.2 Scope of the document

The document covers the functionalities of the Hotel management system. It shows how it streamlines the hotel booking process, enhancing the customer service system and it also provides the overall budget of the system.

1.3 Overview

The hotel management system aims to provide a streamlined, simple and secure means to provide hotel management functions.



2. General

The HMS will provide the following general functions:

- User objectives - Enables the staff to manage bookings efficiently
- Users - Hotel receptionists, managers, administrative staff

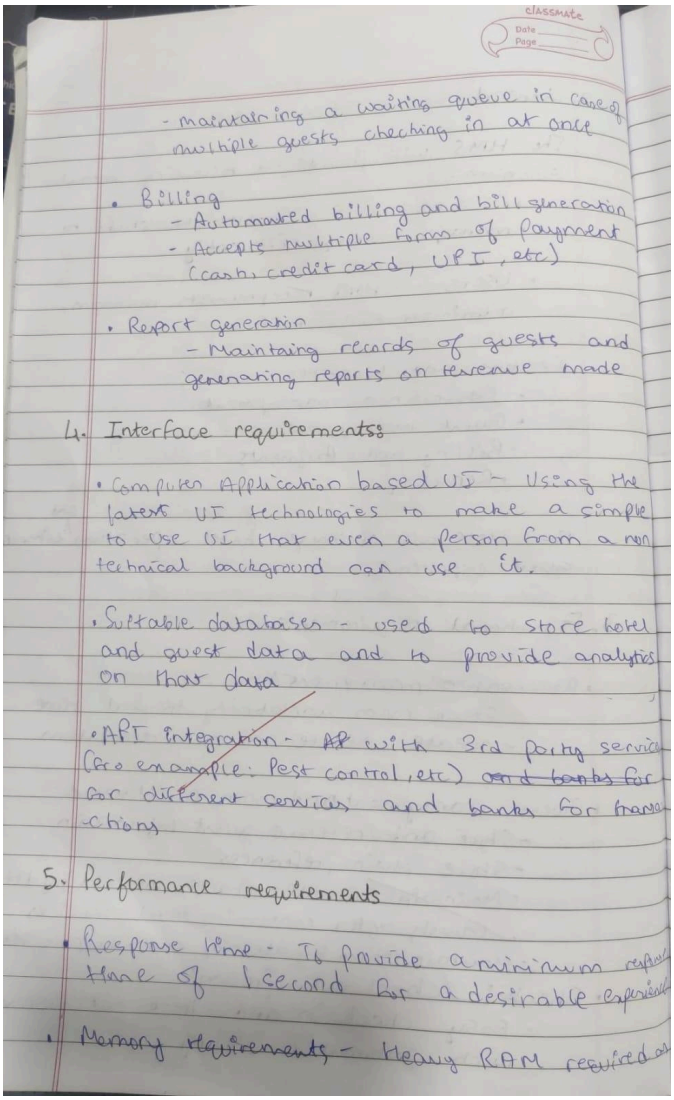
Features and benefits:

- Reservation management
- Guest management
- Billing and payments
- Reporting

The HMS will optimize hotel operations and enhance guest experience.

3. Functional requirements

- Reservation management:
  - check room availability in real time
  - create, delete and book reservations
- Guest management and room maintenance:
  - Store and retrieve guest information
  - Store their preferences
  - Maintain accurate charges incurred by the guests after consuming hotel services
  - Maintain rooms by alerting house keeping staff
- Check-in/check-out process:
  - Easy check in and check out process with automated billing



- Maintaining a waiting queue in case of multiple guests checking in at once

- Billing
  - Automated billing and bill generation
  - Accepts multiple forms of payment (cash, credit card, UPI, etc)

- Report generation
  - Maintains records of guests and generating reports on revenue made

4. Interface requirements:

- Computer Application based UI - Using the latest UI technologies to make a simple to use UI that even a person from a non technical background can use it.

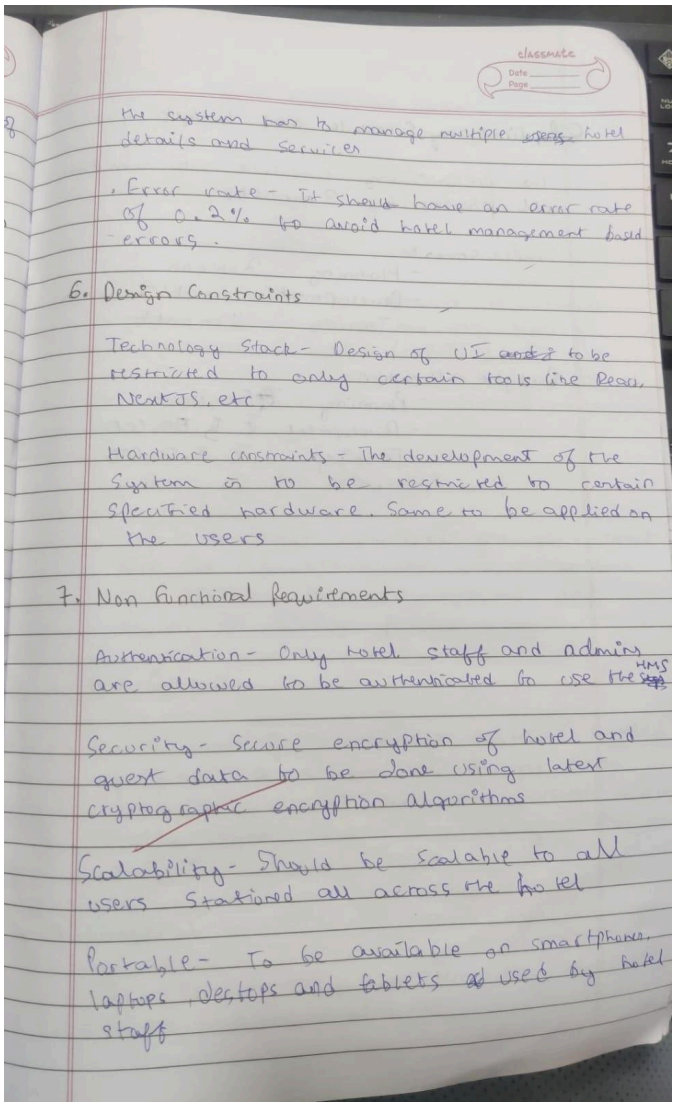
- Suitable databases - used to store hotel and guest data and to provide analytics on that data

- API integration - ~~API~~ with 3rd party services (Eg example: Pest control, etc) ~~and banks for~~ for different services and banks for transactions

5. Performance requirements

- Response time - To provide a minimum response time of 1 second for a desirable experience

- Memory requirements - Heavy RAM required as



the system has to manage multiple ~~users~~ hotel details and services

- Error rate - It should have an error rate of 0.2% to avoid hotel management based errors.

6. Design Constraints

Technology Stack - Design of UI ~~and~~ to be restricted to only certain tools like React, NextJS, etc.

Hardware constraints - The development of the system is to be restricted to certain specified hardware. Same to be applied on the users

7. Non Functional Requirements

Authentication - Only hotel staff and admins are allowed to be authenticated to use the HMS

Security - Secure encryption of hotel and guest data to be done using latest cryptographic encryption algorithms

Scalability - Should be scalable to all users stationed all across the hotel

Portable - To be available on smartphones, laptops, desktops and tablets ~~of~~ used by hotel staff

Preliminary Schedule and budget

The preliminary schedule is to complete all phases by 3 months

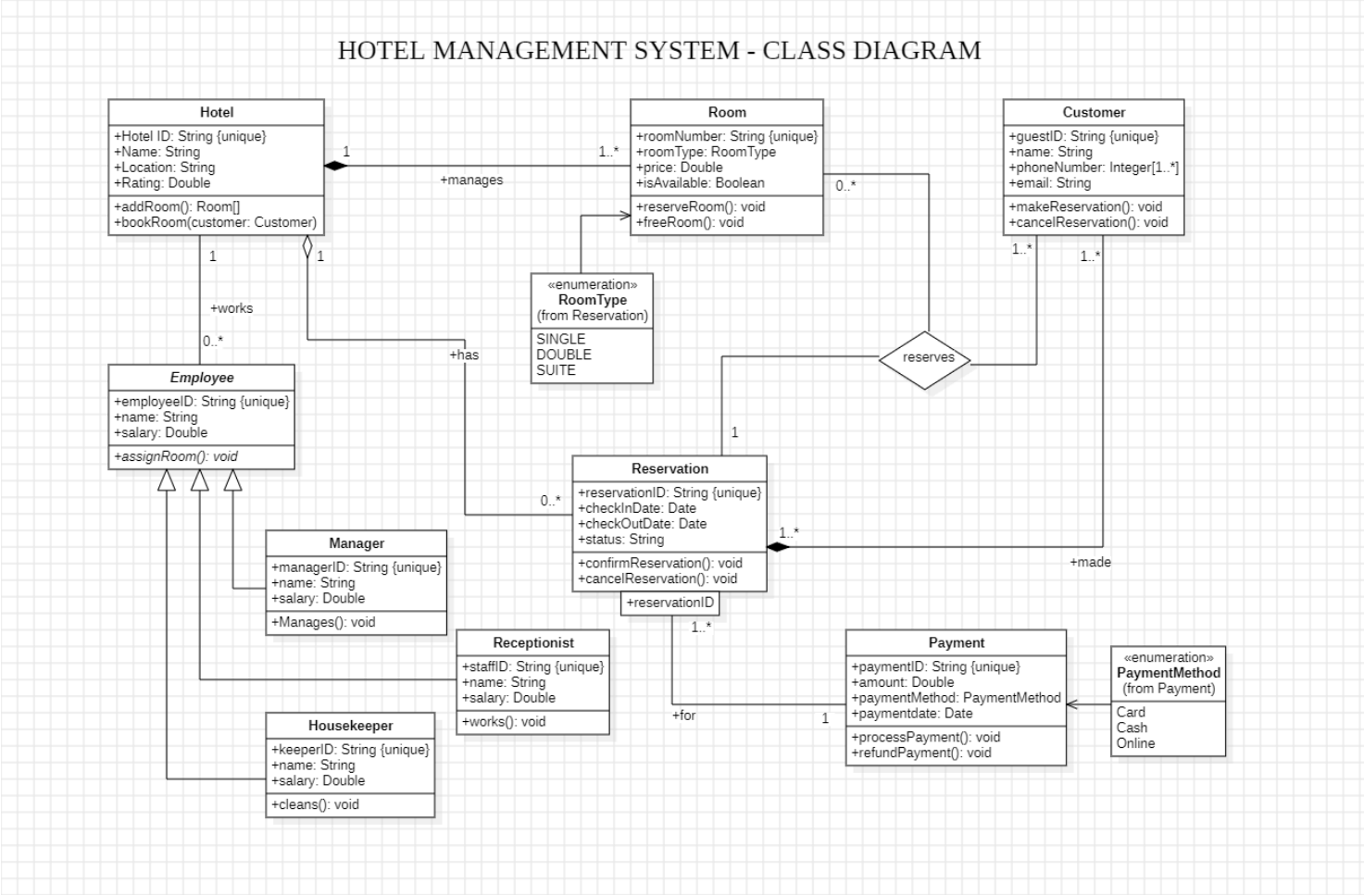
Mile stones :

- Planning 2 weeks
- Development 1 month 2 weeks
- Testing 1 month

Budget allocation

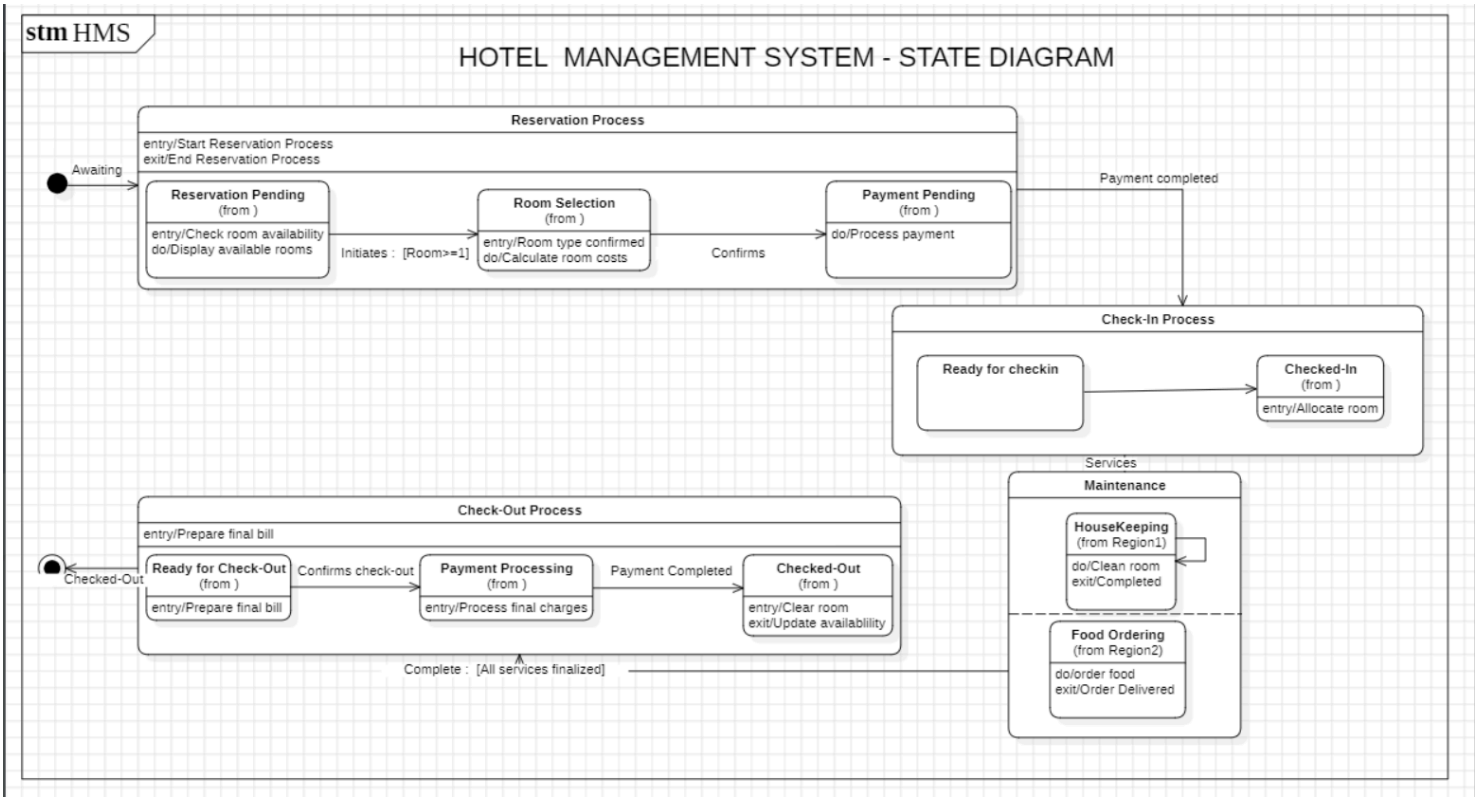
- Planning ₹50,000
- Development ₹ 5,00,000
- Testing ₹ 3,00,000

CLASS DIAGRAM



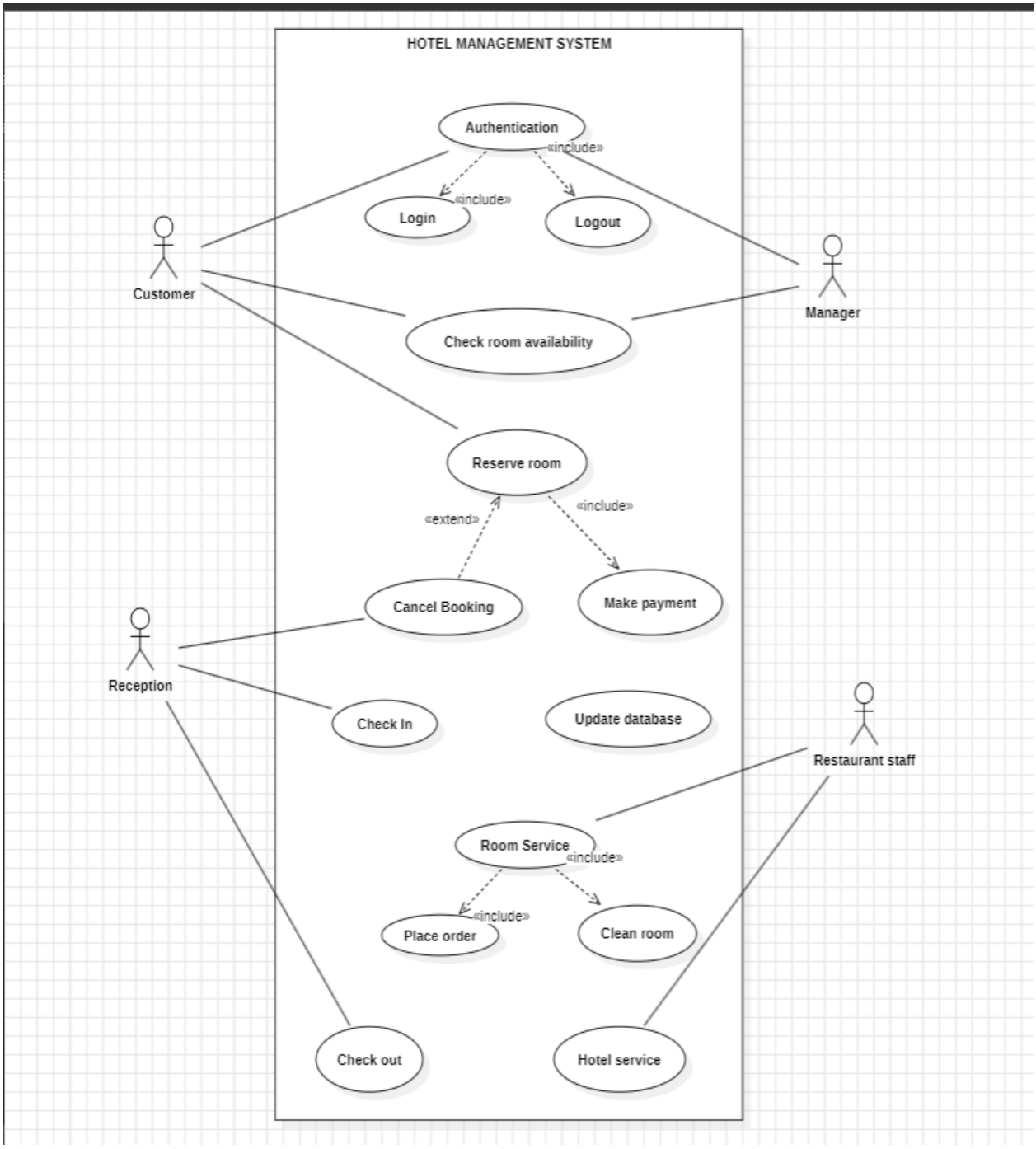
The class diagram illustrates a Hotel Management System comprising various entities and their relationships. The central class, Hotel, manages multiple Rooms, each identified by attributes like roomNumber, roomType, price, and availability. Customers can reserve rooms through the Reservation class, which records details like checkInDate, checkOutDate, and status, and is associated with Payment, handling methods such as processPayment() and refundPayment(). The system includes Employees, categorized as Managers, Receptionists, and Housekeepers, each with specific responsibilities such as managing operations, handling reservations, or maintaining cleanliness. The Customer class allows users to make or cancel reservations. Enumerations for RoomType (e.g., SINGLE, DOUBLE) and PaymentMethod (e.g., Card, Cash) add structured classifications. Overall, the diagram captures the relationships and functionalities necessary for a comprehensive hotel management system.

