VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

OBJECT ORIENTED MODELING

Submitted by

Sanghamitra R
(1BM22CS237)

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 **Sanjana Shetty (1BM22CS238),** 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.

M Lakshmi Neelima
Assistant Professor
Department of CSE
BMSCE, Bengaluru

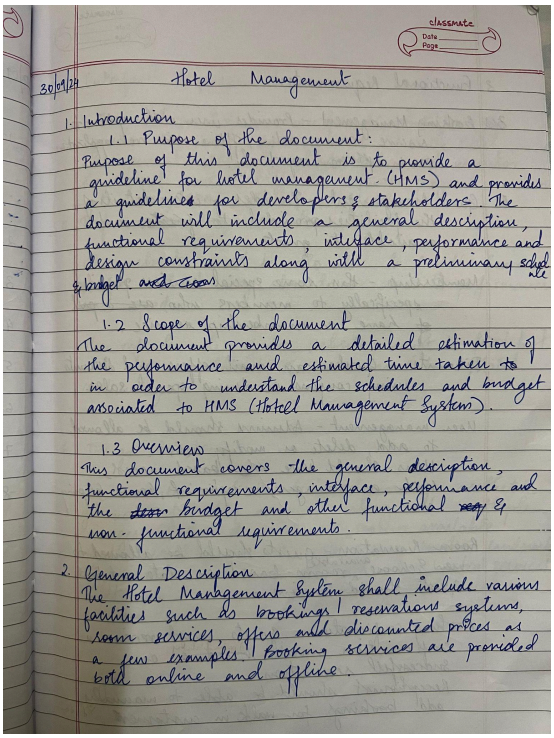
Dr. Kavitha Sooda
Professor and Head
Department of CSE
BMSCE, Bengaluru

Table of Content

https://github.com/sanghamitrarajagopal/oomd_1BM22CS237

Hotel Management System

Software Requirements Specification (SRS)



30/09/24 Hotel Management

1. Introduction

1.1 Purpose of the document:

Purpose of this document is to provide a guideline for hotel management (HMS) and provides a guidelines for developers & stakeholders. The document will include a general description, functional requirements, interface, performance and design constraints along with a preliminary schedule & budget ~~and costs~~

1.2 Scope of the document

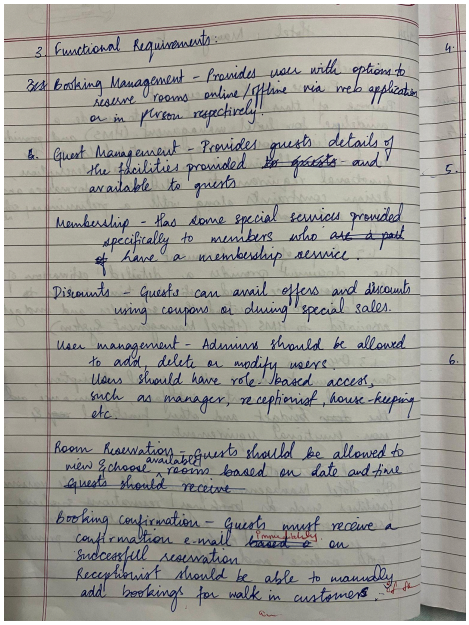
The document provides a detailed estimation of the performance and estimated time taken ~~to~~ in order to understand the schedules and budget associated to HMS (Hotel Management System).

1.3 Overview

This document covers the general description, functional requirements, interface, performance and the ~~desc~~ budget and other functional ~~req~~ & non-functional requirements.

2. General Description

The Hotel Management System shall include various facilities such as bookings/reservations systems, room services, offers and discounted prices as a few examples. Booking services are provided both online and offline.



3. Functional Requirements:

3.1 Booking Management - Provides user with options to reserve rooms online/offline via web application or in person respectively.

4.

5.

6. Guest Management - Provides guests details of the facilities provided ~~to guests~~ and available to guests

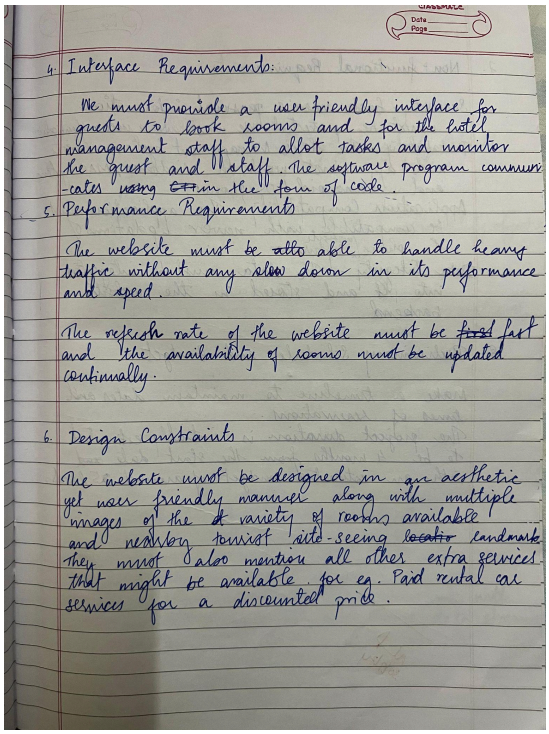
Membership - Has some special services provided specifically to members who ~~are a part~~ of have a membership service.

Discounts - Guests can avail offers and discounts using coupons or during special sales.

User management - Admins should be allowed to add, delete or modify users. Users should have role-based access, such as manager, receptionist, house-keeping etc.

Room Reservation - Guests should be allowed to view & choose available rooms based on date and time. Guests should receive

Booking Confirmation - Guests must receive a confirmation e-mail immediately ~~based~~ on successful reservation. Receptionist should be able to manually add bookings for walk in customers.



4. Interface Requirements:

We must provide a user friendly interface for guests to book rooms and for the hotel management staff to allot tasks and monitor the guest and staff. The software program communicates ~~using~~ ~~C++~~ in the form of code.

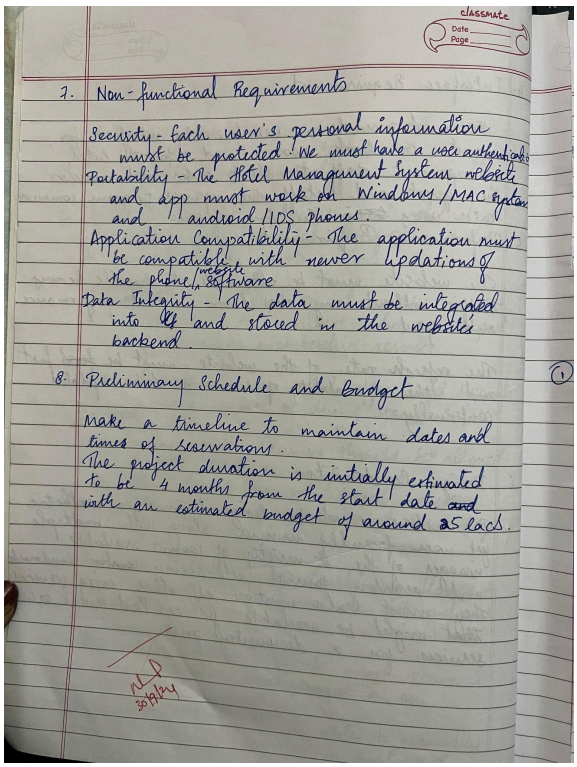
5. Performance Requirements

The website must be ~~atto~~ able to handle heavy traffic without any ~~slow~~ down in its performance and speed.

The refresh rate of the website must be ~~first~~ fast and the availability of rooms must be updated continually.

6. Design Constraints

The website must be designed in an aesthetic yet user friendly manner along with multiple images of the ~~a~~ variety of rooms available and nearby tourist site-seeing ~~locatio~~ landmarks. They must also mention all other extra services that might be available for eg. Paid rental car services for a discounted price.