STATE DIAGRAM



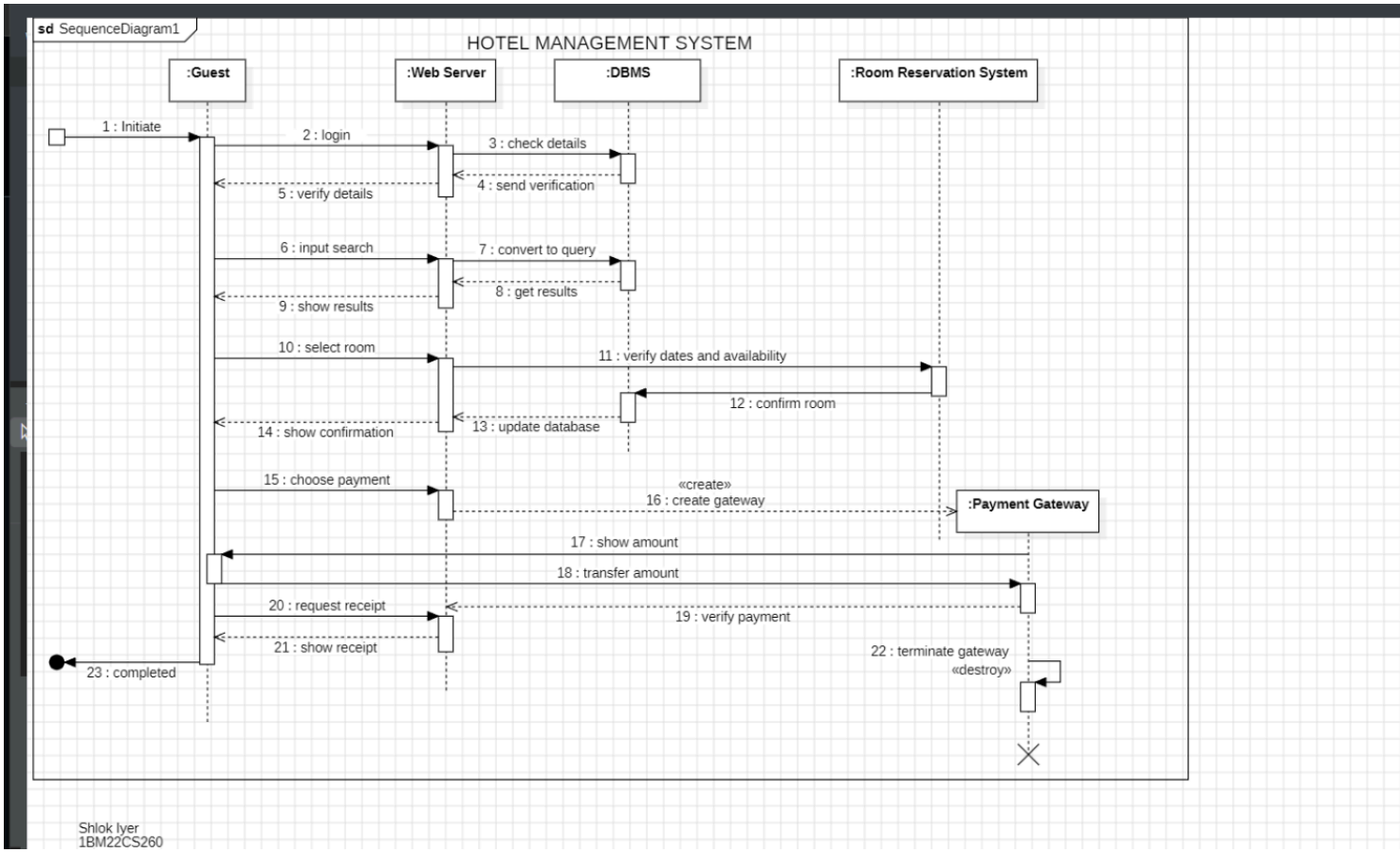
The state diagram depicts the Hotel Management System processes, starting with the Reservation Process (from Reservation Pending to payment completion), followed by the Check-in Process (Ready for Check-In to Checked-In). During the stay, the In-Service Process manages tasks like Housekeeping, Food Ordering, and Laundry Services. Simultaneously, Maintenance Mode ensures room upkeep (Pending to Completed). The system concludes with the Check-Out Process, including billing, payment, and clearing rooms. The diagram effectively outlines the workflow and transitions between states.

USE CASE DIAGRAM



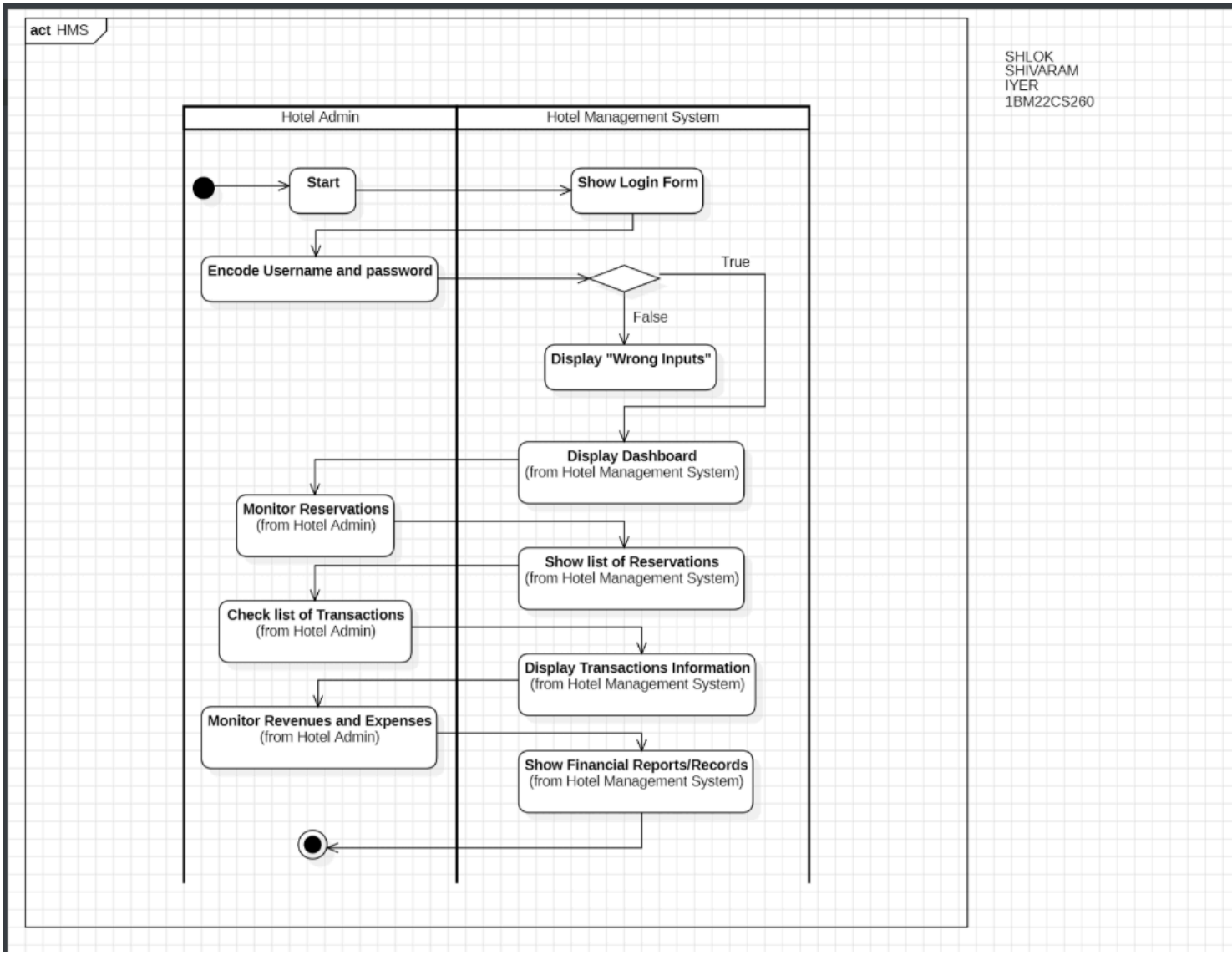
The use case diagram depicts the Hotel Management System, highlighting interactions between Customer, Manager, Receptionist, and Restaurant Staff. Key actions include Authentication, Room Reservation, Payment, Check-in, Room Service, and Checkout. It shows relationships like include and extend to represent interconnected functionalities within the system.

SEQUENCE DIAGRAM



The diagram illustrates the sequence of interactions involved in a hotel booking process. It starts with a customer initiating the request, followed by the hotel system checking availability and returning the status to the customer. Upon confirmation, the reservation system creates a reservation and allocates a room. The customer then proceeds to request payment, which is processed by the payment gateway. Once the payment is successful, an invoice is created and a transaction is generated. Finally, the customer receives the invoice and the booking is confirmed.

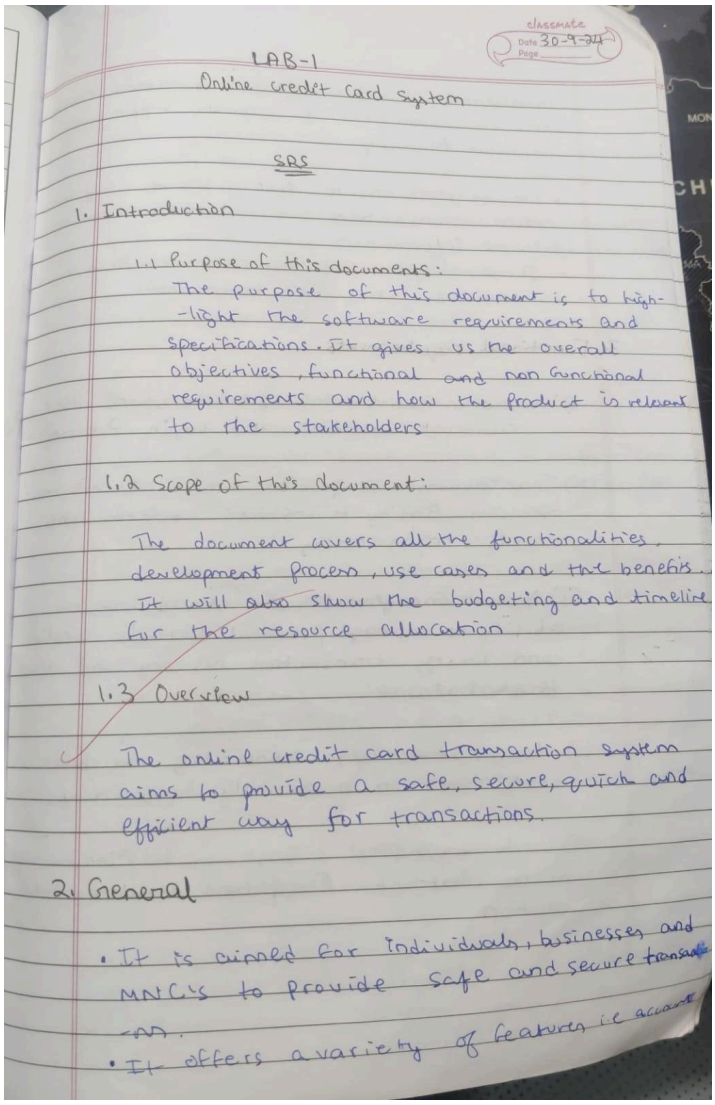
ACTIVITY DIAGRAM



The diagram illustrates the activity flow of a hotel management system. It begins with the hotel admin starting the system and being presented with a login form. After successfully entering their credentials, the admin is granted access to the dashboard. From there, the admin can perform various tasks, such as monitoring reservations, checking transactions, and viewing financial reports. The system provides the admin with the necessary information and tools to manage these aspects of the hotel.

CREDIT CARD PROCESSING

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)



LAB-1

Online credit card system

SRS

1. Introduction

1.1 Purpose of this documents:

The purpose of this document is to highlight the software requirements and specifications. It gives us the overall objectives, functional and non functional requirements and how the product is relevant to the stakeholders

1.2 Scope of this document:

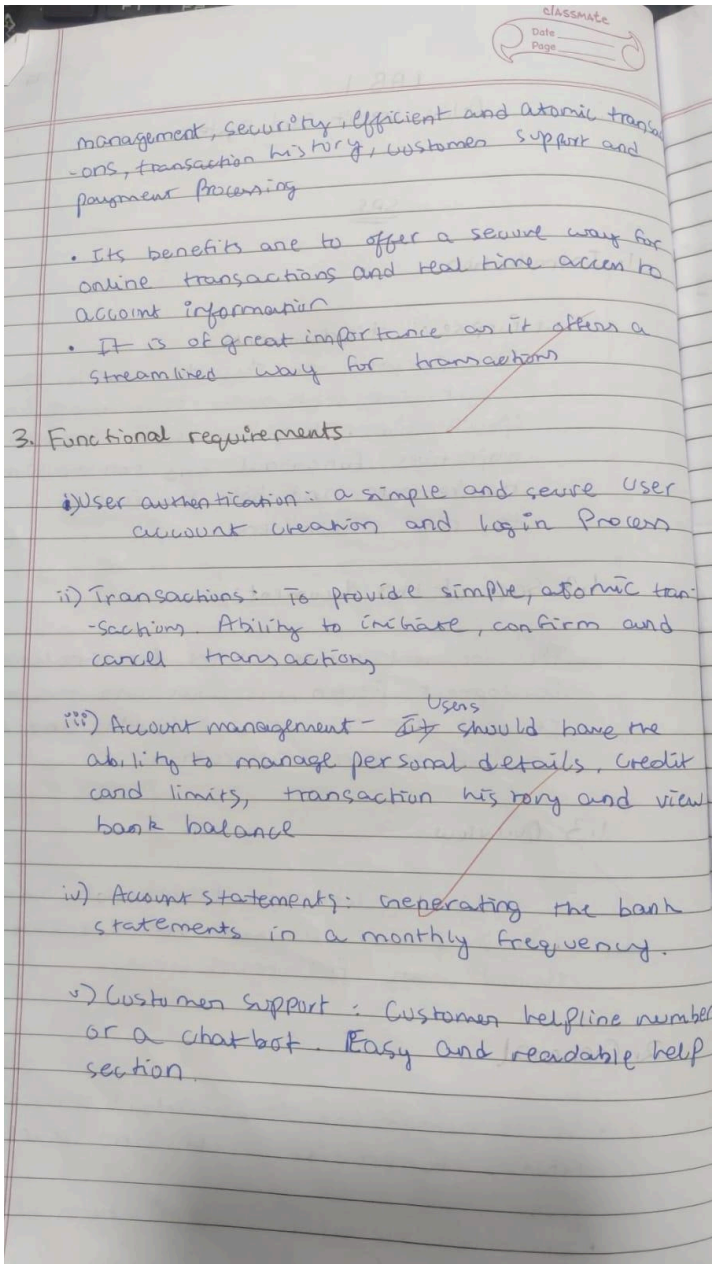
The document covers all the functionalities, development process, use cases and the benefits. It will also show the budgeting and timeline for the resource allocation

1.3 Overview

The online credit card transaction system aims to provide a safe, secure, quick and efficient way for transactions.

2. General

- It is aimed for individuals, businesses and MNC's to provide safe and secure transactions.
- It offers a variety of features i.e account



management, security, efficient and atomic transactions, transaction history, customer support and payment processing

- Its benefits are to offer a secure way for online transactions and real time access to account information
- It is of great importance as it offers a streamlined way for transactions

3. Functional requirements

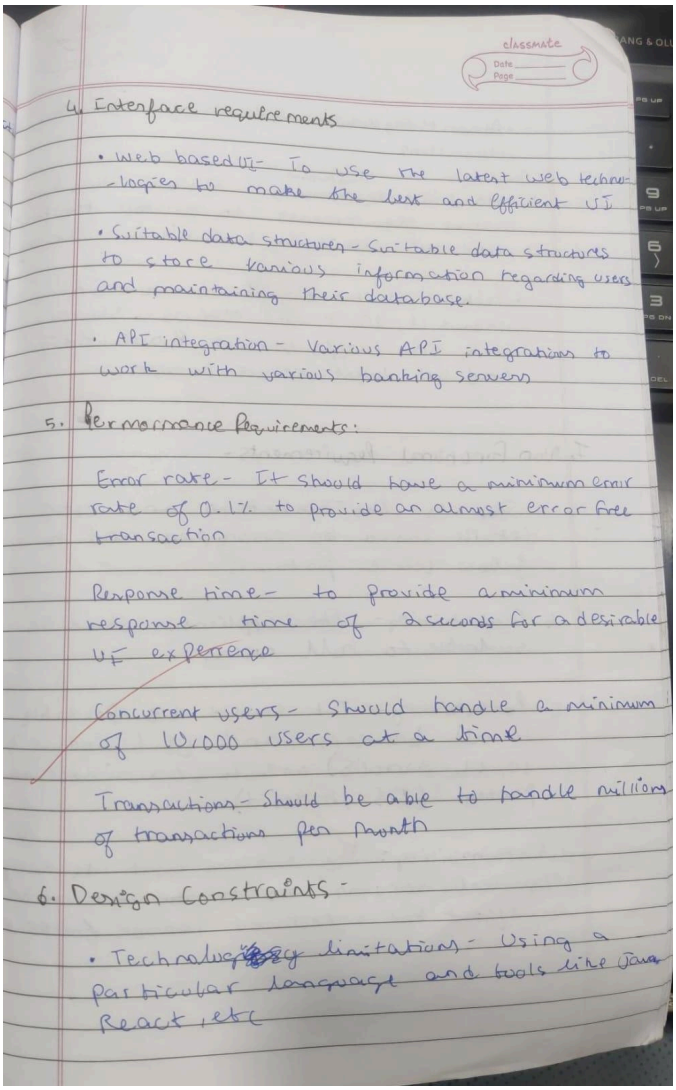
i) User authentication: a simple and secure user account creation and login process

ii) Transactions: To provide simple, atomic transactions. Ability to initiate, confirm and cancel transactions

iii) Account management - Users should have the ability to manage personal details, credit card limits, transaction history and view bank balance

iv) Account statements: Generating the bank statements in a monthly frequency.

v) Customer support: Customer helpline number or a chatbot. Easy and readable help section.



4. Interface requirements

- Web based UI - To use the latest web technologies to make the best and efficient UI
- Suitable data structures - Suitable data structures to store various information regarding users and maintaining their database.
- API integration - Various API integrations to work with various banking servers

5. Performance Requirements:

Error rate - It should have a minimum error rate of 0.1% to provide an almost error free transaction

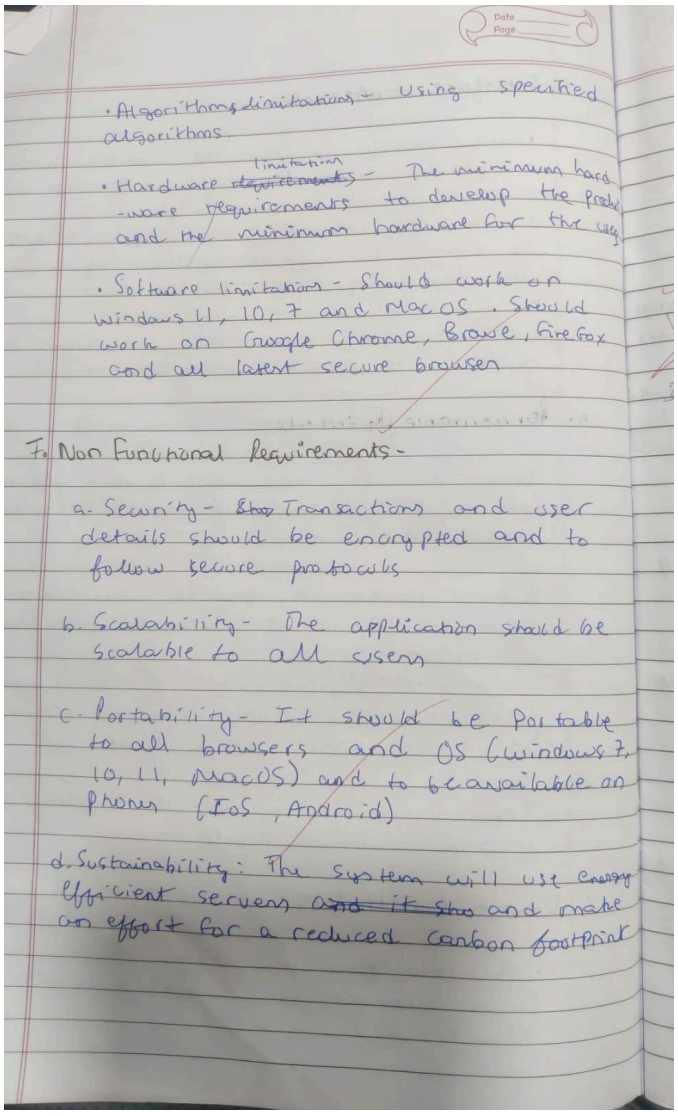
Response time - to provide a minimum response time of 2 seconds for a desirable UI experience

Concurrent users - Should handle a minimum of 10,000 users at a time

Transactions - Should be able to handle millions of transactions per month

6. Design Constraints -

- Technology limitations - Using a particular language and tools like Java React, etc



- Algorithms limitations - Using specified algorithms.
- Hardware limitations - The minimum hardware requirements to develop the prod and the minimum hardware for the use
- Software limitations - Should work on Windows 11, 10, 7 and MacOS. Should work on Google Chrome, Brave, Firefox and all latest secure browsers

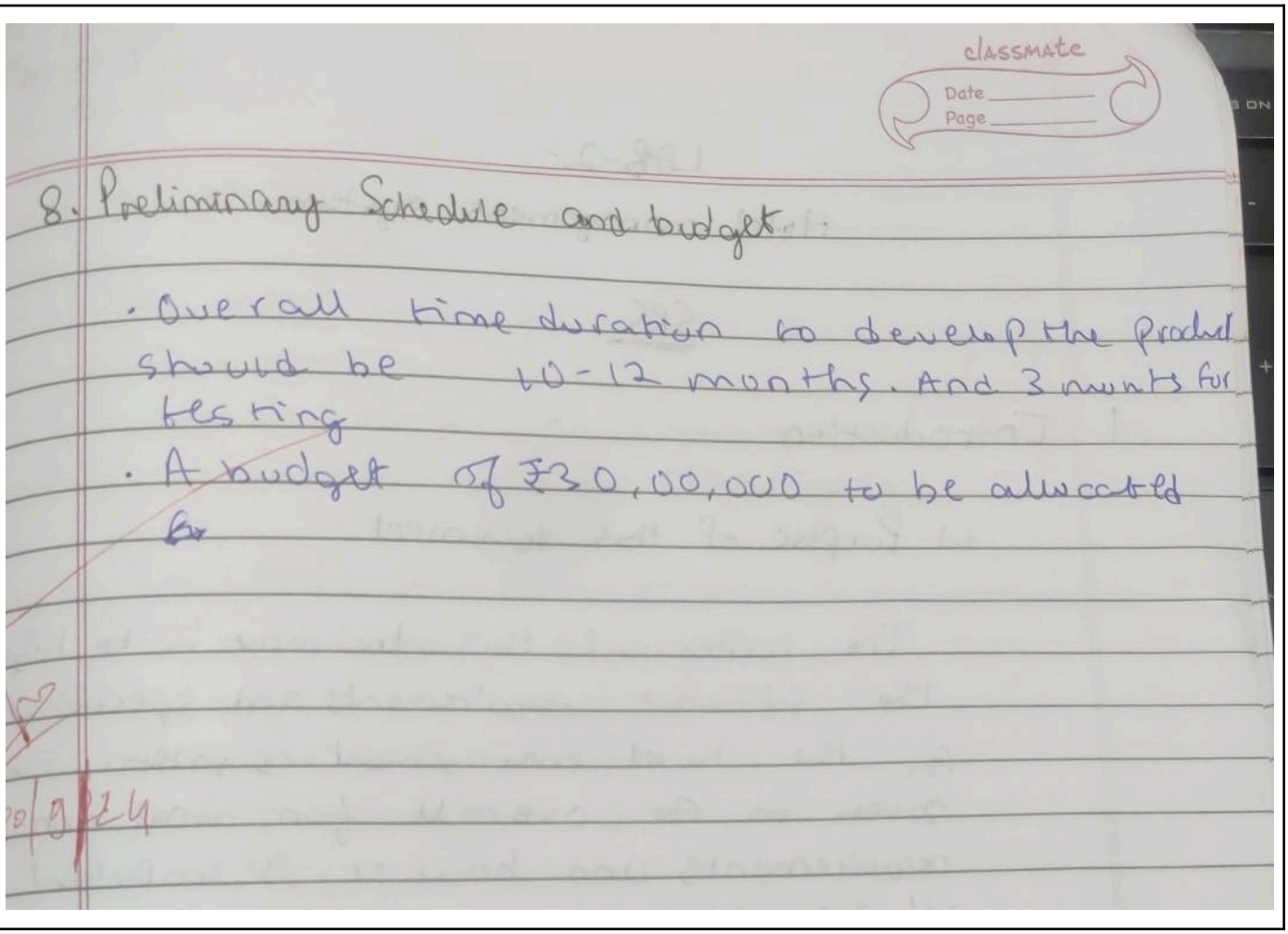
7. Non Functional Requirements -

a. Security - Transactions and user details should be encrypted and to follow secure protocols

b. Scalability - The application should be scalable to all users

c. Portability - It should be portable to all browsers and OS (Windows 7, 10, 11, MacOS) and to be available on phones (IoS, Android)

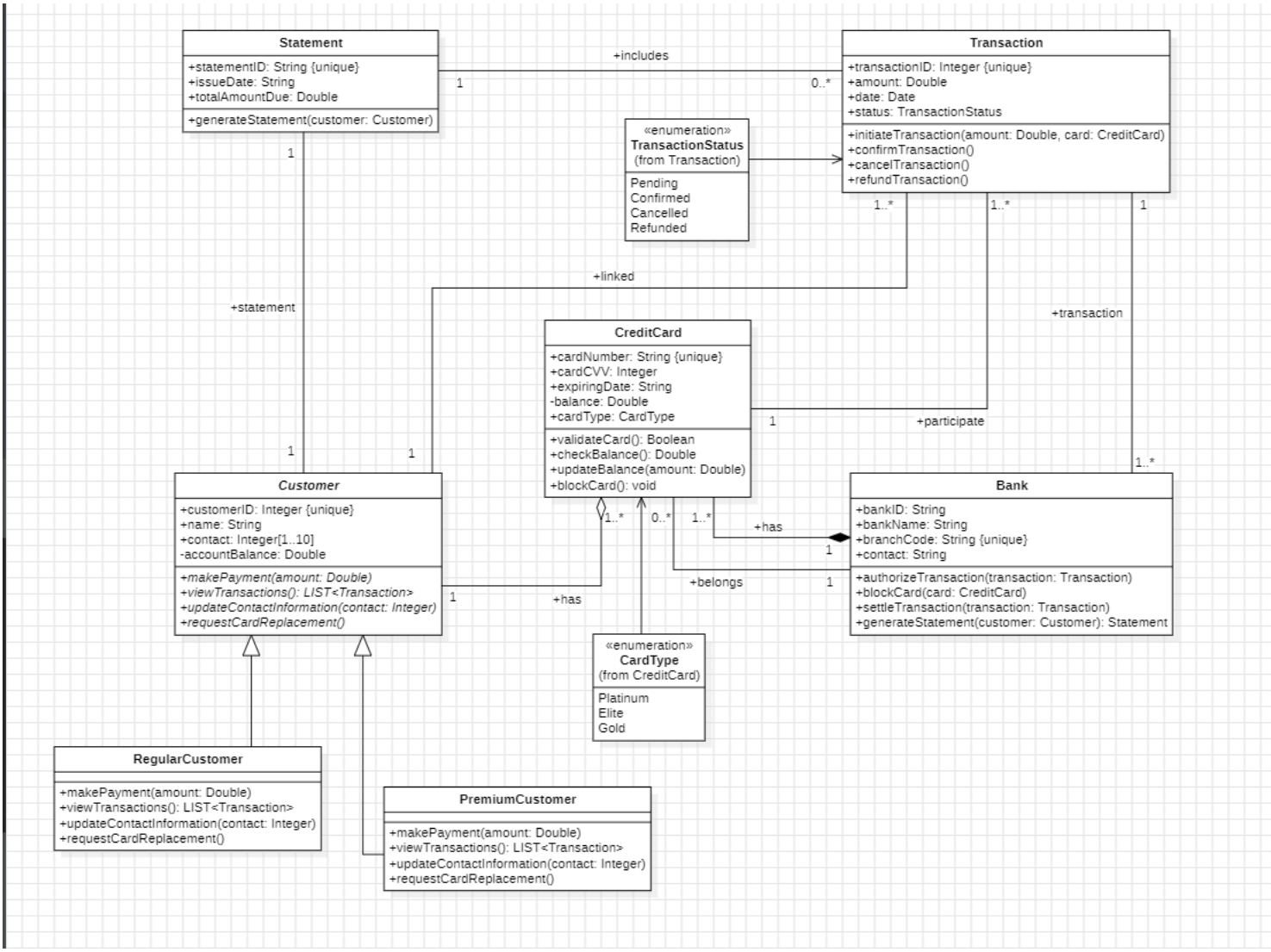
d. Sustainability: The system will use energy efficient servers and make an effort for a reduced carbon footprint



8. Preliminary Schedule and budget.

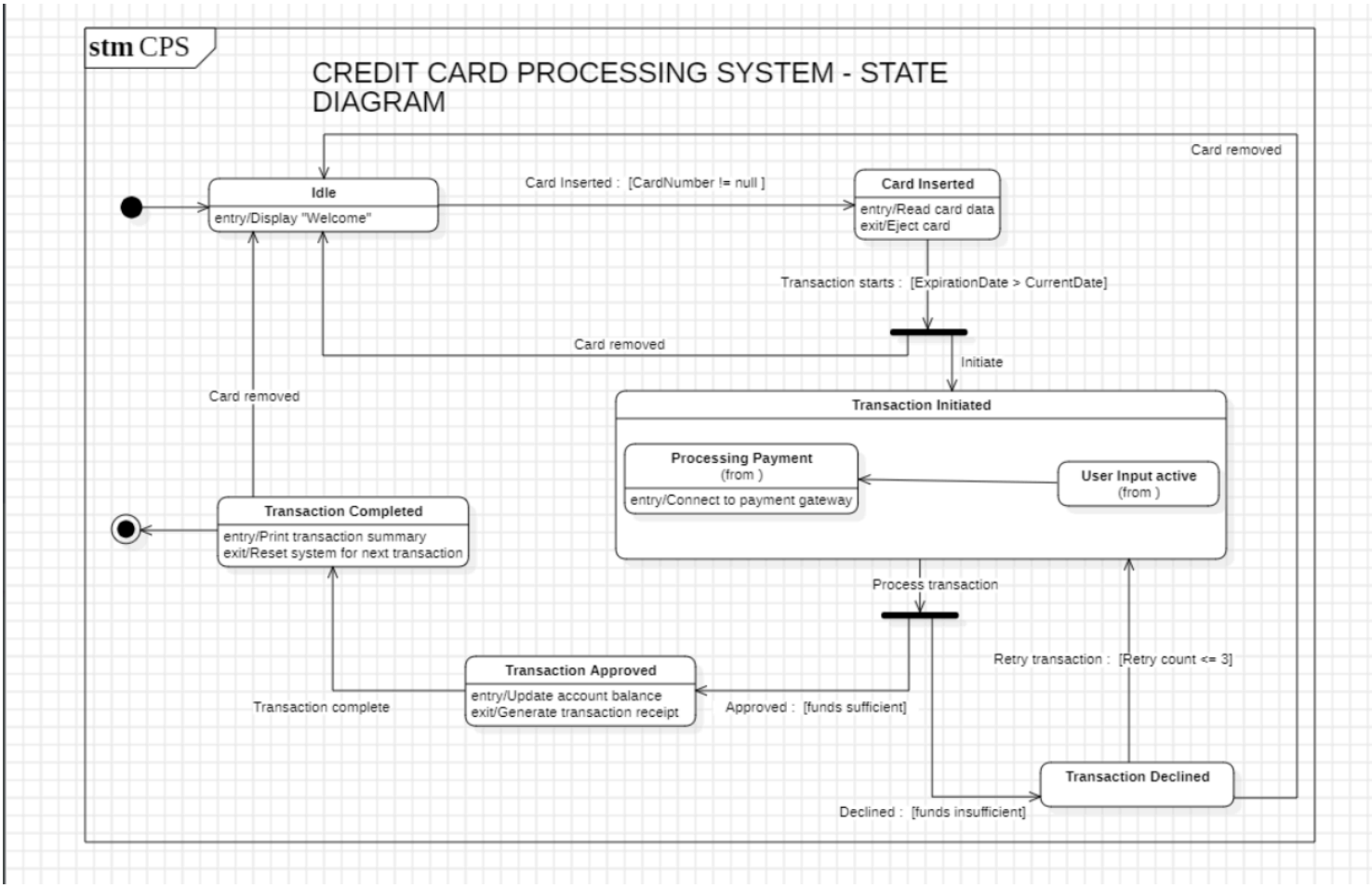
- Overall time duration to develop the product should be 10-12 months. And 3 months for testing
- A budget of ₹30,00,000 to be allocated

CLASS DIAGRAM



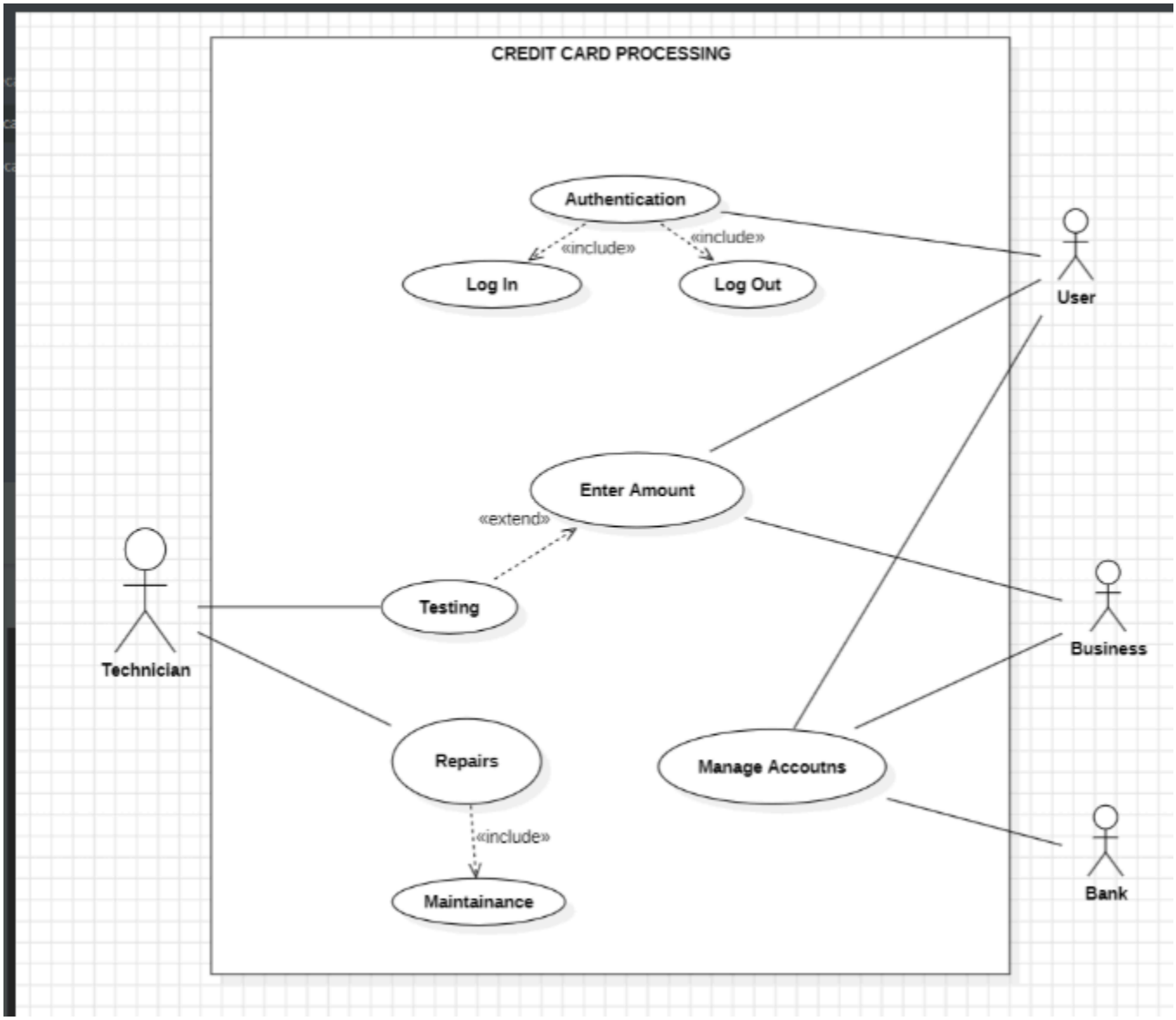
This UML class diagram illustrates the core components of a financial system. It depicts entities such as Customer, CreditCard, Transaction, Statement, and Bank, along with their attributes and relationships. Key features include transaction management, statement generation, and customer categorization based on their spending habits. The diagram also includes enumerations for transaction status, card type, and customer rank.