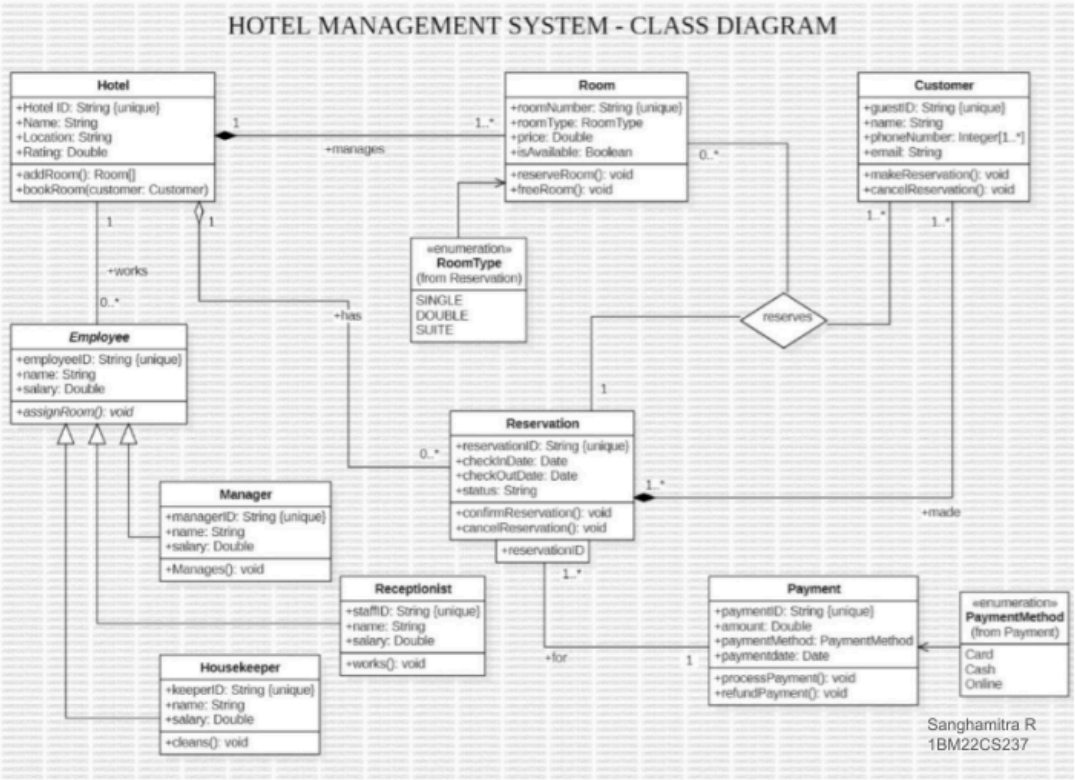
7. Non-functional Requirements

Security - Each user's personal information must be protected. We must have a user authentication.
Portability - The Hotel Management System website and app must work on Windows/MAC system and android/IOS phones.
Application Compatibility - The application must be compatible with newer updations of the phone/website software.
Data Integrity - The data must be integrated into ~~the~~ and stored in the website's backend.

8. Preliminary Schedule and Budget

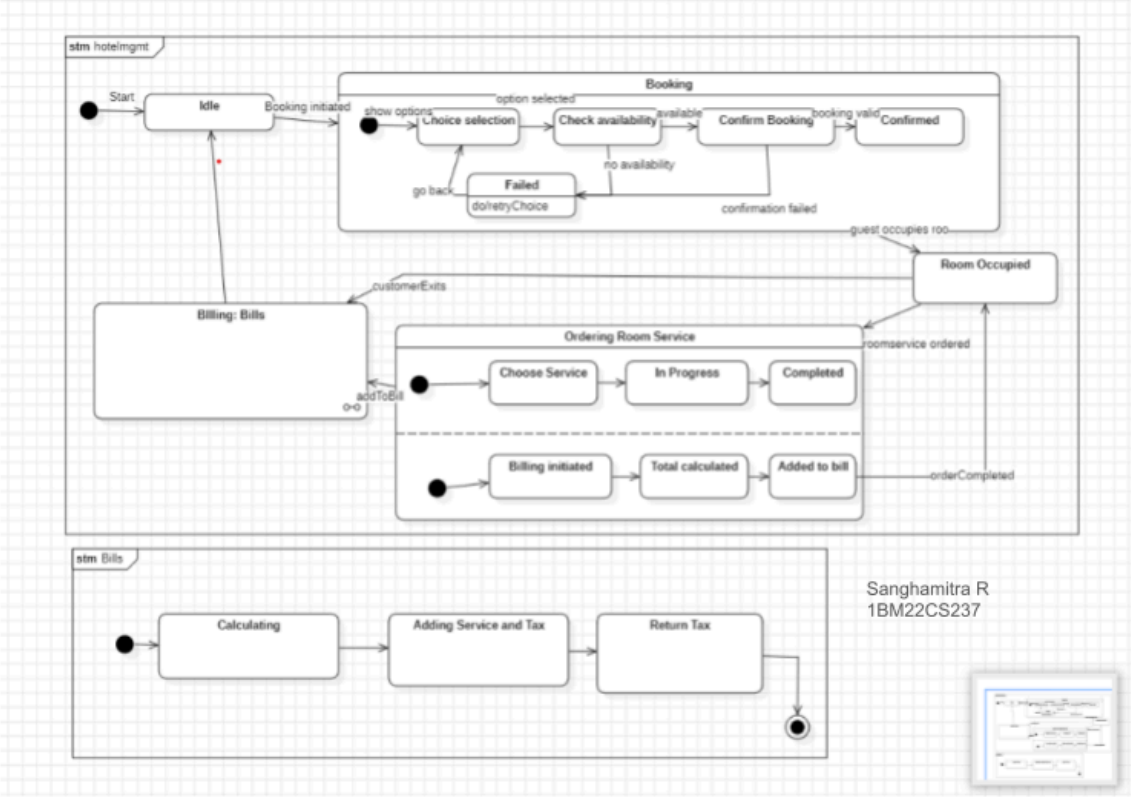
Make a timeline to maintain dates and times of reservations.
The project duration is initially estimated to be 4 months from the start date ~~and~~ with an estimated budget of around 25 lacs.

CLASS DIAGRAM



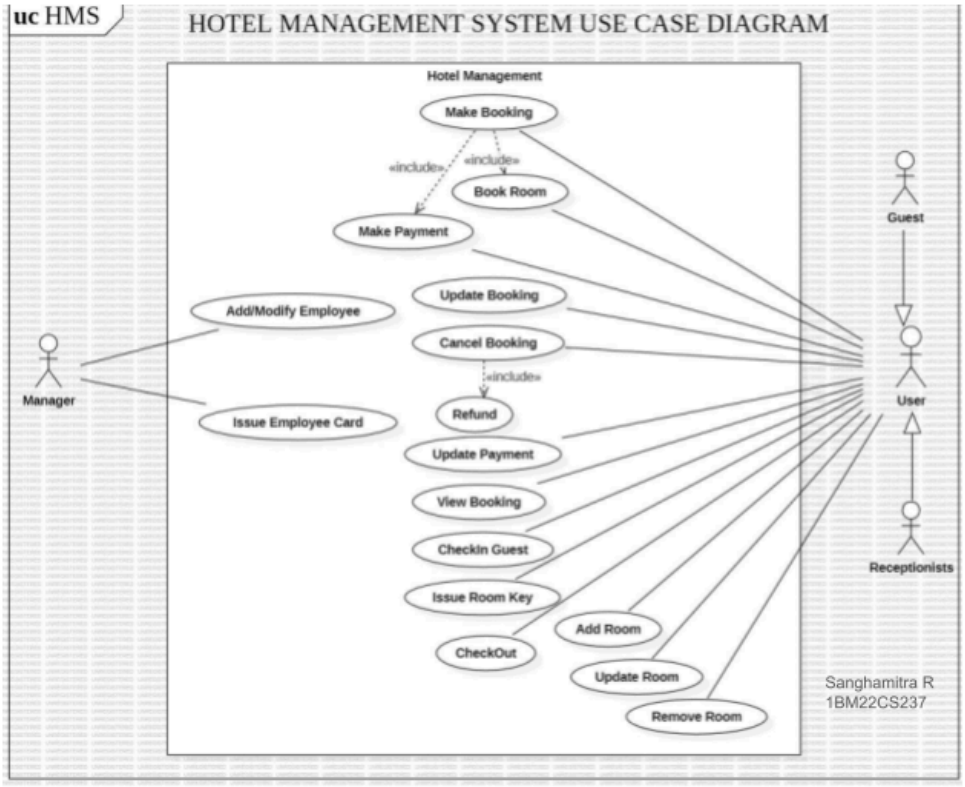
The class diagram illustrates a Hotel Management System comprising various entities and their relationships. The central class, Hotel, manages multiple Rooms, each identified by attributes like roomNumber, roomType, price, and availability. Customers can reserve rooms through the Reservation class, which 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