STATE DIAGRAM



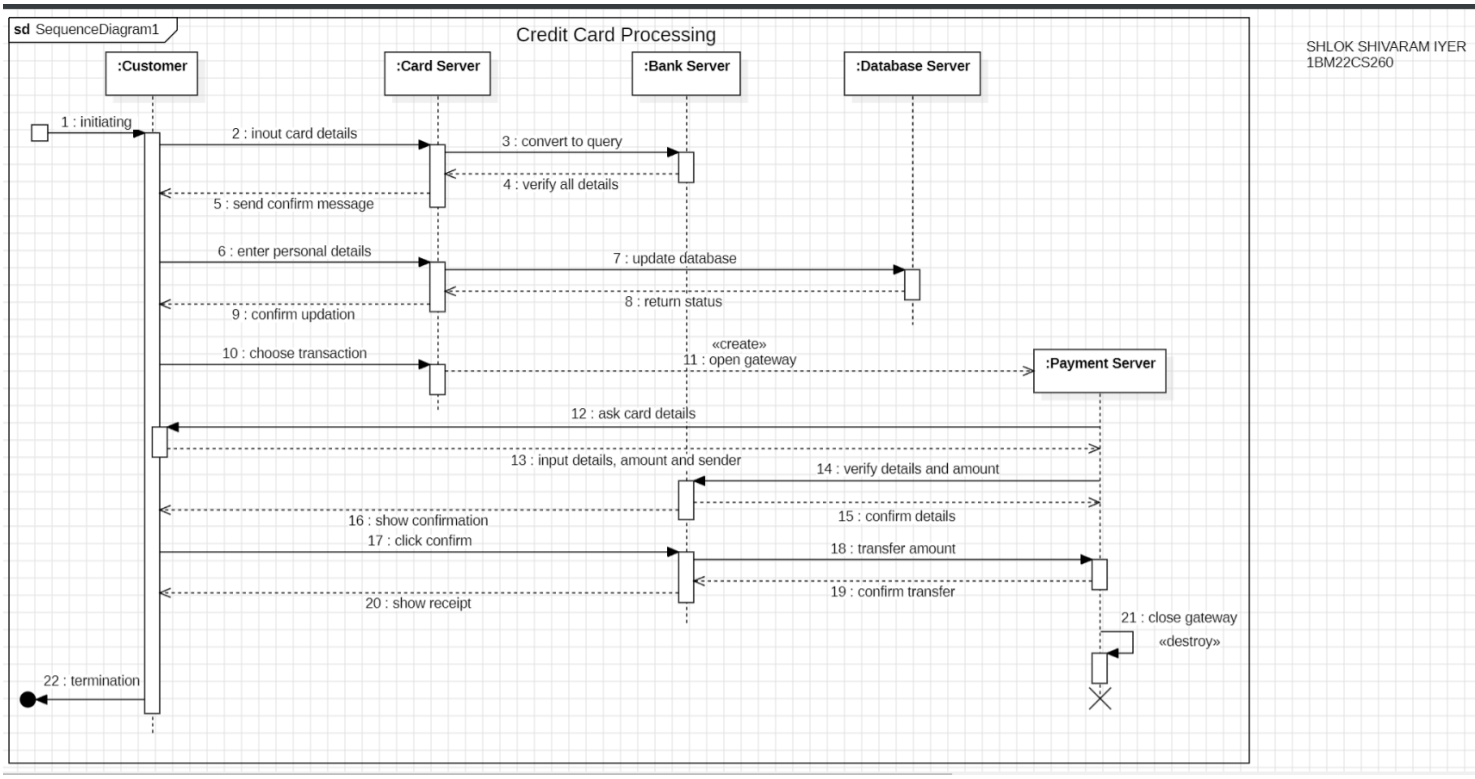
This state diagram models a credit card processing system. It shows the system transitioning through states like Idle, Transaction Initiated, Transaction Processing, and either Approved or Declined. The system processes card data, verifies transactions, and updates account balances accordingly.

USE CASE DIAGRAM



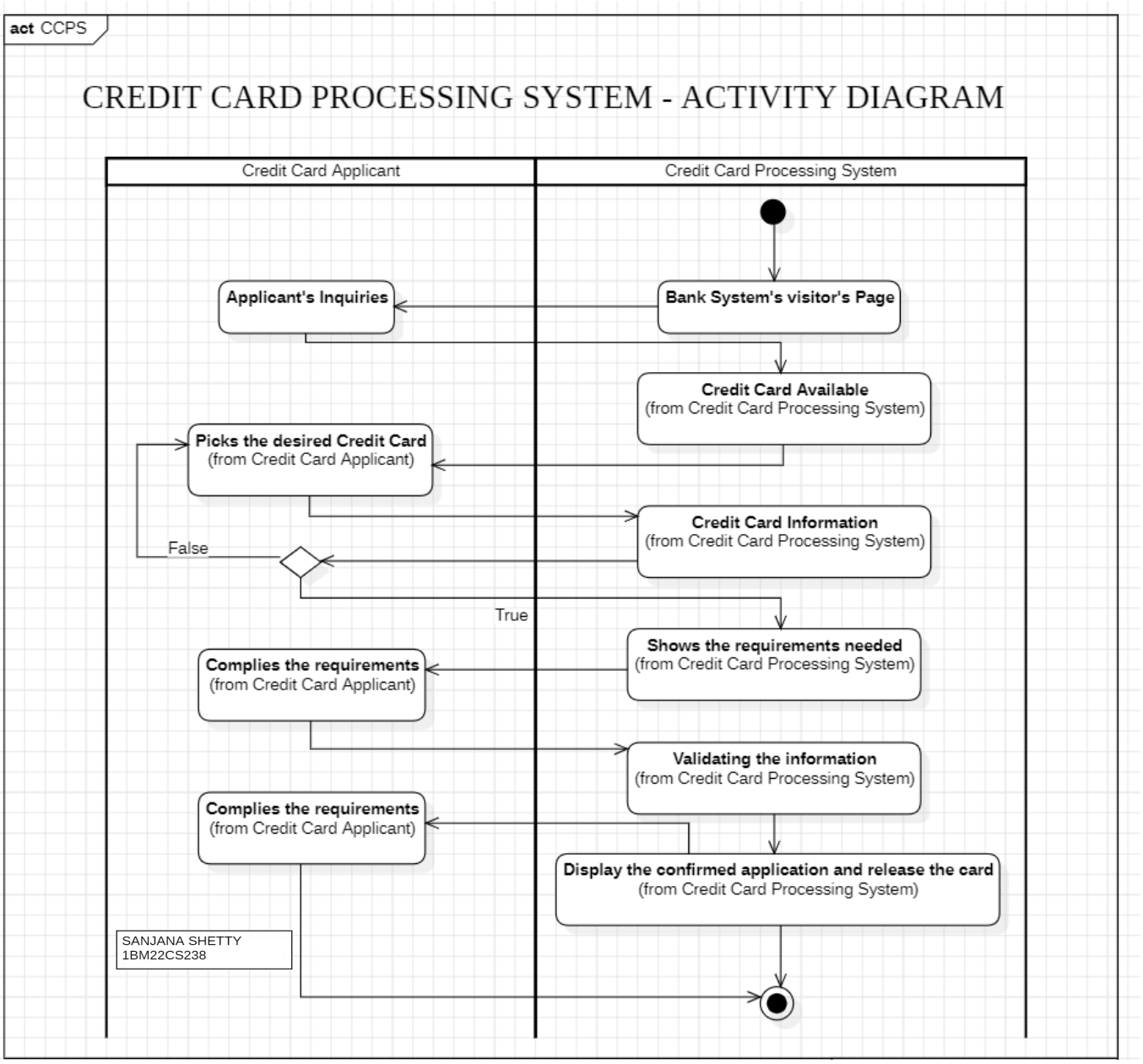
This UML use case diagram outlines the functionalities of a CreditCardProcessing system. It shows various actors like Cardholder (Personal and Business), Bank, Admin, Merchant, and Payment Gateway interacting with the system through different use cases. Key features include authorizing transactions, validating cards, processing payments, issuing cards, detecting fraud, generating statements, and managing refunds. The diagram also utilizes relationships like <<include>> and <<extend>> to represent dependencies between use cases.

SEQUENCE DIAGRAM



This UML sequence diagram models the functionalities of a credit card processing system. It shows how various actors, including cardholders, banks, merchants, and administrators, interact with the system. Key use cases include authorizing transactions, validating cards, processing payments, issuing cards, detecting fraud, generating statements, and managing refunds. The diagram also incorporates relationships like <<include>> and <<extend>> to represent dependencies between use cases, providing a more comprehensive view of the system's behavior.

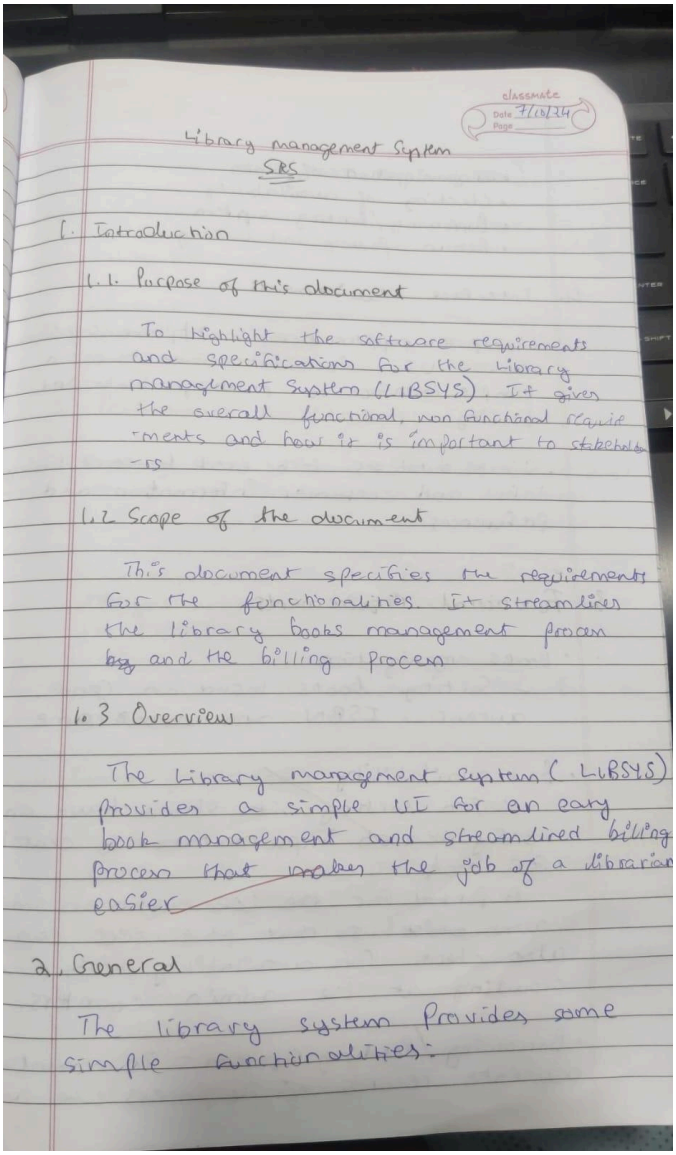
ACTIVITY DIAGRAM



This activity diagram models the process of a credit card application. It starts with the applicant making inquiries, followed by the system displaying available credit card options. The applicant then selects a desired card. The system checks if the applicant meets the requirements for the selected card. If they do, the system validates the information provided by the applicant. Finally, if the validation is successful, the system confirms the application and releases the credit card. If the applicant fails to meet the requirements at any stage, the system displays the missing requirements.

LIBRARY MANAGEMENT SYSTEM

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)



Library management System
SRS

1. Introduction

1.1. Purpose of this document

To highlight the software requirements and specifications for the Library management System (LIBSYS). It given the overall functional, non functional requirements and how it is important to stakeholders

1.2 Scope of the document

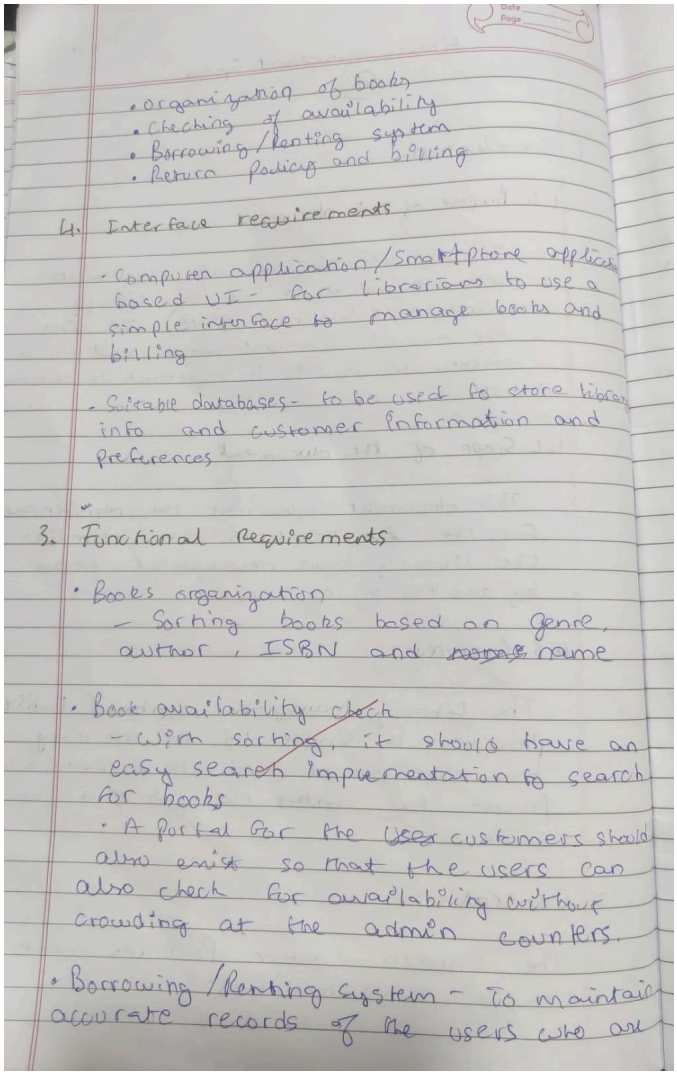
This document specifies the requirements for the functionalities. It streamlines the library books management proces and the billing procesn

1.3 Overview

The library management system (LIBSYS) provides a simple UI for an easy book management and streamlined billing process that makes the job of a librarian easier

2. General

The library system Provides some simple functionalities:



- organization of books
- Checking of availability
- Borrowing/Renting system
- Return policy and billing

4. Interface requirements

- Computer application/Smartphone application based UI - for Librarians to use a simple interface to manage books and billing

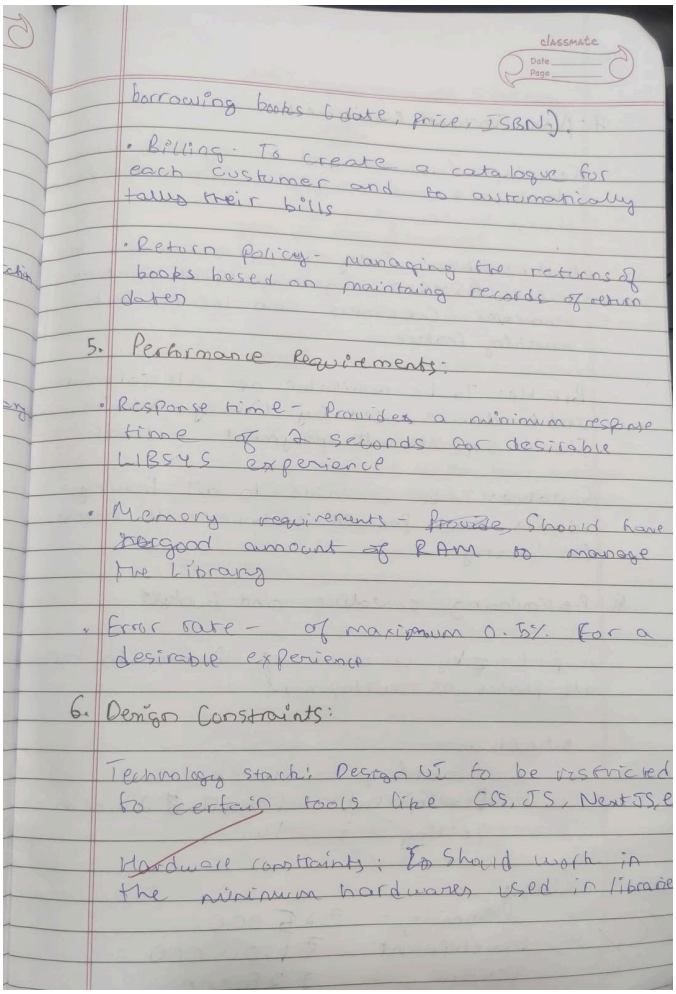
- Suitable databases - to be used to store library info and customer information and preferences

3. Functional Requirements

- Books organization
  - Sorting books based on genre, author, ISBN and name

- Book availability check
  - With sorting, it should have an easy search implementation to search for books
  - A portal for the customers should also exist so that the users can also check for availability without crowding at the admin counters.

- Borrowing/Renting system - To maintain accurate records of the users who are



borrowing books (date, price, ISBN).

- Billing: To create a catalogue for each customer and to automatically tally their bills

- Return policy: Managing the returns of books based on maintaing records of return dates

5. Performance Requirements:

- Response time: Provides a minimum response time of 2 seconds for desirable LIBSYS experience

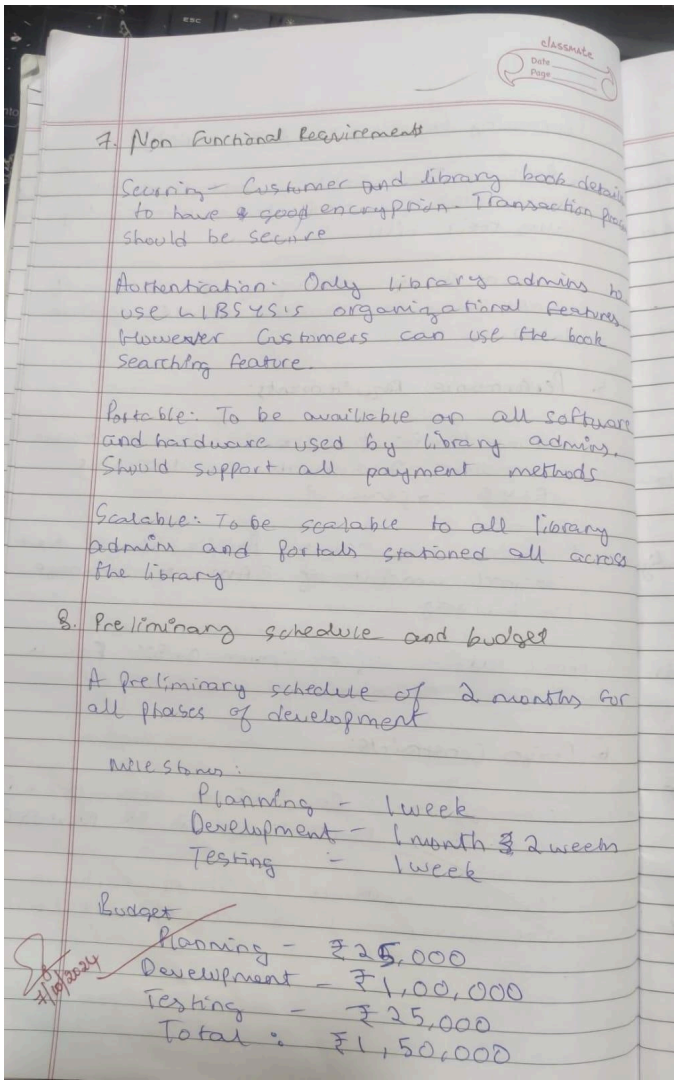
- Memory requirements: Should have good amount of RAM to manage the library

- Error rate: of maximum 0.5%. For a desirable experience

6. Design Constraints:

Technology stack: Design UI to be restricted to certain tools like CSS, JS, NextJS, e

Hardware constraints: It should work in the minimum hardwares used in libraries



7. Non Functional Requirements

Security: Customer and library book details to have good encryption. Transaction process should be secure

Authentication: Only library admins to use LIBSYS's organizational features. However customers can use the book searching feature.

Portable: To be available on all software and hardware used by library admins. Should support all payment methods

Scalable: To be scalable to all library admins and portals stationed all across the library

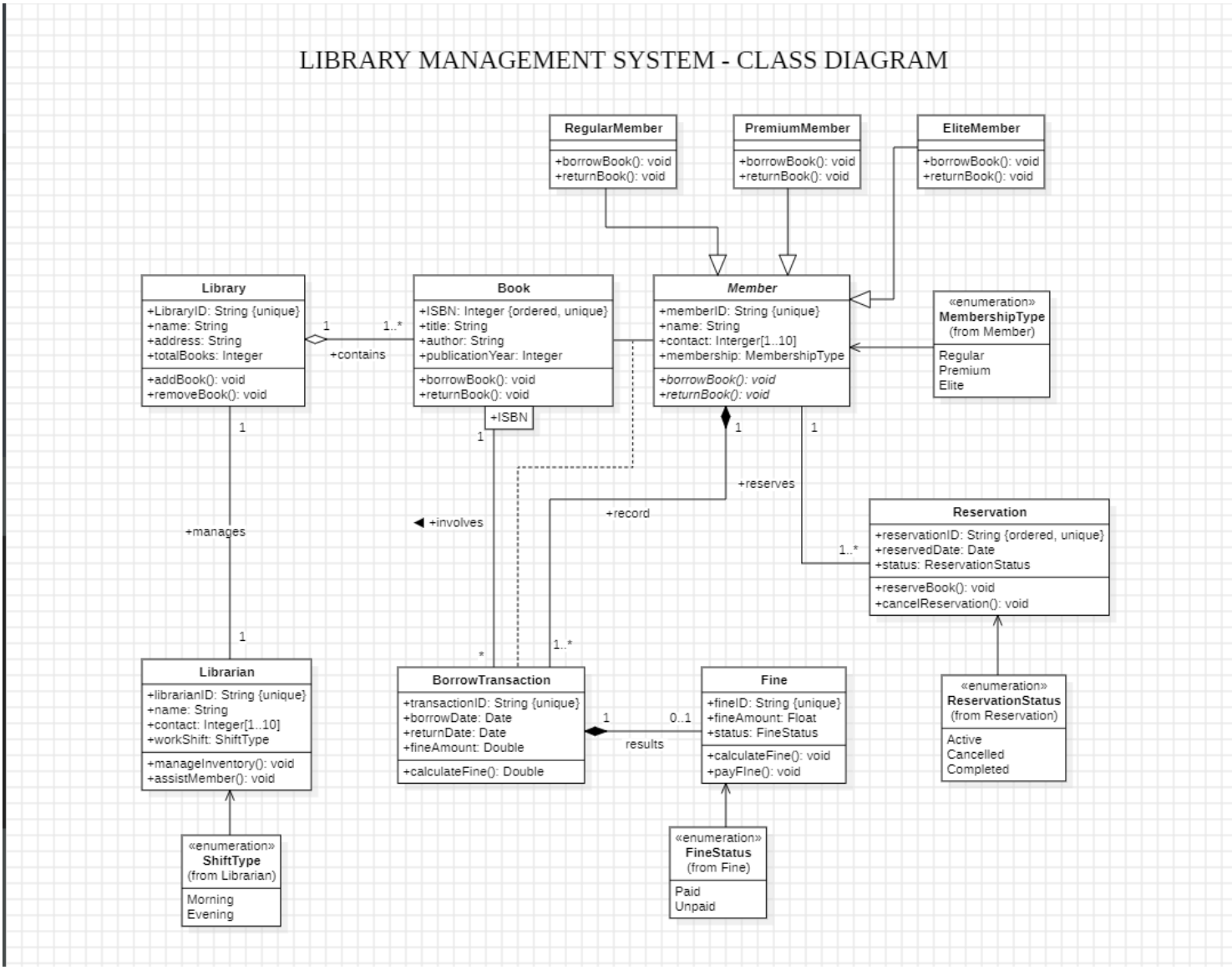
8. Preliminary schedule and budget

A preliminary schedule of 2 months for all phases of development

Milestones:
Planning - 1week
Development - 1month & 2 weeks
Testing - 1week

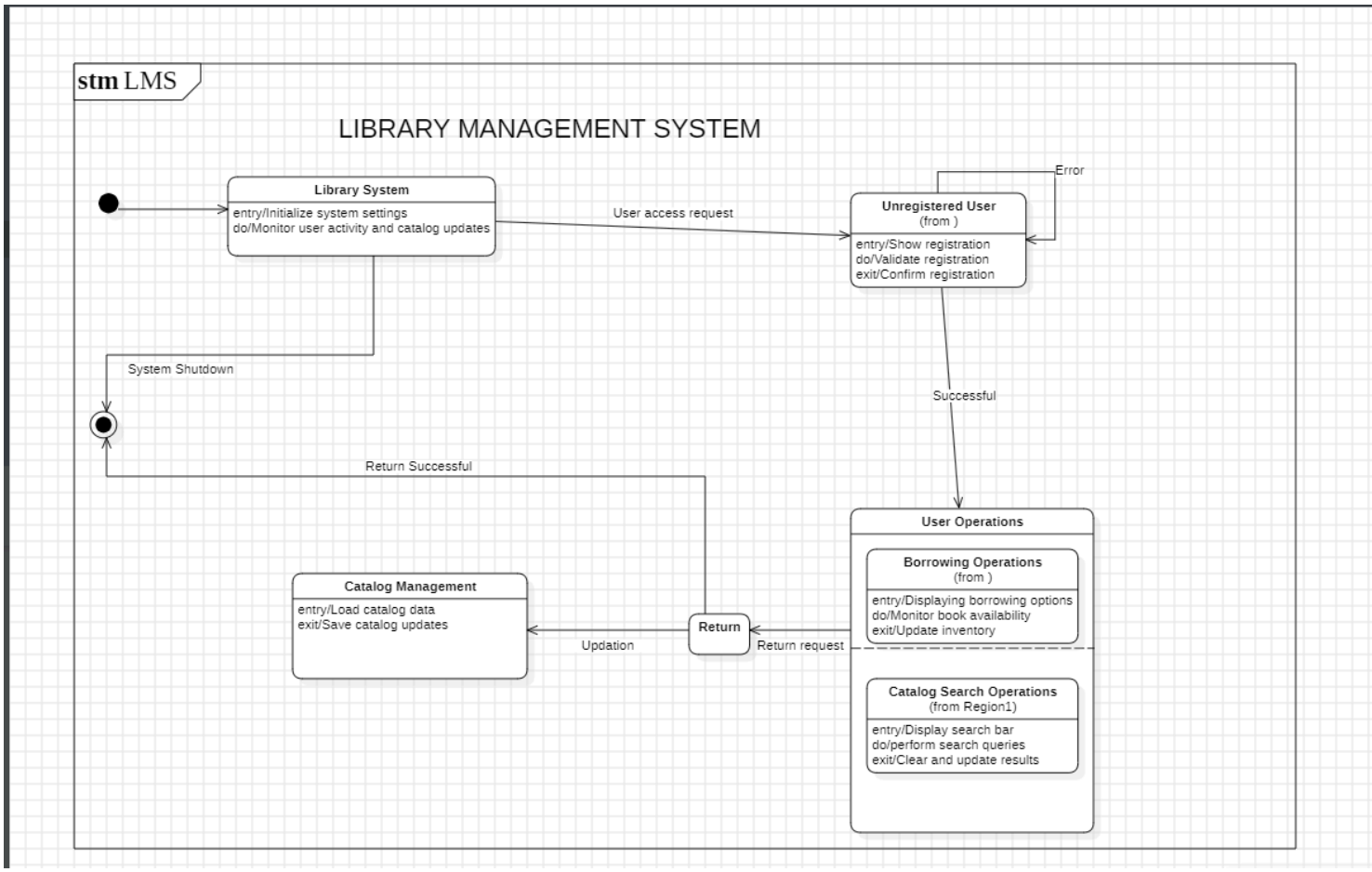
Budget
Planning - ₹25,000
Development - ₹1,00,000
Testing - ₹25,000
Total : ₹1,50,000

CLASS DIAGRAM



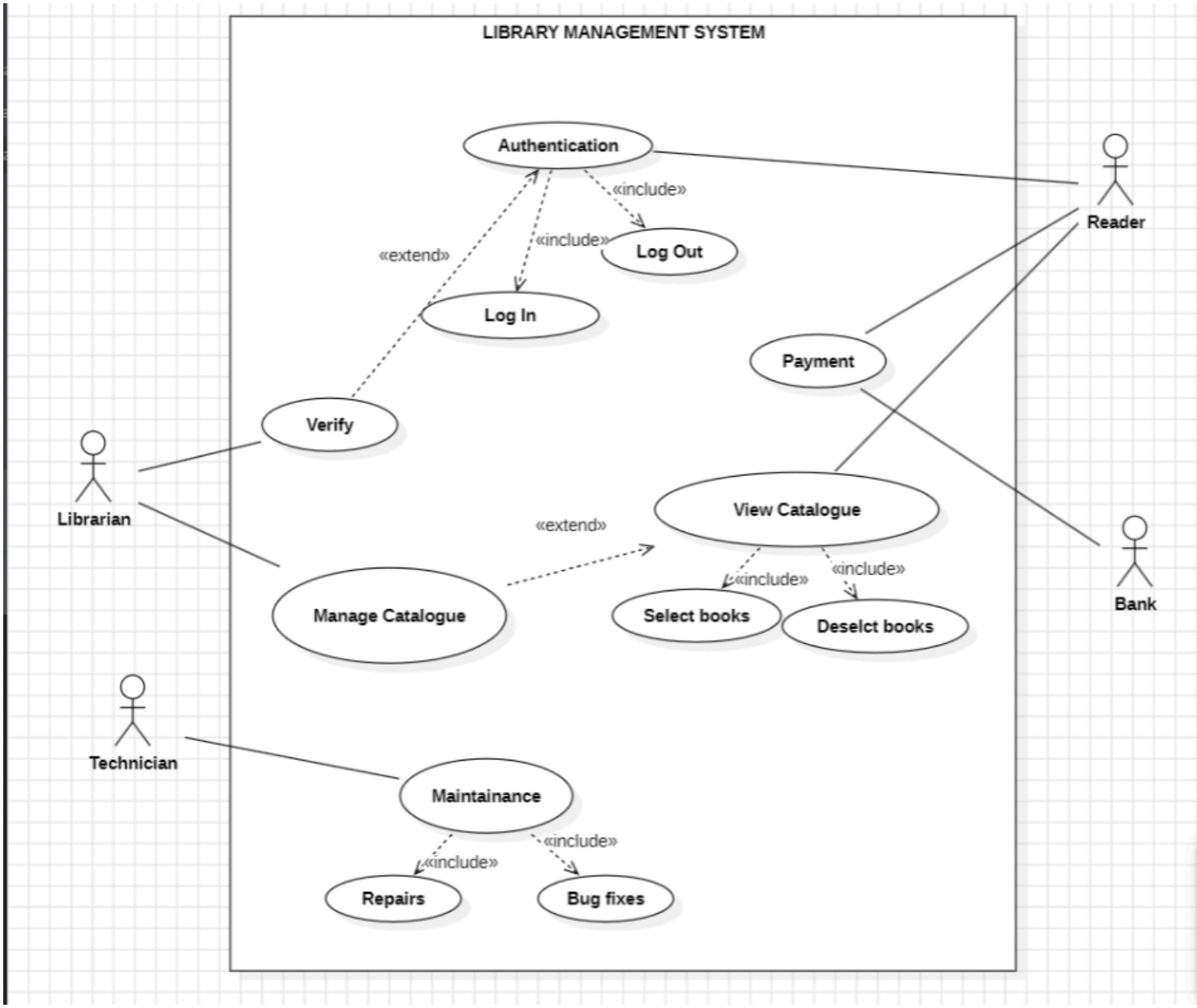
This UML class diagram illustrates the core components of a library management system. It depicts entities such as Library, Book, Member, Librarian, and their relationships. Key features include book reservations, borrowing transactions, and fine management. The diagram also includes enumerations for different member types, shifts, and statuses.

STATE DIAGRAM



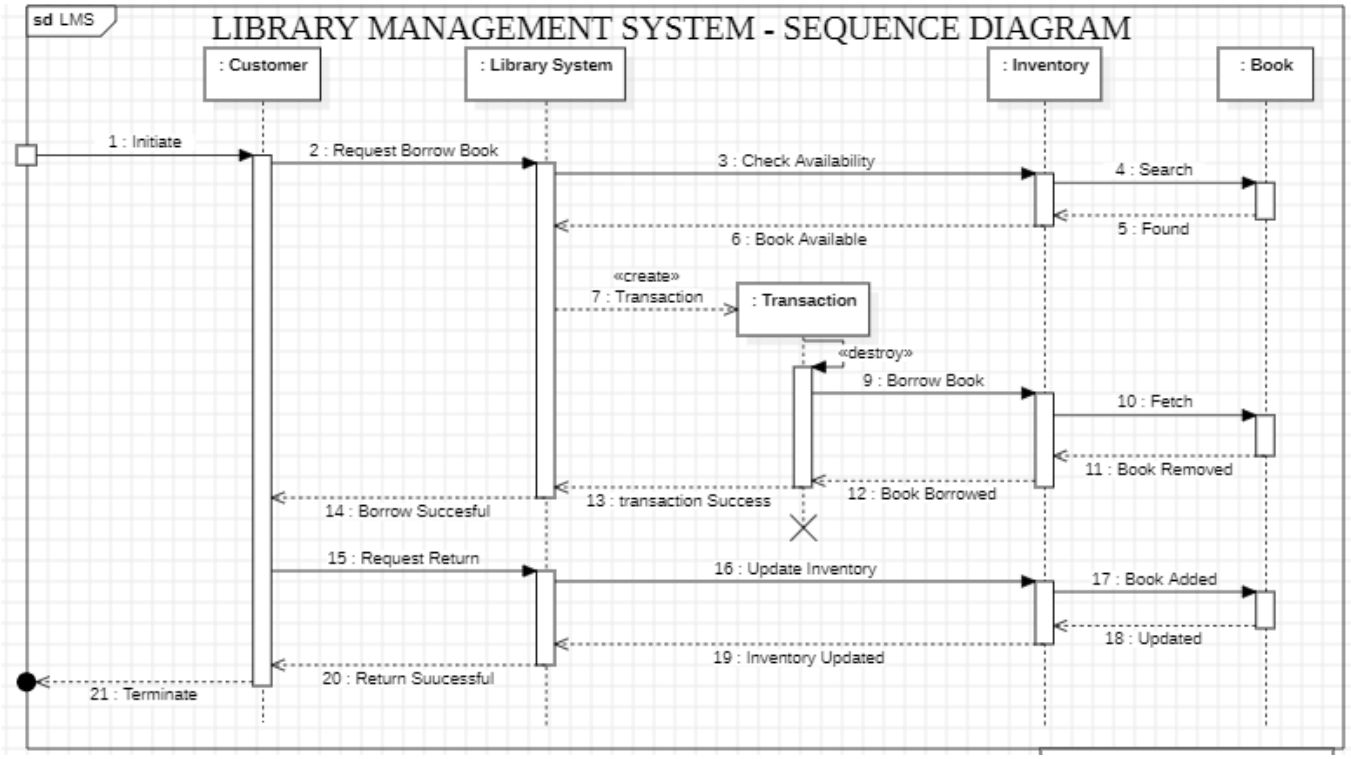
This state diagram illustrates the workflow of a library management system. It starts with the system initializing settings and monitoring user activity. Users can register and log in to access operations like catalog management, book management, borrowing operations, and catalog search. The system handles various states, including user registration, login, catalog loading, book management, search operations, and user operations. The diagram also includes error handling for unsuccessful registration or login attempts.

USE CASE DIAGRAM



This UML use case diagram depicts the functionalities of a Library Management System. It shows how different actors, such as Users (Students and Staff), Library, Library Database, and Suppliers, interact with the system. Key use cases include Authentication, Book-related operations (Requesting, Reserving, Paying fines), Feedback, User Registration, and Library Management tasks like maintaining the database and managing books. The diagram also utilizes relationships like <<include>> and <<extend>> to represent dependencies between use cases.

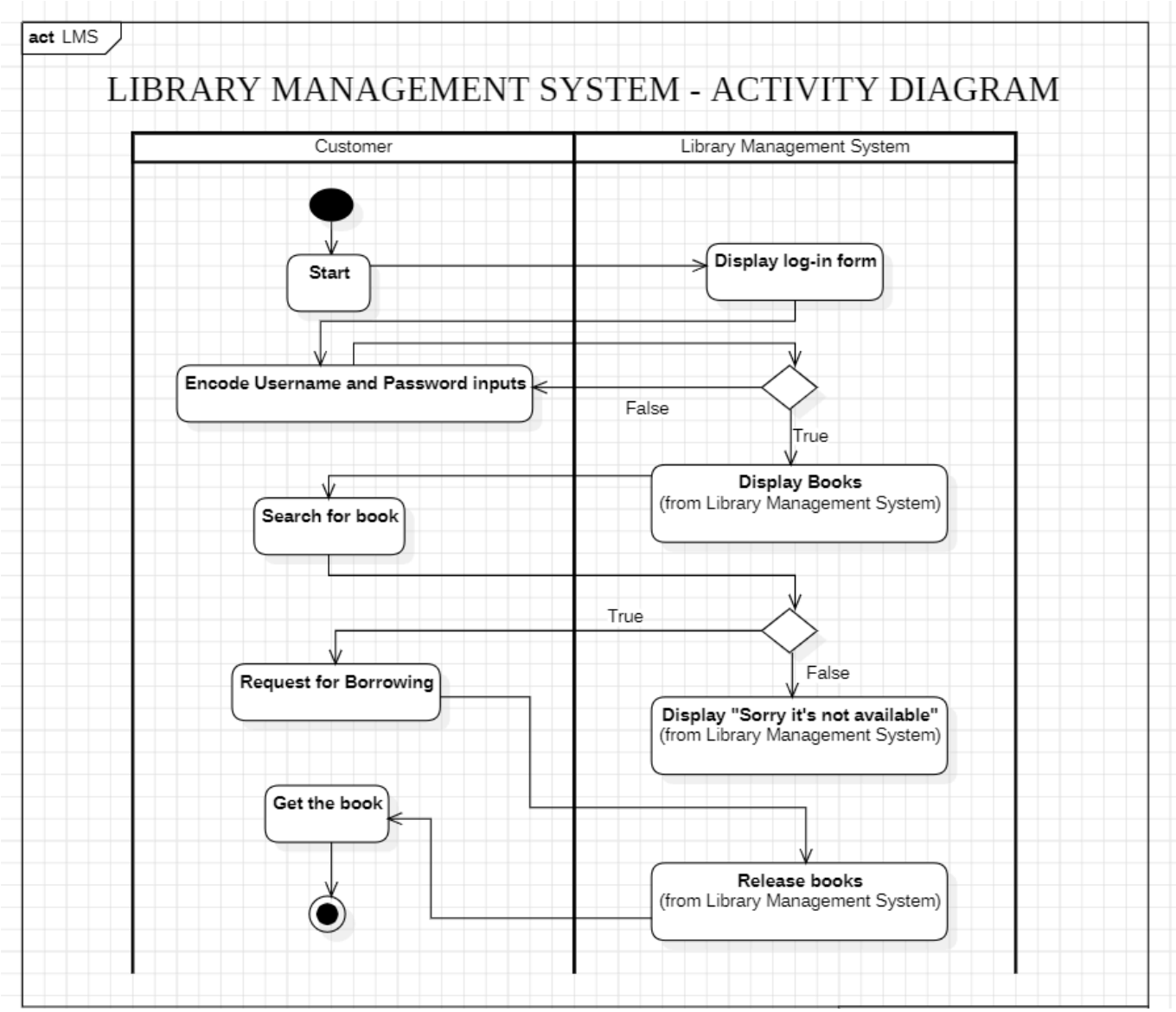
SEQUENCE DIAGRAM



This sequence diagram represents a **Library Management System**, illustrating interactions between the **Customer**, **Library System**, **Inventory**, **Book**, and a **Transaction**.