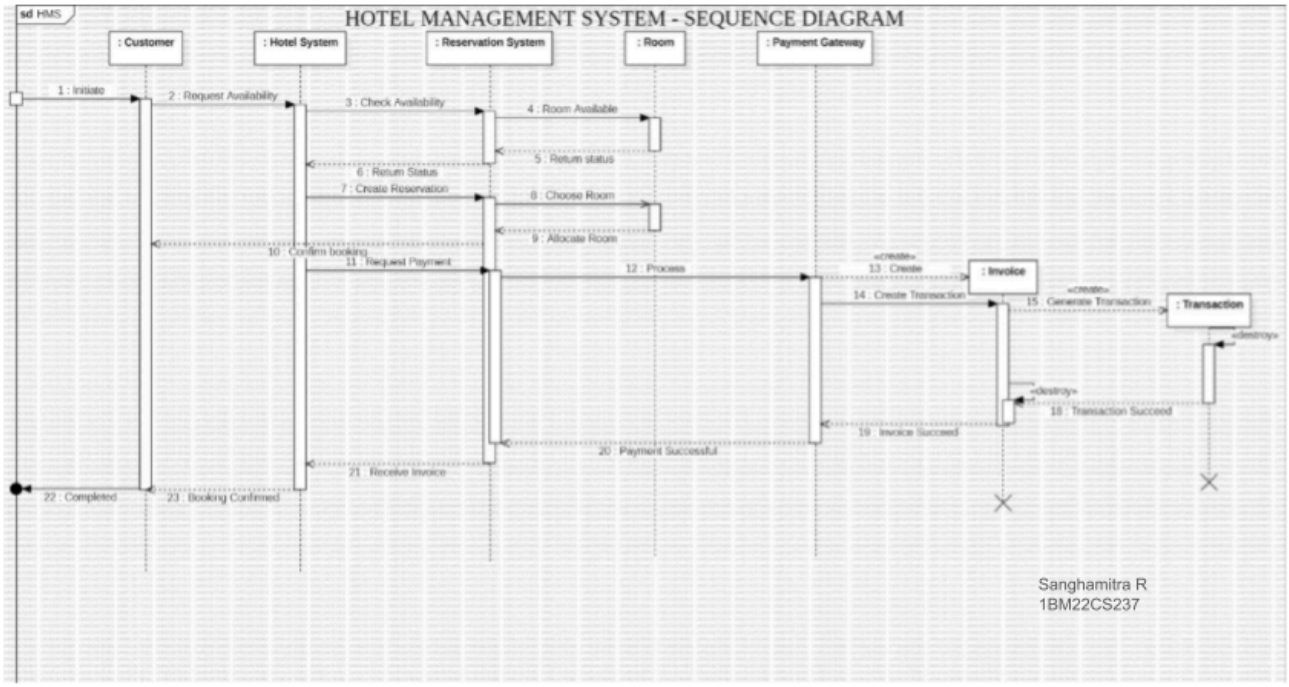
The state diagram depicts the Hotel Management System processes, starting with the Reservation Process (from Reservation Pending to payment completion), followed by the Check-in Process (Ready for Check-In to Checked-In). During the stay, the In-Service Process manages tasks like Housekeeping, Food Ordering, 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



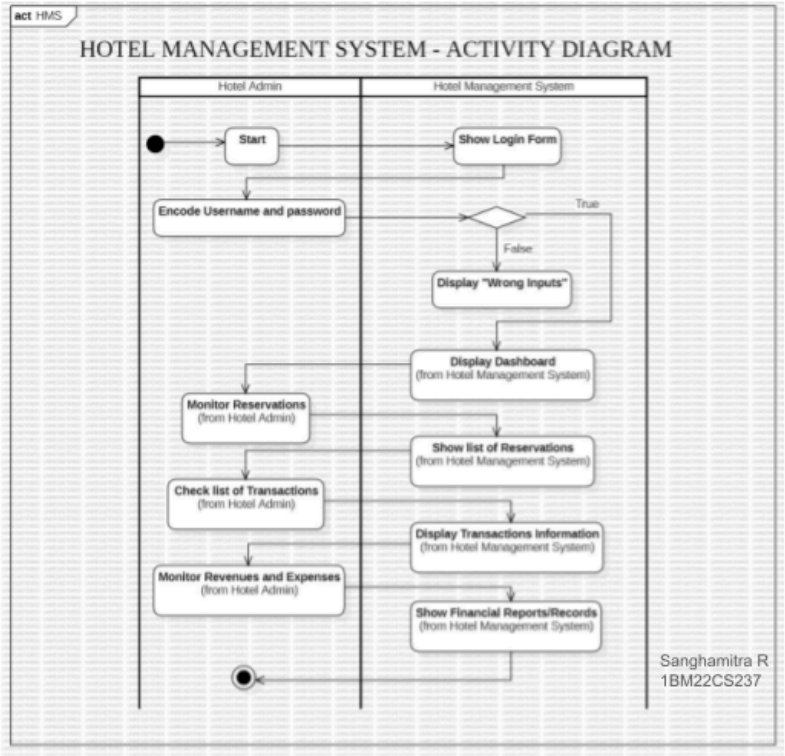
The use case diagram depicts the Hotel Management System, highlighting interactions between Customer, Manager, Receptionist, and Restaurant Staff. Key actions include Authentication, Room Reservation, Payment, Check-in, Room Service, and Checkout. It shows relationships like include and extend to represent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

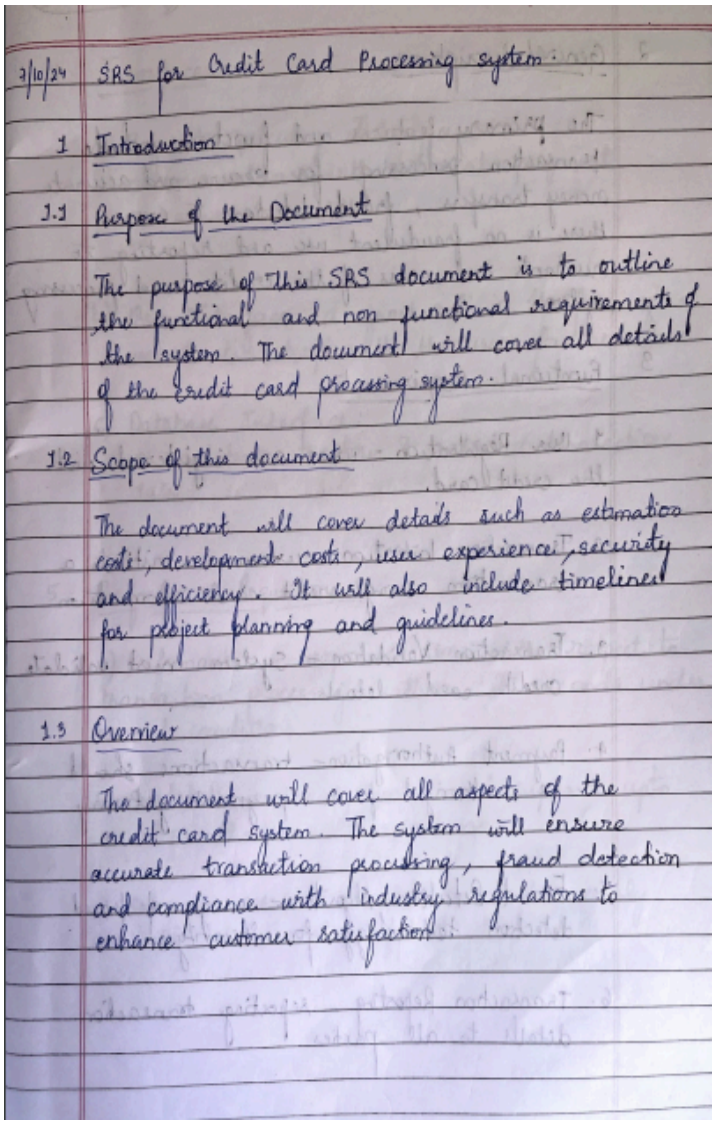
ACTIVITY DIAGRAM



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)



3/10/24 SRS for Credit Card Processing system.

1 Introduction

1.1 Purpose of the Document

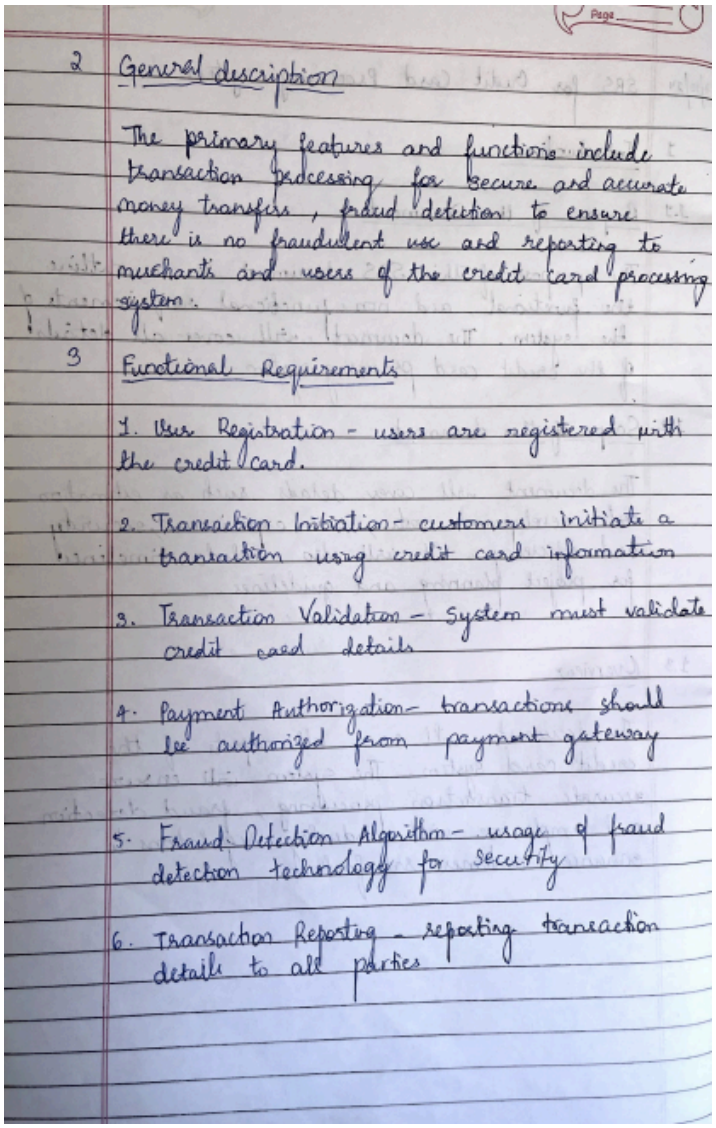
The purpose of this SRS document is to outline the functional and non functional requirements of the system. The document will cover all details of the credit card processing system.

1.2 Scope of this document

The document will cover details such as estimation costs, development costs, user experience, security and efficiency. It will also include timelines for project planning and guidelines.

1.3 Overview

The document will cover all aspects of the credit card system. The system will ensure accurate transaction processing, fraud detection and compliance with industry regulations to enhance customer satisfaction.

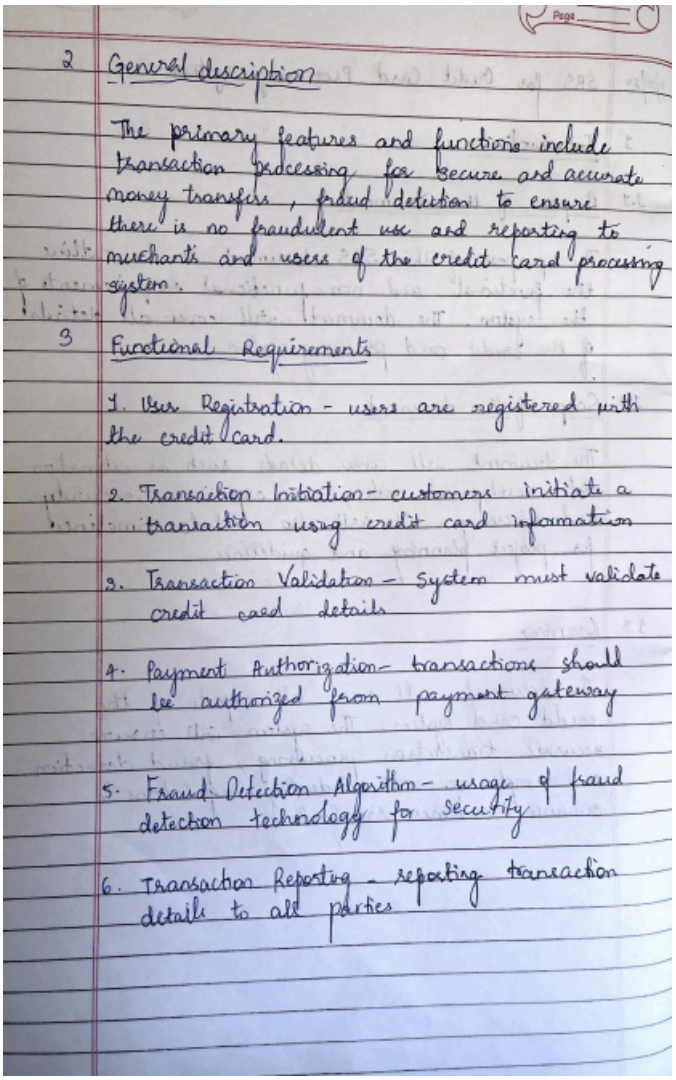


2 General description

The primary features and functions include transaction processing for secure and accurate money transfers, fraud detection to ensure there is no fraudulent use and reporting to merchants and users of the credit card processing system.

3 Functional Requirements

1. User Registration - users are registered with the credit card.
2. Transaction Initiation - customers initiate a transaction using credit card information
3. Transaction Validation - System must validate credit card details
4. Payment Authorization - transactions should be authorized from payment gateway
5. Fraud Detection Algorithm - usage of fraud detection technology for security
6. Transaction Reporting - reporting transaction details to all parties

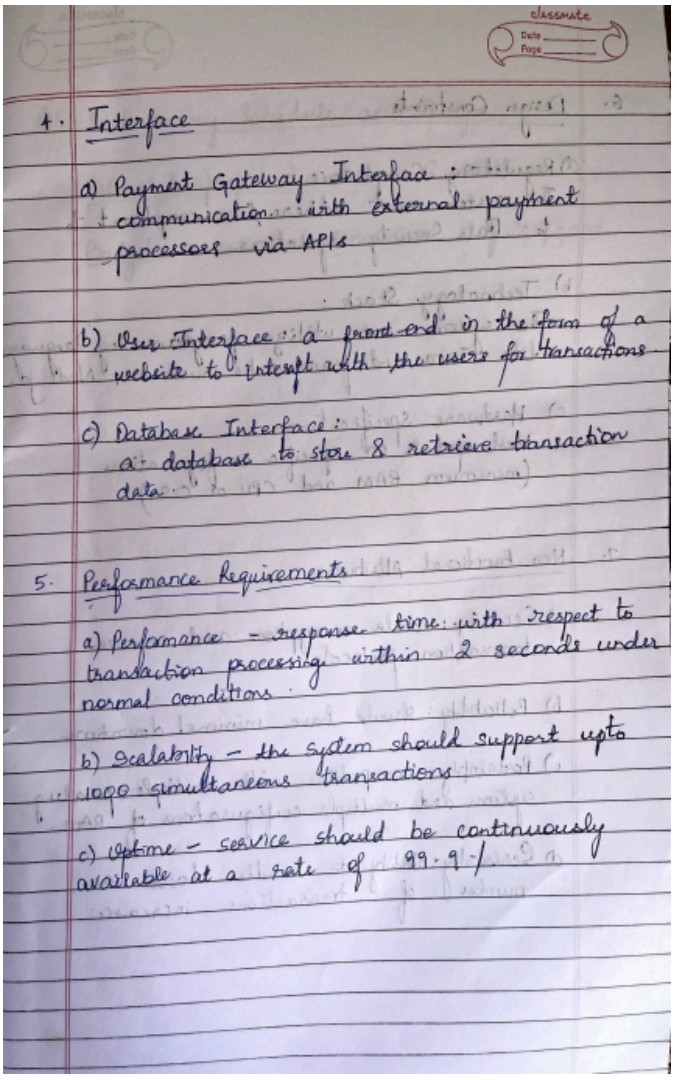


2 General description

The primary features and functions include transaction processing for secure and accurate money transfers, fraud detection to ensure there is no fraudulent use and reporting to merchants and users of the credit card processing system.