1. A customer initiates a request to borrow a book, which is processed by the library system by checking availability in the inventory.
2. If the book is available, a transaction is created, and the book is fetched and removed from inventory.
3. Upon returning the book, the library system updates the inventory by re-adding the book.
4. The system concludes the process with successful borrow and return operations.

ACTIVITY DIAGRAM

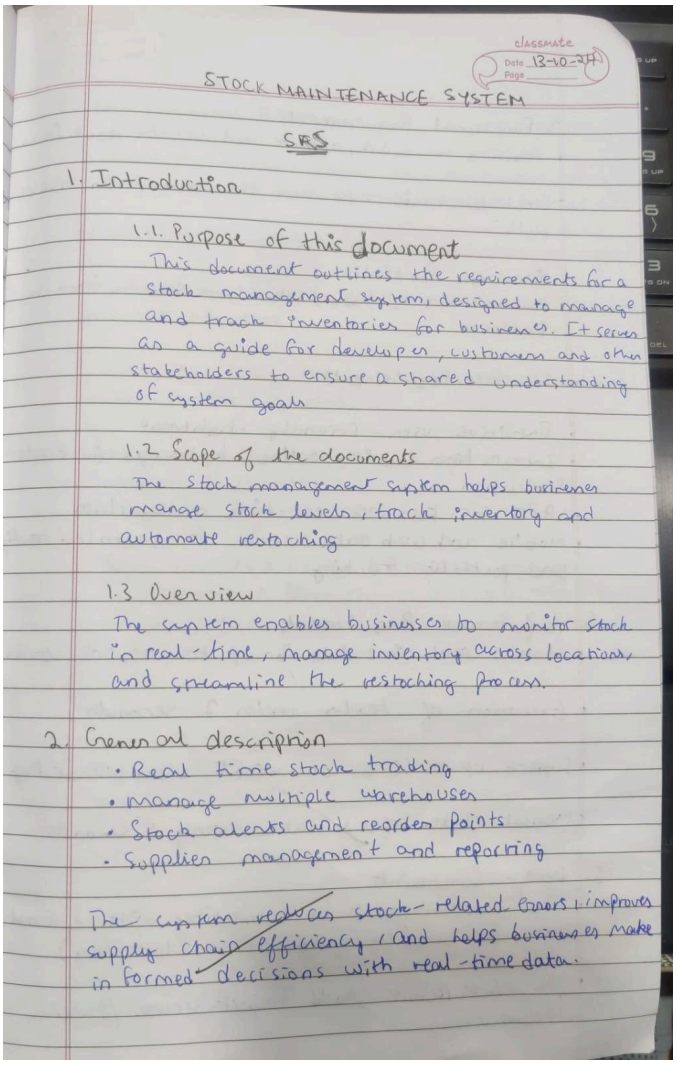


This activity diagram outlines the Library Management System process for borrowing a book, detailing actions between the Customer and the Library Management System.

1. The process starts with the customer logging in by encoding their username and password, which the system verifies.
2. The customer searches for a book, and the system displays available options.
3. If the book is available, the customer requests borrowing, and the system releases the book.
4. If the book is not available, the system displays a "Sorry, it's not available" message, ending the activity.

STOCK MAINTENANCE SYSTEM

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)



STOCK MAINTENANCE SYSTEM

SRS

1. Introduction

1.1. Purpose of this document

This document outlines the requirements for a stock management system, designed to manage and track inventories for businesses. It serves as a guide for developers, customers and other stakeholders to ensure a shared understanding of system goals

1.2 Scope of the documents

The Stock management system helps businesses manage stock levels, track inventory and automate restocking

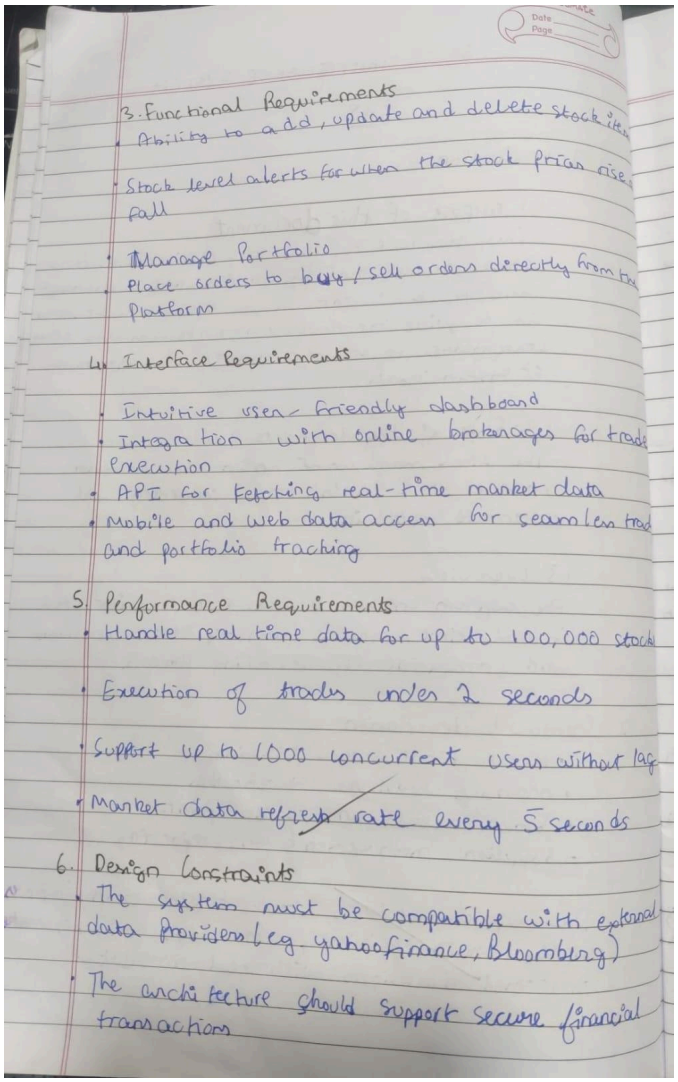
1.3 Overview

The system enables businesses to monitor stock in real-time, manage inventory across locations, and streamline the restocking process.

2. General description

- Real time stock trading
- manage multiple warehouses
- Stock alerts and reorder points
- Supplier management and reporting

The system reduces stock-related errors, improves supply chain efficiency, and helps businesses make informed decisions with real-time data.



3. Functional Requirements

- Ability to add, update and delete stock items
- Stock level alerts for when the stock price rise/fall
- Manage Portfolio
- Place orders to buy/sell orders directly from the platform

4. Interface Requirements

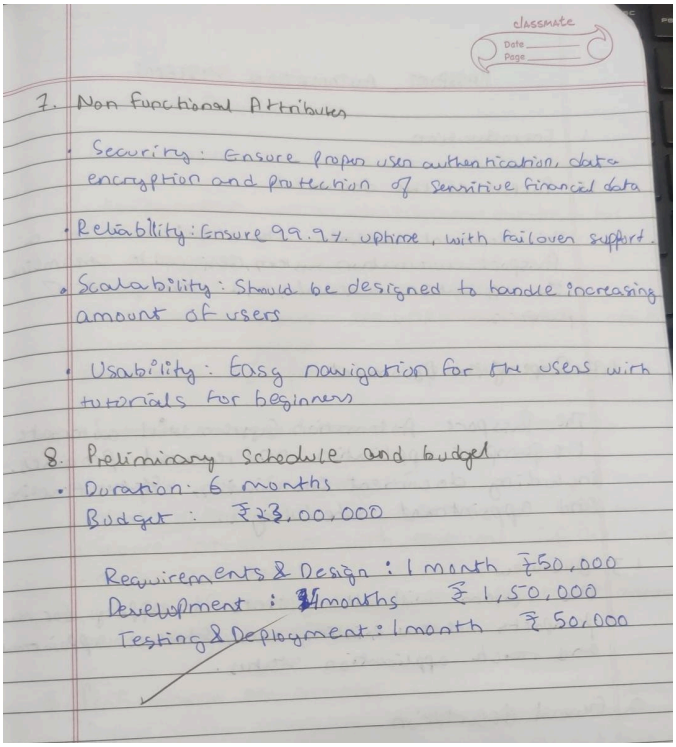
- Intuitive user-friendly dashboard
- Integration with online brokerages for trade execution
- API for fetching real-time market data
- Mobile and web data access for seamless trad and portfolio tracking

5. Performance Requirements

- Handle real time data for up to 100,000 stock
- Execution of trades under 2 seconds
- Support up to 1000 concurrent users without lag
- Market data refresh rate every 5 seconds

6. Design Constraints

- The system must be compatible with external data providers (eg. yahoofinance, Bloomberg)
- The architecture should support secure financial transactions



7. Non Functional Attributes

- Security: Ensure proper user authentication, data encryption and protection of sensitive financial data
- Reliability: Ensure 99.9% uptime, with failover support.
- Scalability: Should be designed to handle increasing amount of users
- Usability: Easy navigation for the users with tutorials for beginners

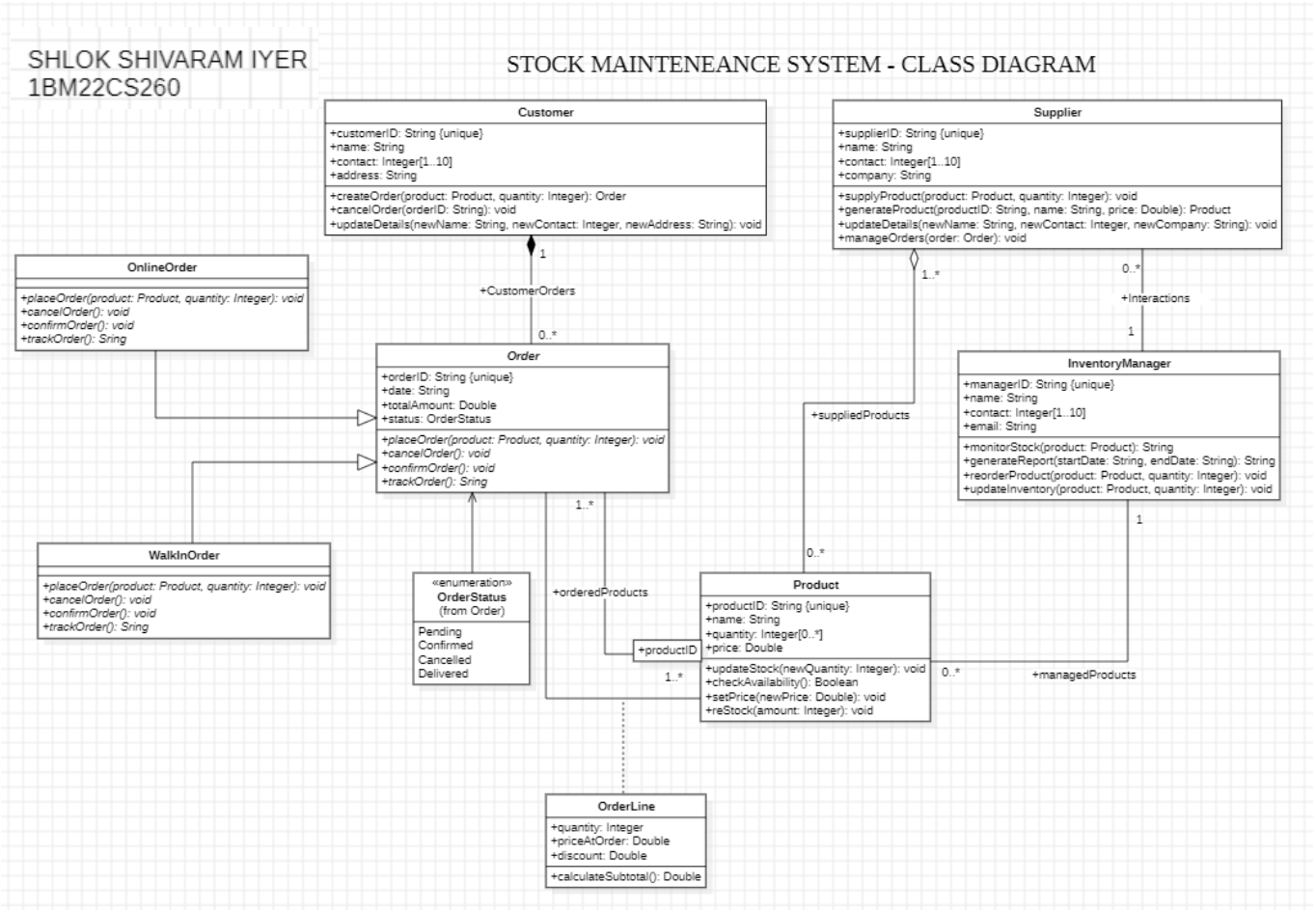
8. Preliminary schedule and budget

- Duration: 6 months

Budget : ₹3,00,000

Requirements & Design : 1 month ₹50,000
Development : 4 months ₹ 1,50,000
Testing & Deployment : 1 month ₹ 50,000

CLASS DIAGRAM

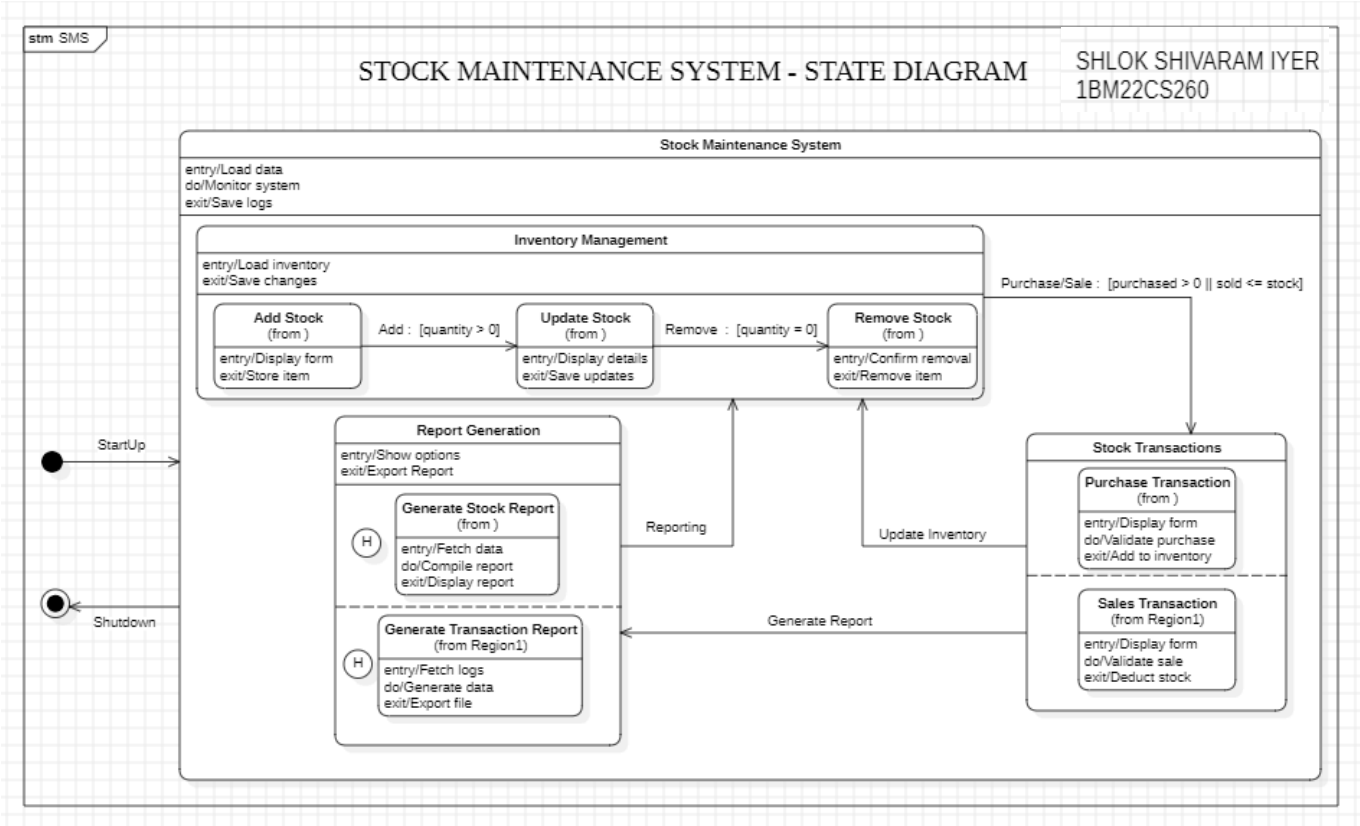


This class diagram represents a Stock Maintenance System, detailing the relationships and functionalities of its main entities.

1. Customer: Can place, cancel, or track orders (either online or walk-in).
2. Supplier: Supplies products to the system, manages orders, and updates product details.
3. Order: Tracks order details, including status (Pending, Confirmed, Cancelled, Delivered) and associated items.
4. Inventory Manager: Monitors stock, generates reports, updates inventory, and manages product availability.

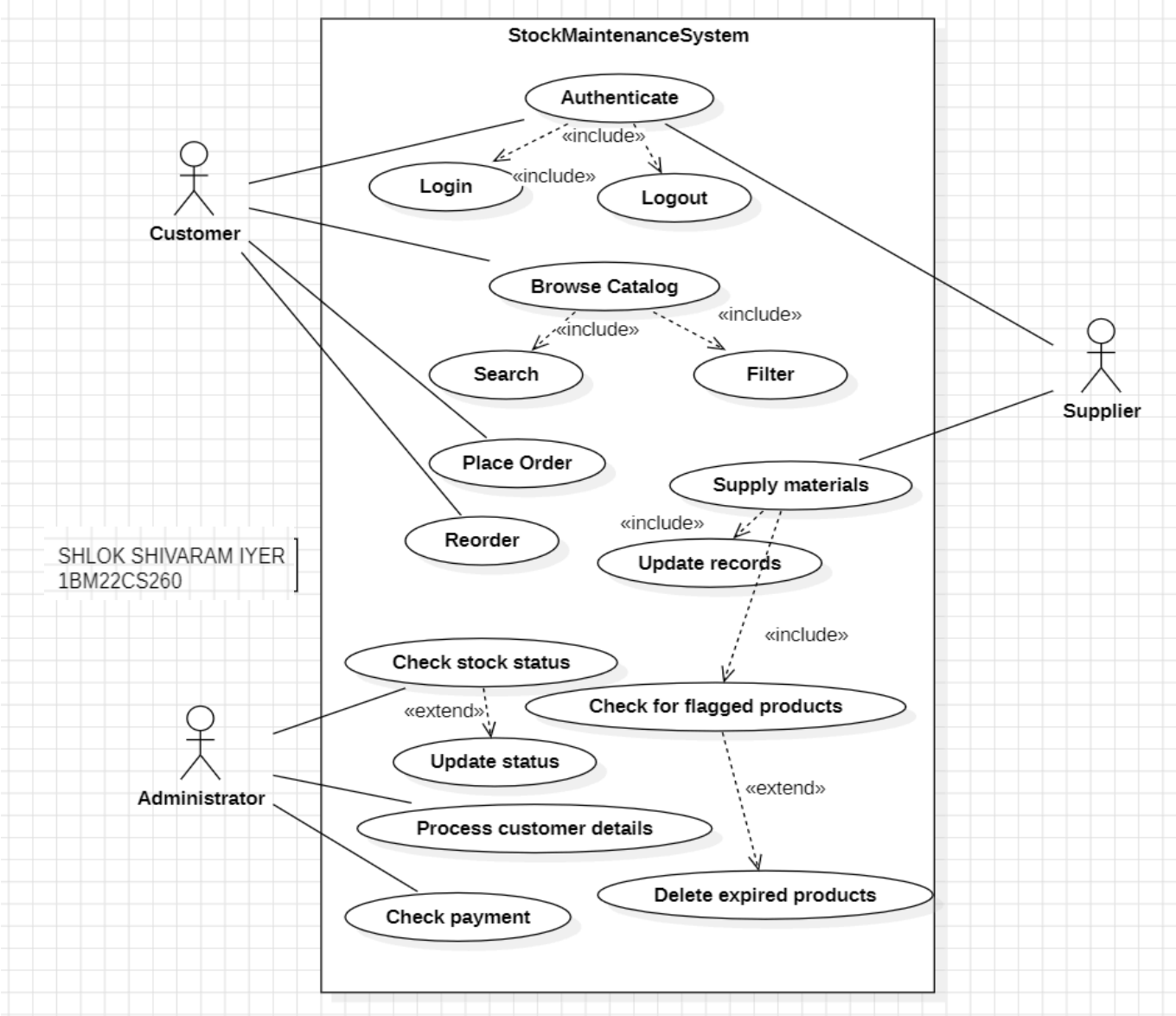
Key entities include Product, linked to orders and inventory, and OrderLine, which details individual order items.

STATE DIAGRAM



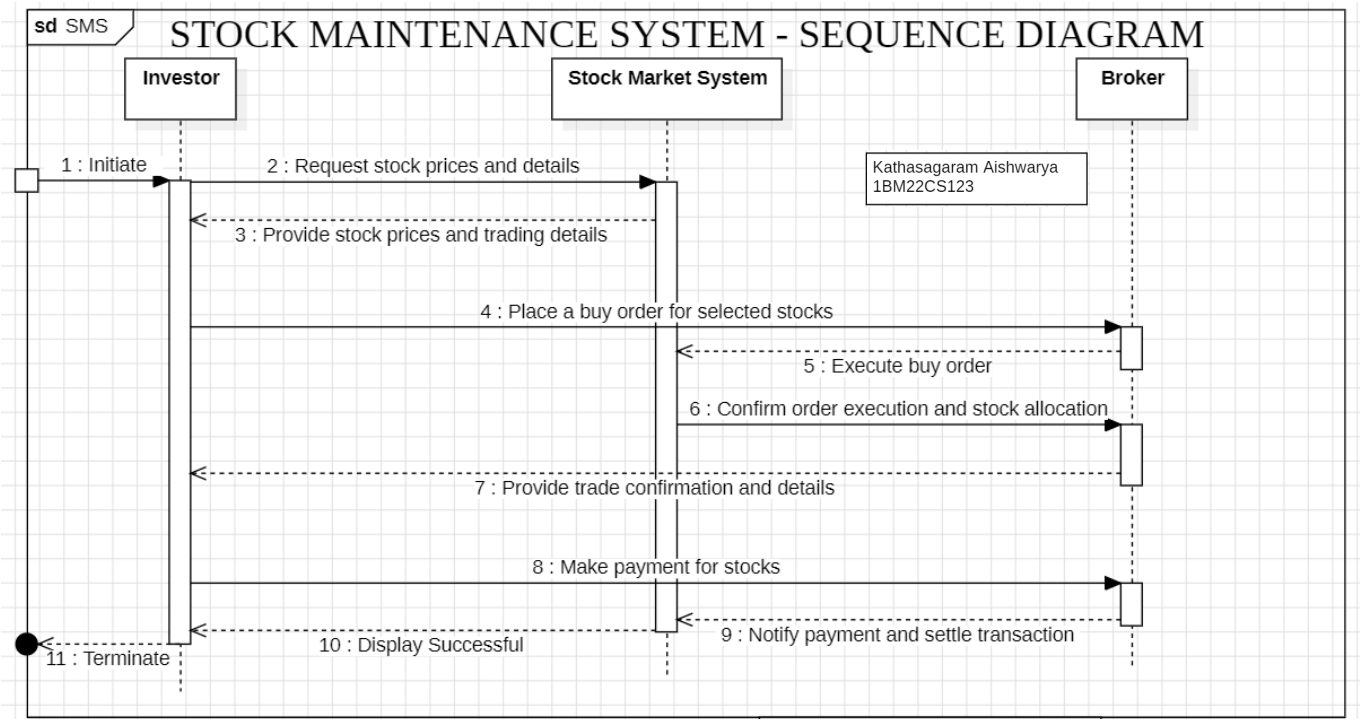
This diagram depicts the workflow of a stock management system. It begins with the "Initialization" state, where the database is loaded and the system displays a login form. The "User Verification" state is a composite state, encompassing the activities of user login and new account creation. If the user exists, the system proceeds to the "Product and Supplier Addition" state. 35 The "Product and Supplier Addition" state is another composite state, allowing for the addition of new products and suppliers. The addition of a new product triggers the "Categorizing" activity to determine the product's category. If any product has a low stock level, the system enters the "Inventory Updation" state to restock the inventory. The "Order Processing" state is also a composite state, encompassing the activities of inputting an order, placing the order, checking the stock level, and processing the payment. The "Checking Stock Level" activity uses a fork and join mechanism, splitting the flow to check the stock level for each product in the order and then rejoining to proceed with payment if stock is sufficient. Overall, this Activity Diagram provides a comprehensive view of the stock management system's workflow, utilizing composite states and fork/join constructs to model complex processes and parallel activities. It effectively visualizes the system's behavior and the

USECASE DIAGRAM



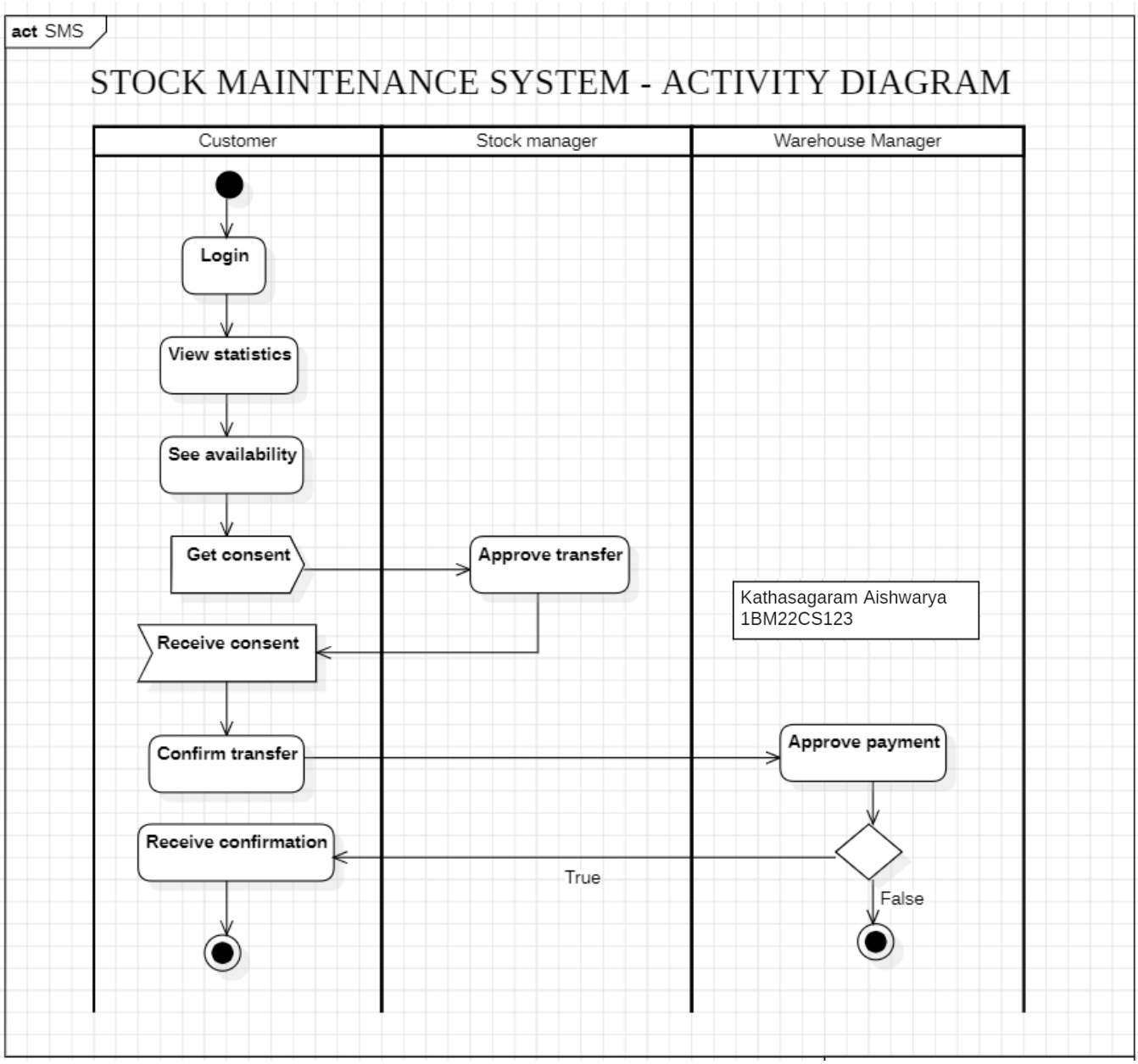
This use case diagram illustrates the various functionalities of a Stock Maintenance System. It outlines the interactions between different actors like Customer, Supplier, and Administrator, and the system itself. The system provides features such as browsing catalogs, placing orders, managing stock, and checking payment status.

SEQUENCE DIAGRAM



This sequence diagram illustrates the steps involved in a stock purchase transaction. An Investor requests stock information, places an order, and makes payment. The Stock Market System processes the order, executes it, and notifies the Investor. A Broker may facilitate the interaction between the Investor and the Stock Market System.

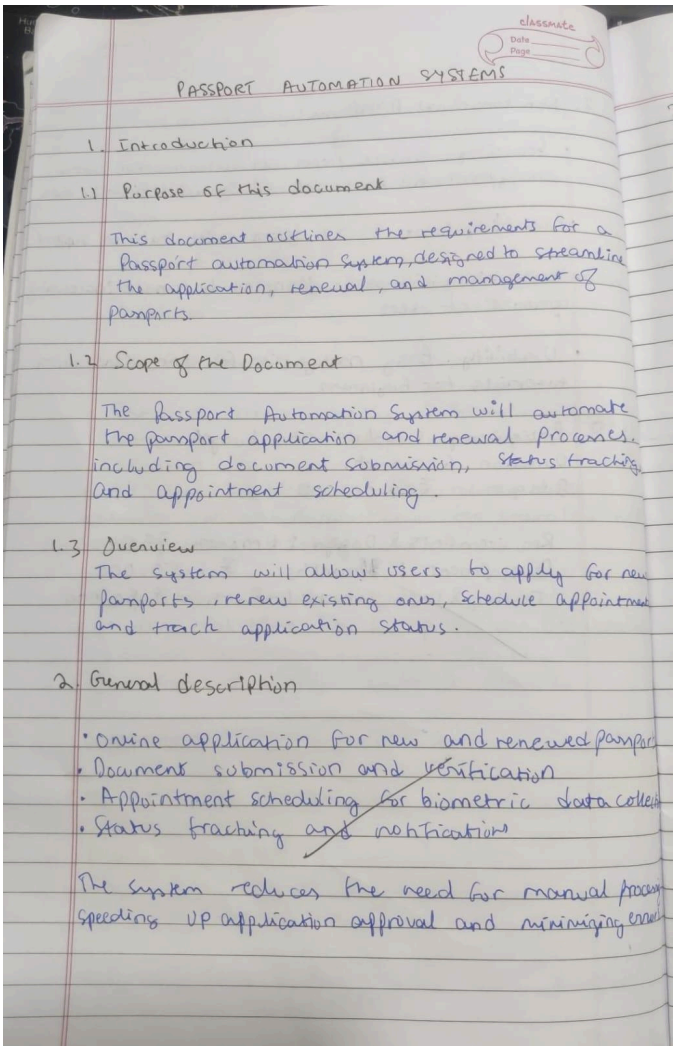
ACTIVITY DIAGRAM



This activity diagram depicts the process flow for transferring stock in a Stock Maintenance System. A customer initiates the process by logging in and viewing stock availability. The customer then requests consent from the stock manager, which is approved or denied. If approved, the warehouse manager confirms the transfer and approves payment. Finally, the customer receives a confirmation.

PASSPORT AUTOMATION SYSTEM

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)



PASSPORT AUTOMATION SYSTEMS

1. Introduction

1.1 Purpose of this document

This document outlines the requirements for a Passport automation System, designed to streamline the application, renewal, and management of passports.

1.2 Scope of the Document

The Passport Automation System will automate the passport application and renewal processes, including document submission, status tracking and appointment scheduling.

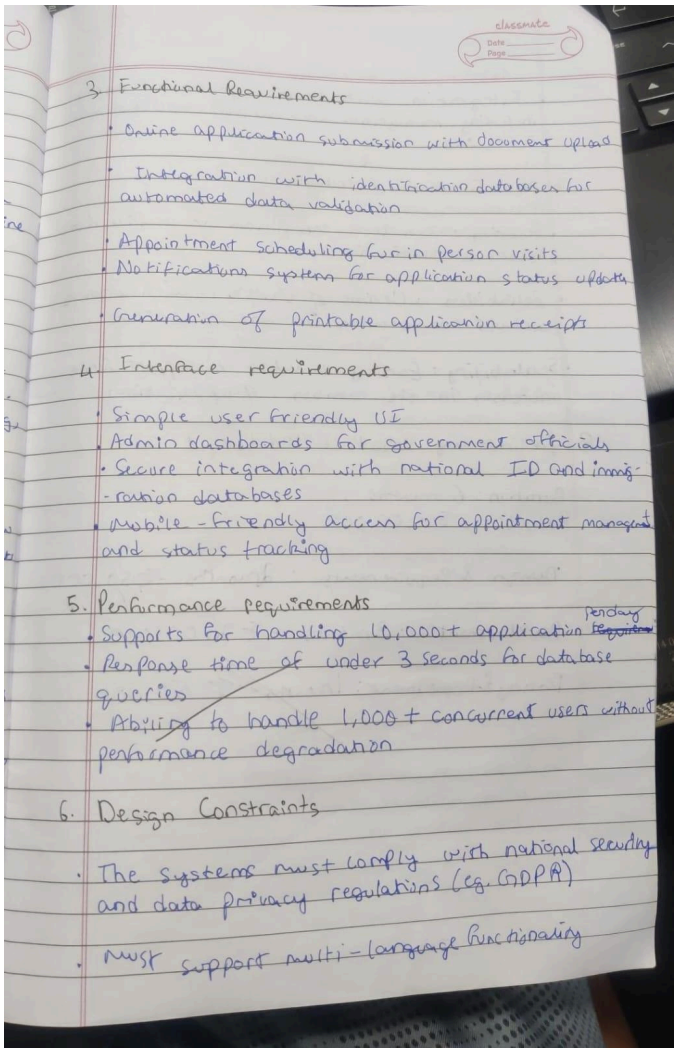
1.3 Overview

The system will allow users to apply for new passports, renew existing ones, schedule appointments and track application status.

2. General description

- Online application for new and renewed passports
- Document submission and verification
- Appointment scheduling for biometric data collection
- Status tracking and notifications

The system reduces the need for manual processing speeding up application approval and minimizing errors



3. Functional Requirements

- Online application submission with document upload
- Integration with identification databases for automated data validation
- Appointment scheduling for in person visits
- Notifications system for application status updates
- Generation of printable application receipts

4. Interface requirements

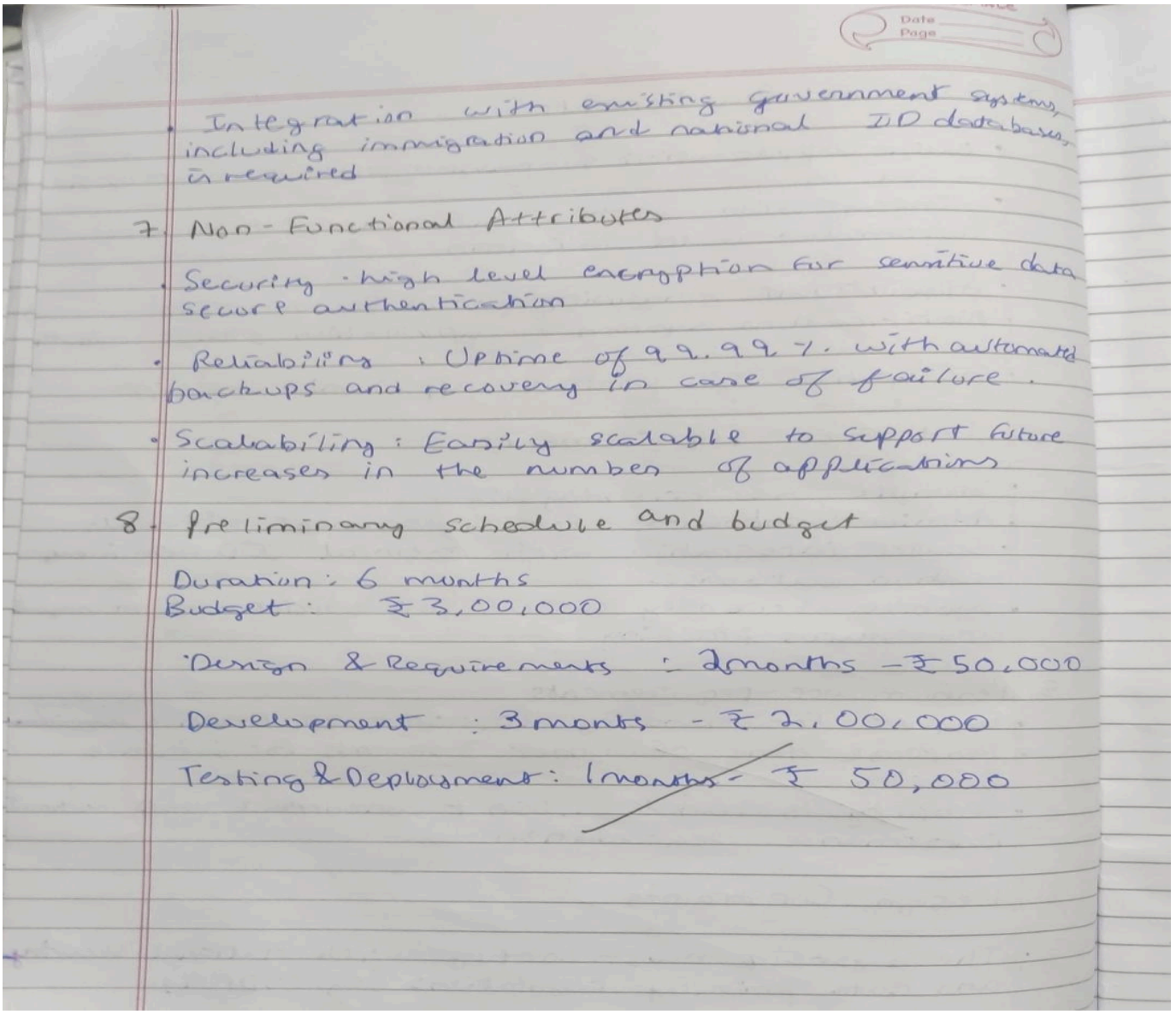
- Simple user friendly UI
- Admin dashboards for government officials
- Secure integration with national ID and immigration databases
- Mobile-friendly access for appointment management and status tracking