3 Functional Requirements

1. User Registration - users are registered with the credit card.
2. Transaction Initiation - customers initiate a transaction using credit card information
3. Transaction Validation - System must validate credit card details
4. Payment Authorization - transactions should be authorized from payment gateway
5. Fraud Detection Algorithm - usage of fraud detection technology for security
6. Transaction Reposting - reporting transaction details to all parties



4. Interface

a) Payment Gateway Interface : communication with external payment processors via APIs

b) User Interface : a front-end in the form of a website to interact with the users for transactions

c) Database Interface : a database to store & retrieve transaction data

5. Performance Requirements

a) Performance - response time with respect to transaction processing within 2 seconds under normal conditions

b) Scalability - the system should support upto 1000 simultaneous transactions

c) Uptime - service should be continuously available at a rate of 99.9%

6. Design Constraints

a) Regulatory Compliance :
The system must adhere / be compliant to to Data Security regulations

b) Technology Stack :
The system must utilize programming language like Java and frameworks such as Node.js

c) Hardware Specifications
Must operate on specific configurations (minimum RAM and CPU of OS)

7. Non-Functional Attributes

a) Security : data encryption and secure transaction protocols

b) Reliability : Should have minimal downtime

c) Portability : compatible with multiple operating systems and multiple configurations of RAM.

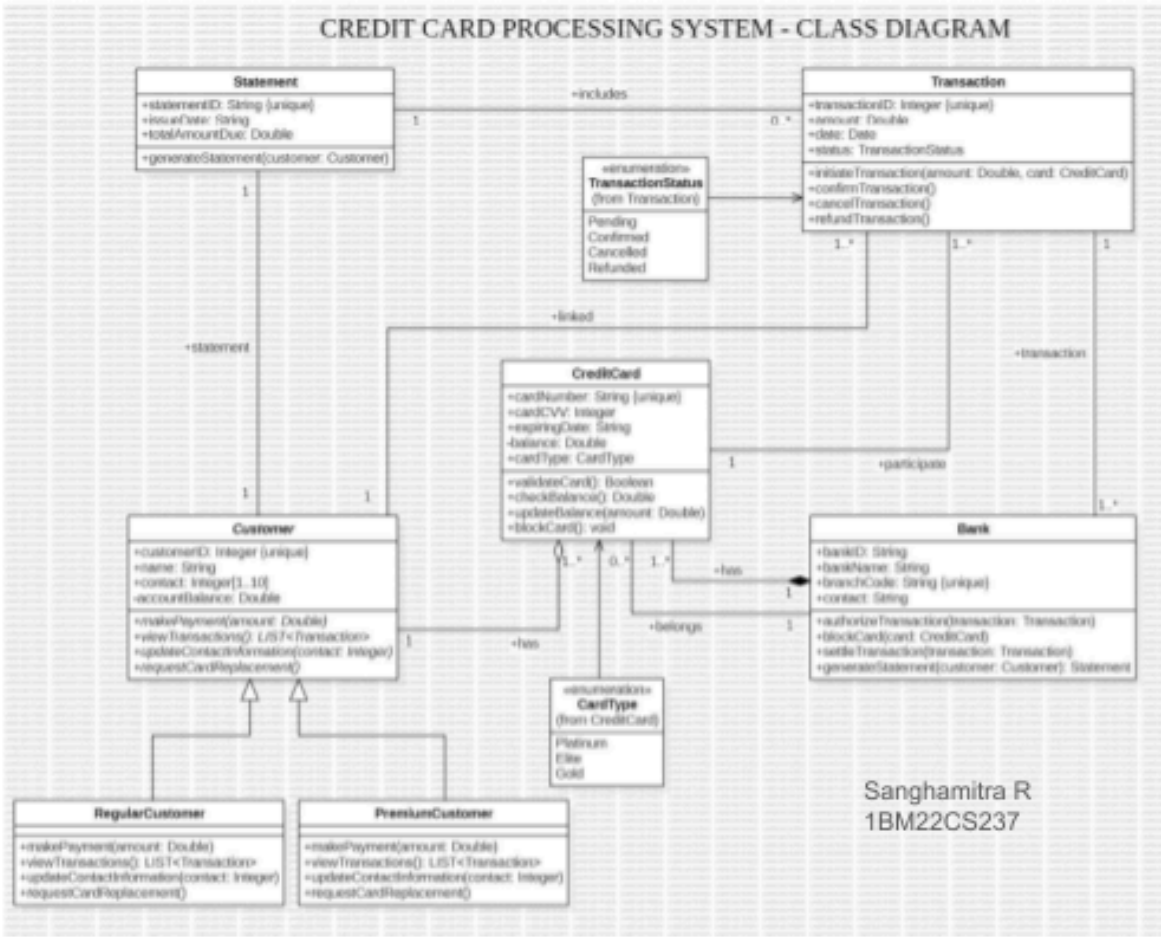
d) Scalability : ability to handle loads as number of transactions increase

8. Preliminary Schedule and Budget

Estimated project duration : 6 months

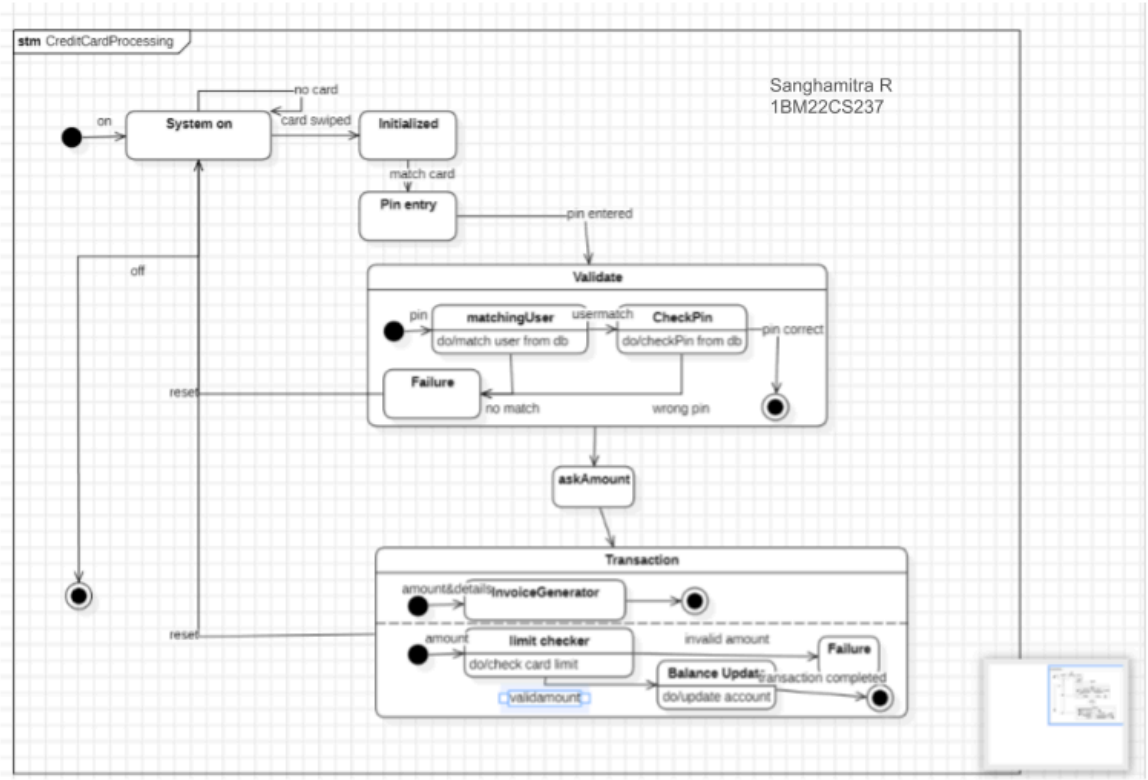
Budget : overall cost estimation is Rs. 200,000