5. Performance requirements

- Supports for handling 10,000+ application per day
- Response time of under 3 seconds for database queries
- Ability to handle 1,000+ concurrent users without performance degradation

6. Design Constraints

- The systems must comply with national security and data privacy regulations (eg. GDPR)
- must support multi-language functionality



- Integration with existing government systems, including immigration and national ID databases, is required

7. Non-Functional Attributes

- Security : high level encryption for sensitive data secure authentication
- Reliability : Uptime of 99.99% with automated backups and recovery in case of failure.
- Scalability : Easily scalable to support future increases in the numbers of applications

8. Preliminary schedule and budget

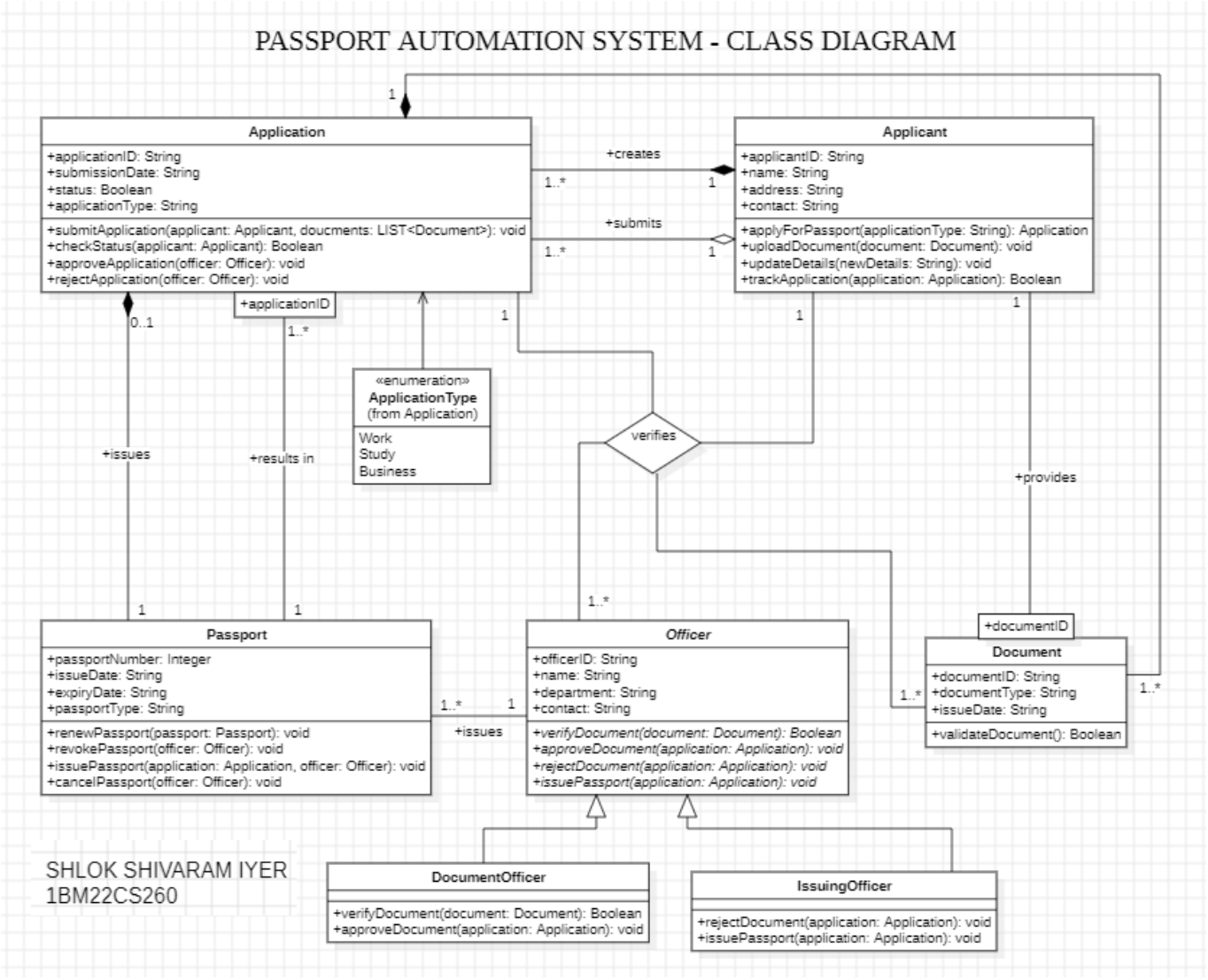
Duration : 6 months
Budget : ₹3,00,000

Design & Requirements : 2months - ₹50,000

Development : 3months - ₹2,00,000

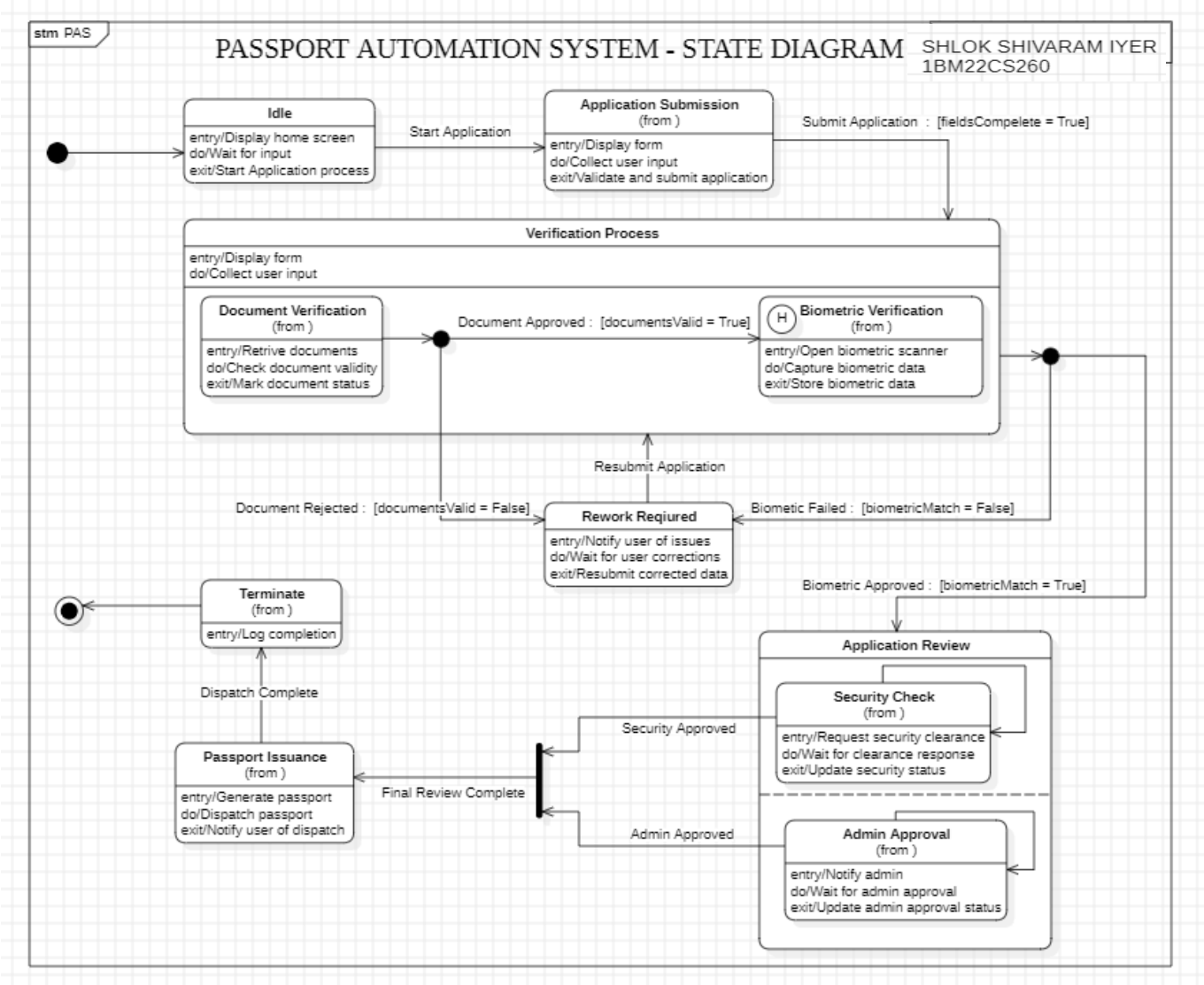
Testing & Deployment : 1months - ₹50,000

CLASS DIAGRAM



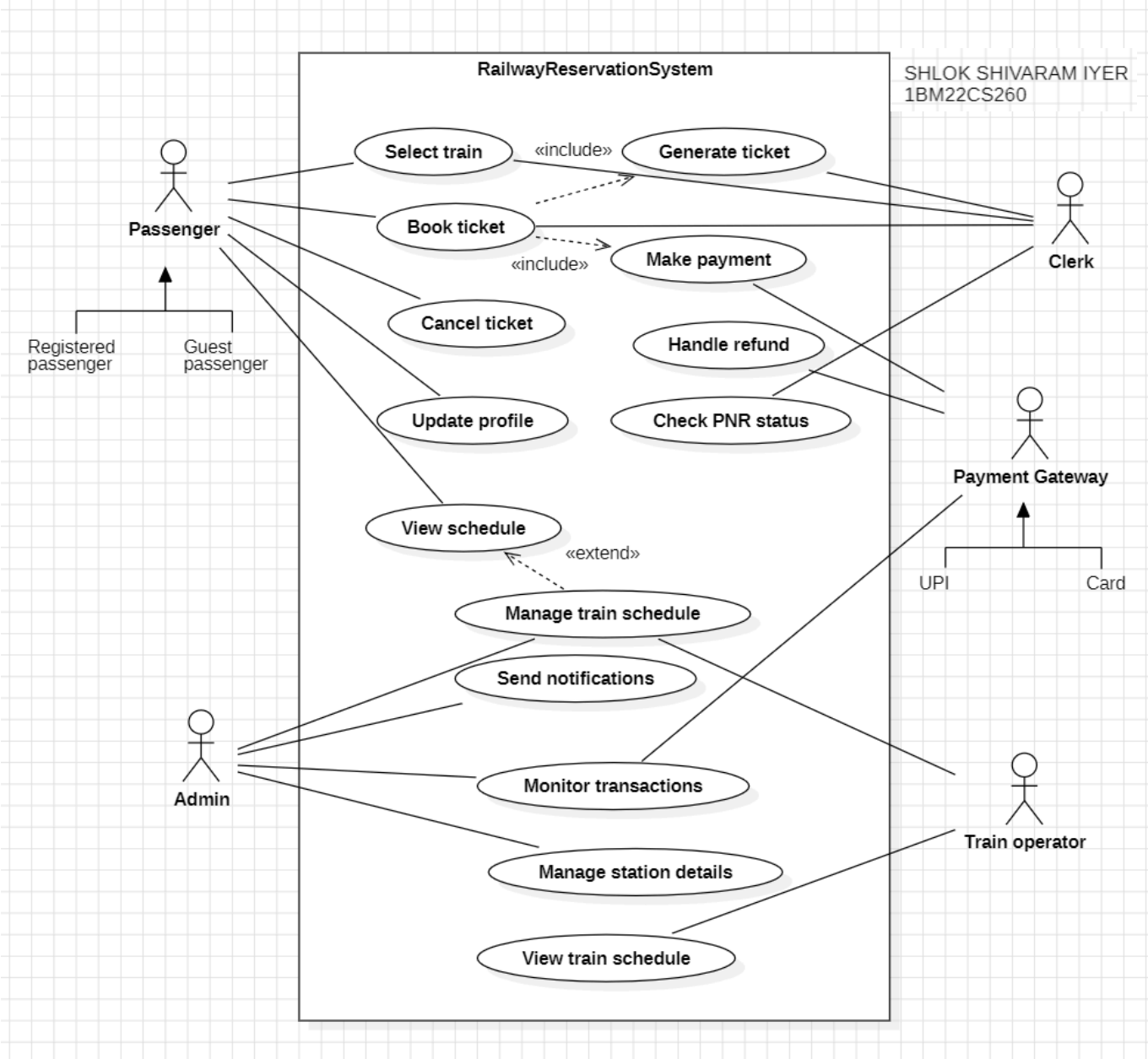
This class diagram models the entities and their relationships in a passport automation system. It includes classes for Applicant, Application, Passport, Officer, Document, and DocumentOfficer. The diagram shows how these classes interact, such as an Applicant submitting an Application, an Officer verifying documents, and a DocumentOfficer issuing a Passport.

STATE DIAGRAM



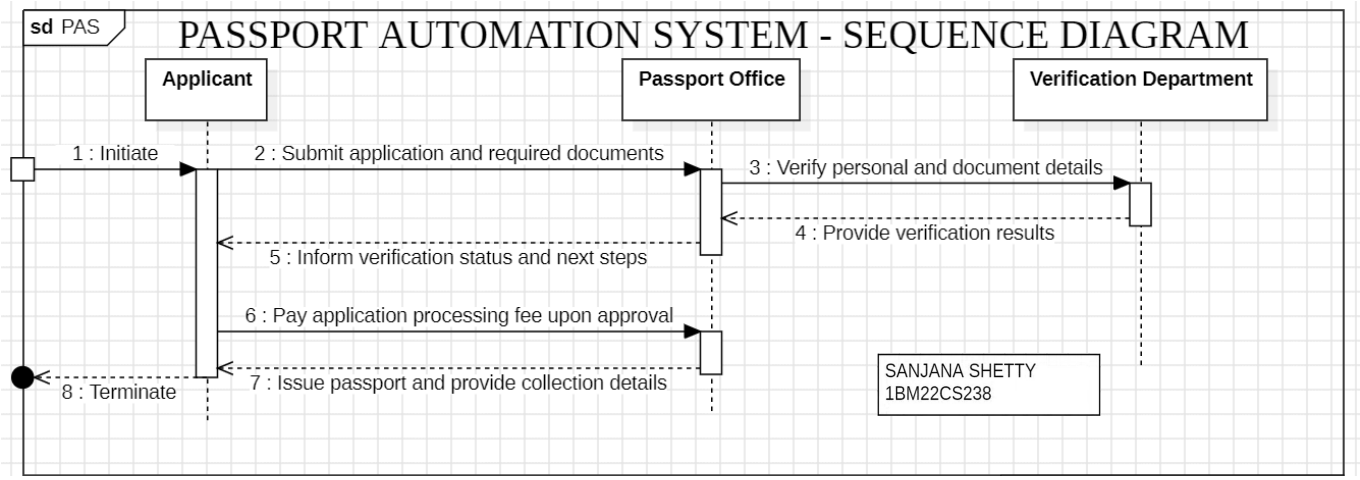
This state diagram illustrates the workflow of a passport application. It starts with an idle state and transitions through application submission, document verification, biometric verification, application review, and finally, passport issuance. The diagram also includes rejection and rework paths for incomplete or invalid applications

USE CASE DIAGRAM



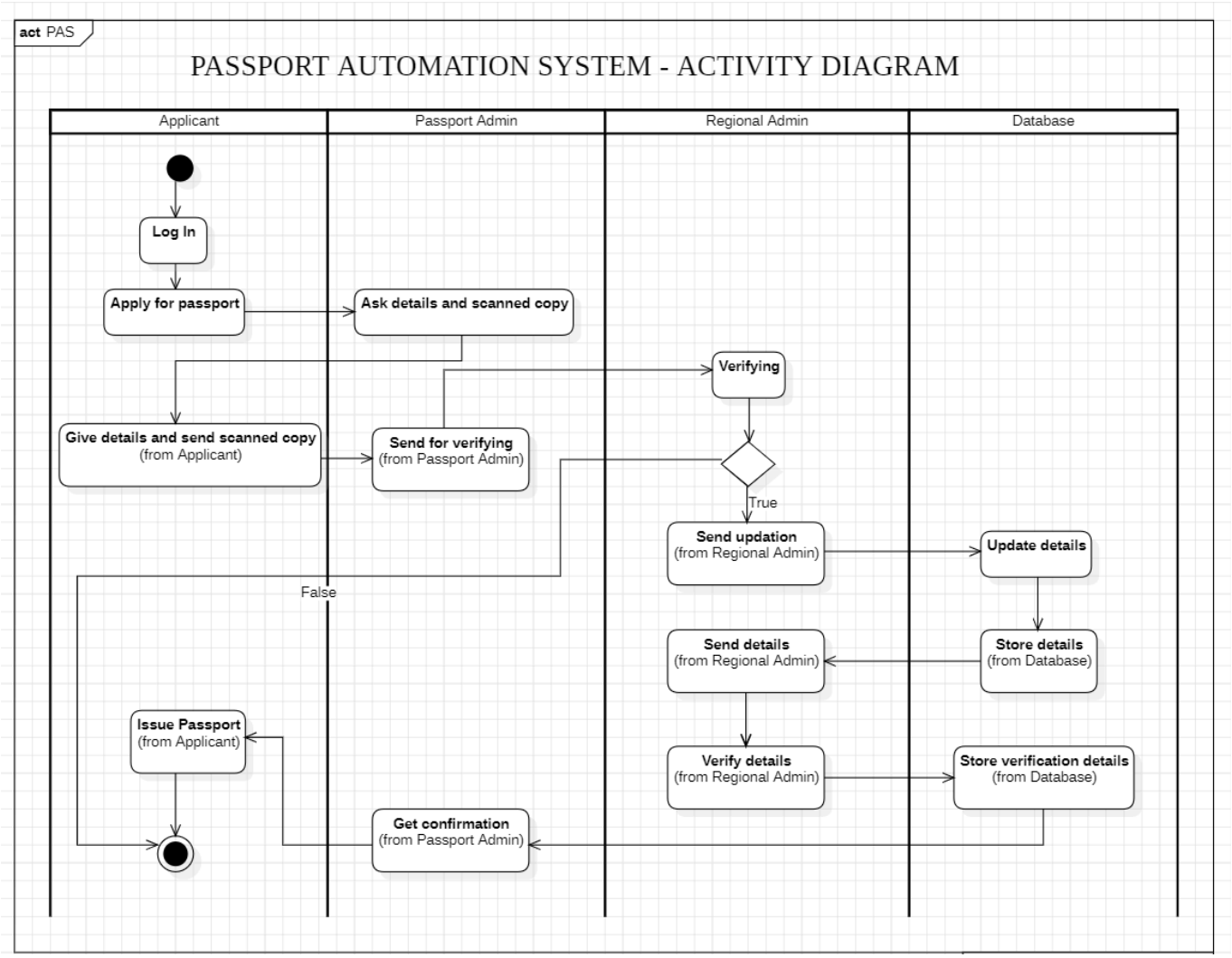
This use case diagram illustrates the various functionalities of a Railway Reservation System. It outlines the interactions between different actors like Passenger, Clerk, Admin, Train Operator, and Payment Gateway. The system provides features such as booking tickets, viewing schedules, managing train schedules, and handling refunds

SEQUENCE DIAGRAM



This sequence diagram depicts the passport application process. The Applicant submits an application and documents to the Passport Office. The Verification Department verifies the details and informs the Passport Office. The Passport Office notifies the Applicant of the status, collects the fee (if approved), and issues the passport

ACTIVITY DIAGRAM



This diagram depicts the passport application process. The Applicant submits details and documents to the Passport Admin. The Passport Admin forwards it to the Regional Admin for verification. Upon verification, the database is updated, and the Passport Admin informs the Applicant about the status. If approved, the passport is issued