CLASS DIAGRAM



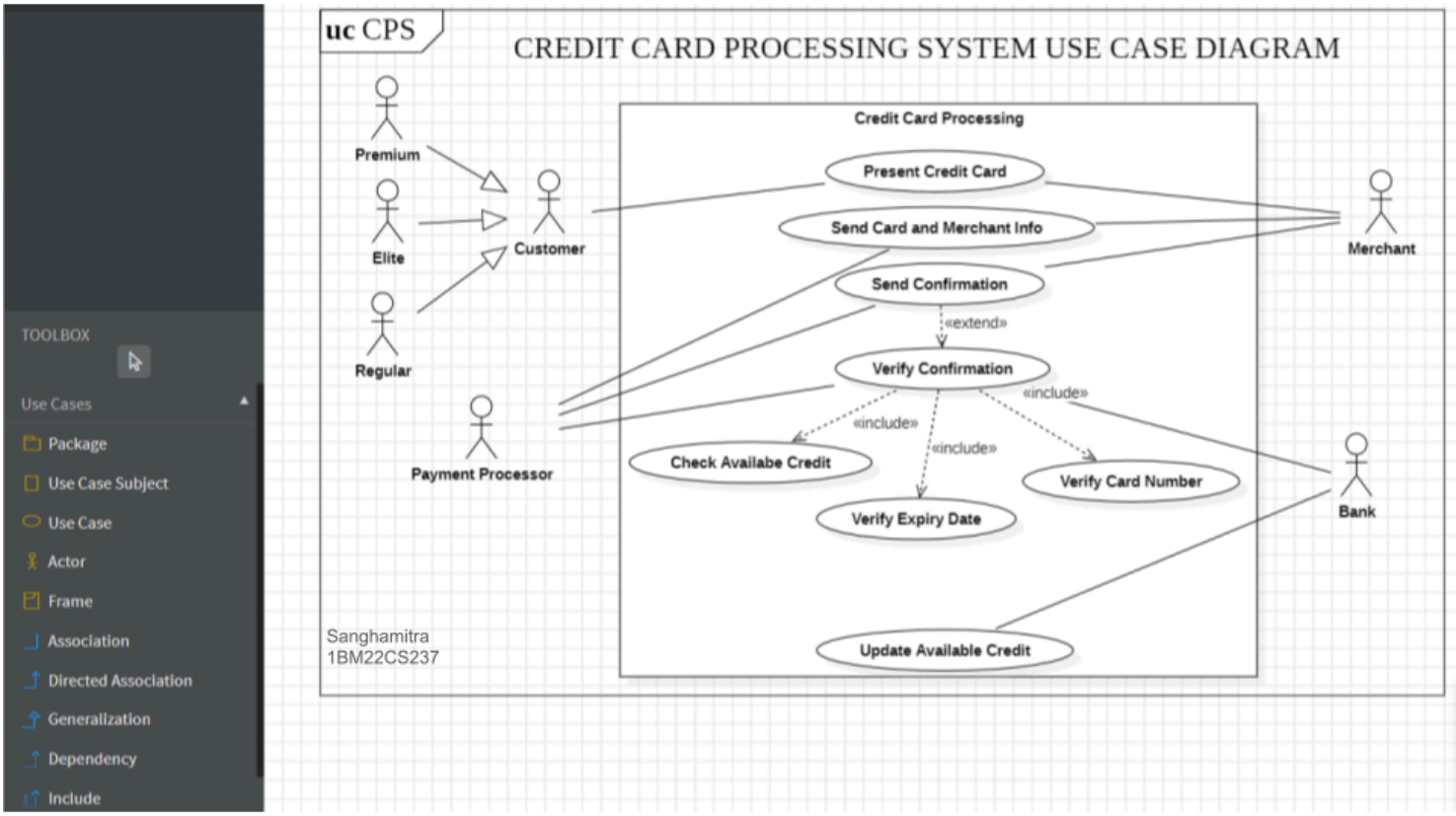
This UML class diagram illustrates the core components of a financial system. It depicts entities such as Customer, CreditCard, Transaction, Statement, and Bank, along with their attributes and relationships. Key features include transaction management, statement generation, and customer categorization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

STATE DIAGRAM



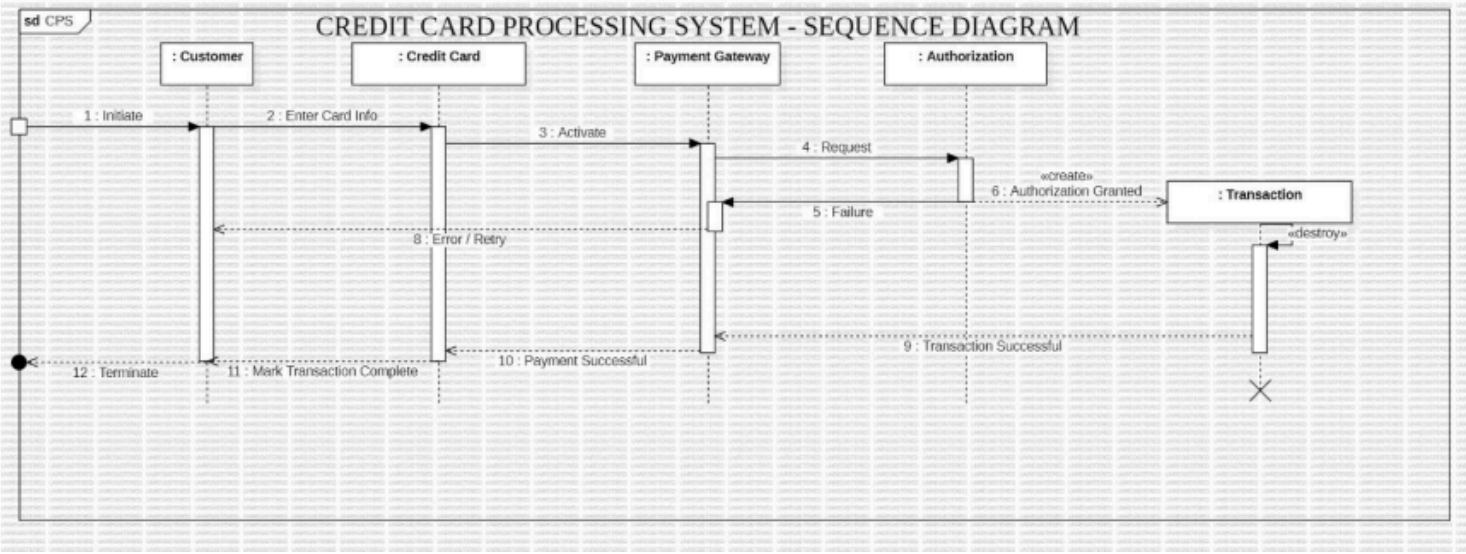
This state diagram models a credit card processing system. It shows the system transitioning through states like Idle, Transaction Initiated, Transaction Processing, and either Approved or Declined. The system processes card data, verifies transactions, and updates account balances accordingly.

USE CASE DIAGRAM



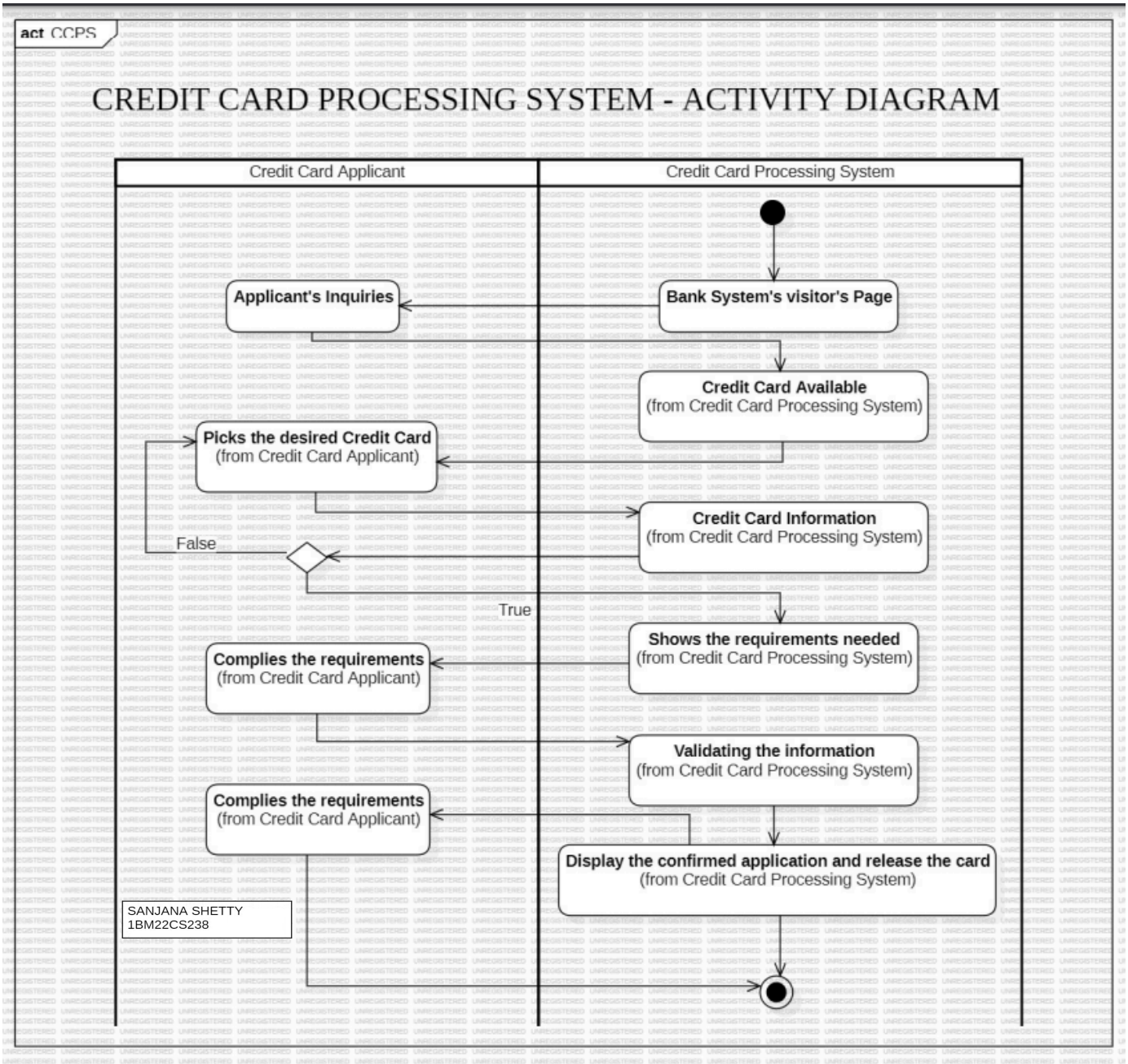
This UML use case diagram outlines the functionalities of a CreditCardProcessing system. It shows various actors like Cardholder (Personal and Business), Bank, Admin, Merchant, and Payment Gateway interacting with the system through different use cases. Key features include authorizing transactions, 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 use case 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.

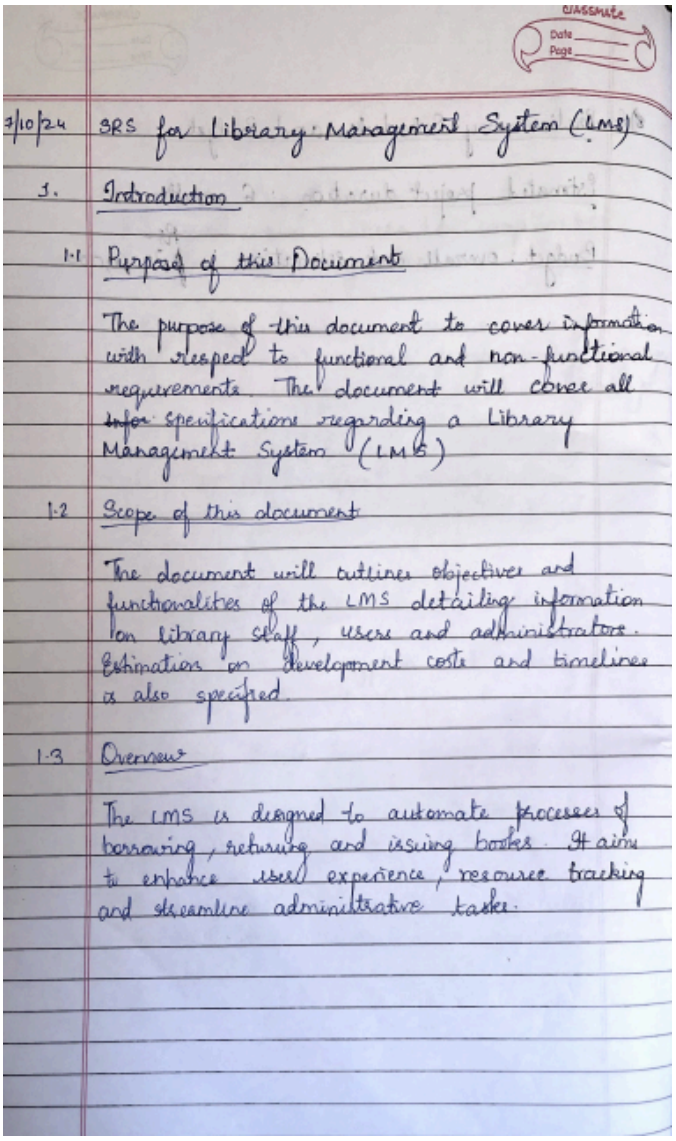
ACTIVITY DIAGRAM



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)



7/10/24 SRS for Library Management System (LMS)

1. Introduction

1.1 Purpose of this Document

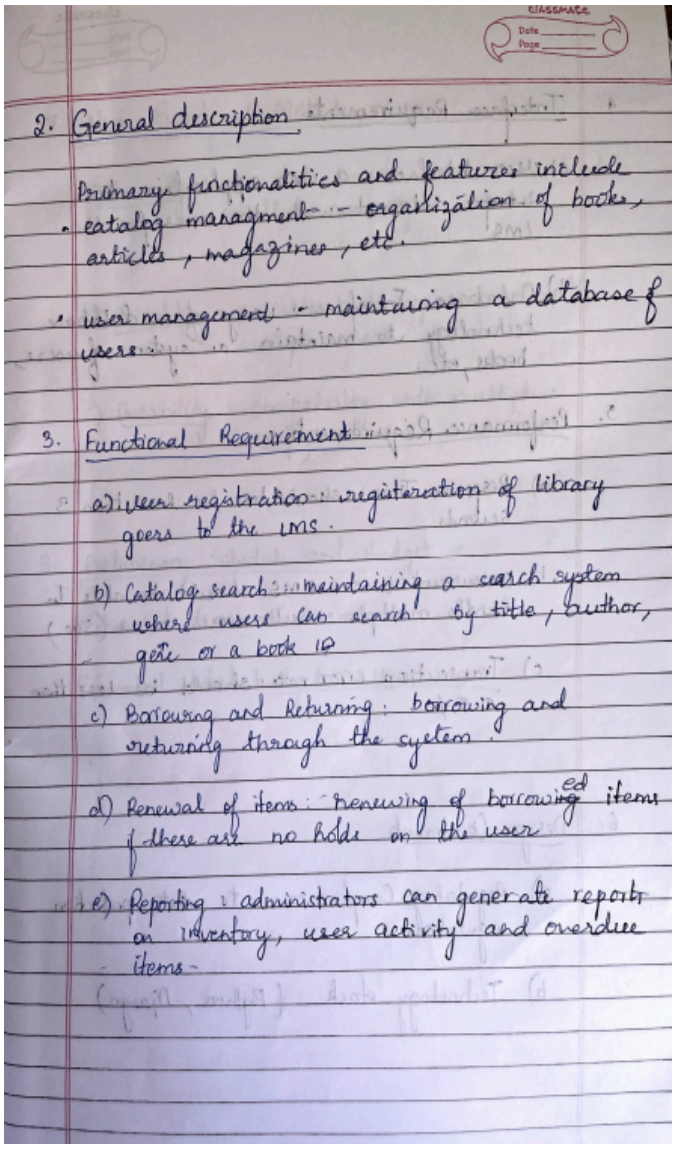
The purpose of this document to cover information with respect to functional and non-functional requirements. The document will cover all the specifications regarding a Library Management System (LMS)

1.2 Scope of this document

The document will outlines objectives and functionalities of the LMS detailing information for library staff, users and administrators. Estimations on development costs and timelines is also specified.

1.3 Overview

The LMS is designed to automate processes of borrowing, returning and issuing books. It aims to enhance user experience, resource tracking and streamline administrative tasks.



2. General description

Primary functionalities and features include
- catalog managment - organization of books, articles, magazines, etc.
- user management - maintaining a database of users.

3. Functional Requirement

a) User registration: registeration of library goers to the LMS.

b) Catalog search: maintaining a search system where users can search by title, author, genre or a book ID

c) Borrowing and Returning: borrowing and returning through the system.

d) Renewal of items: renewing of borrowed items if there are no holds on the user

e) Reporting: administrators can generate reports on inventory, user activity and overdue items.

4. Interface Requirements

a) User Interface : a web application to interact with the users for the LMS

b) Database Interface : usage of a database technology to maintain a system of users, books, etc.

5. Performance Requirements

a) Response Time : should be for less than 3 seconds

b) concurrent users : LMS should be able to handle multiple simultaneous users (500)

c) Transaction error rate : should be less than 0.5%

6. Design Constraints

a) Regulatory Compliance : to data protection regulations

b) Technology stack (Python, Django)

7. Non Functional Attributes

a) Security : user data must be secure and handled carefully.

b) Reliability : reliable web apps to handle multiple users.

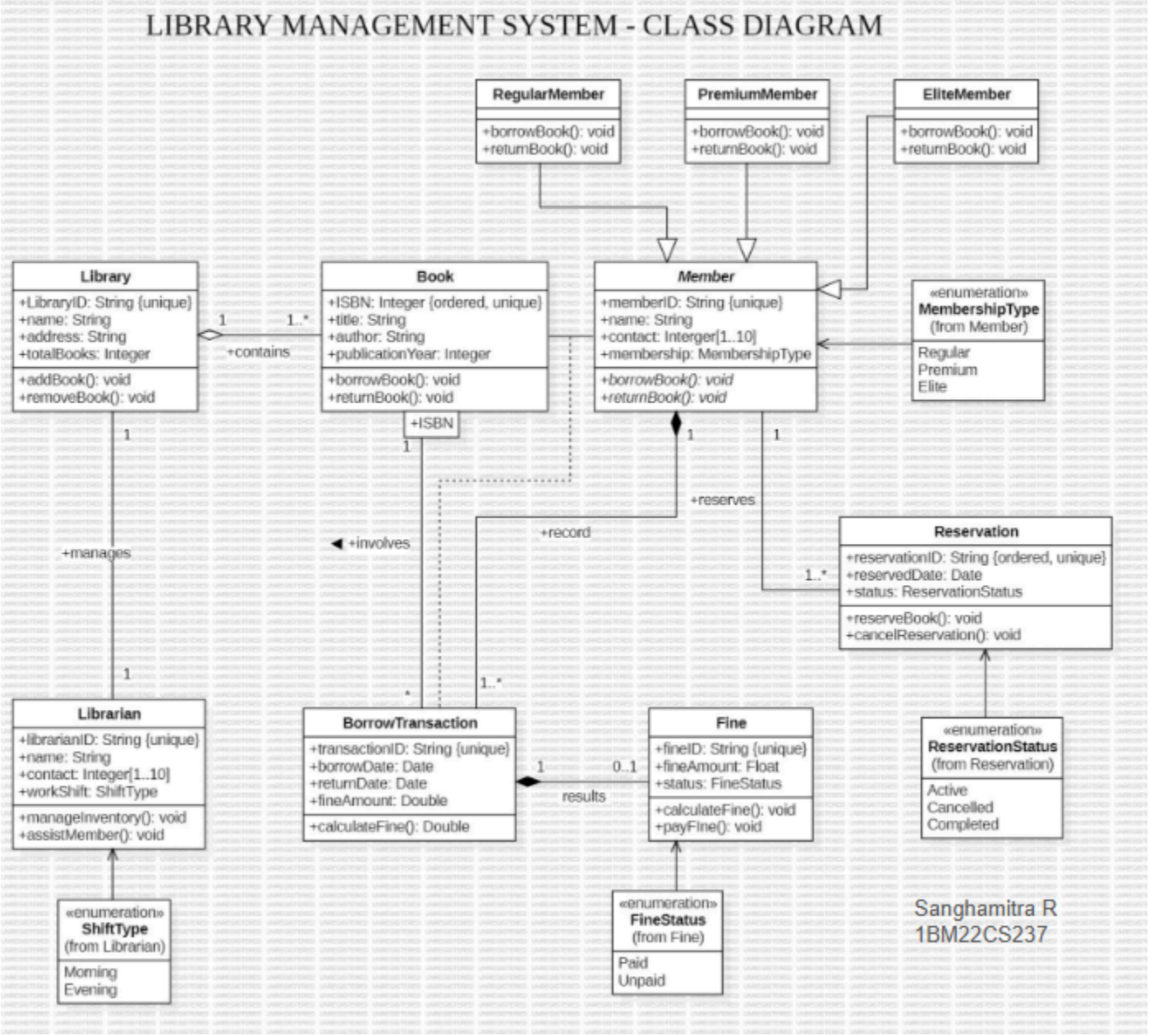
c) Portability : compatibility with multiple configurations.

8. Preliminary Schedule and Budget

Estimated time : 6-7 months

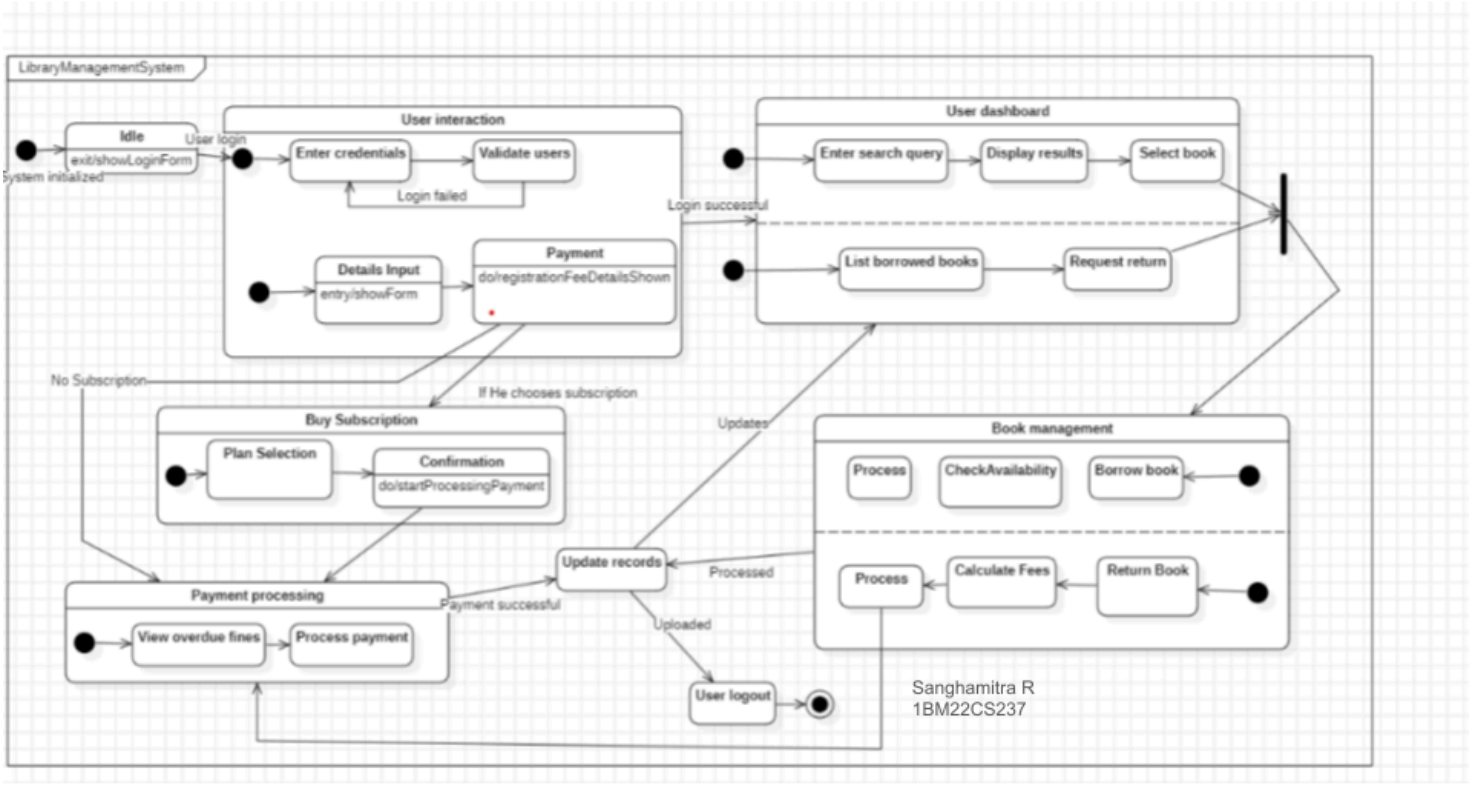
Estimated total budget : 15,00,000

CLASS DIAGRAM



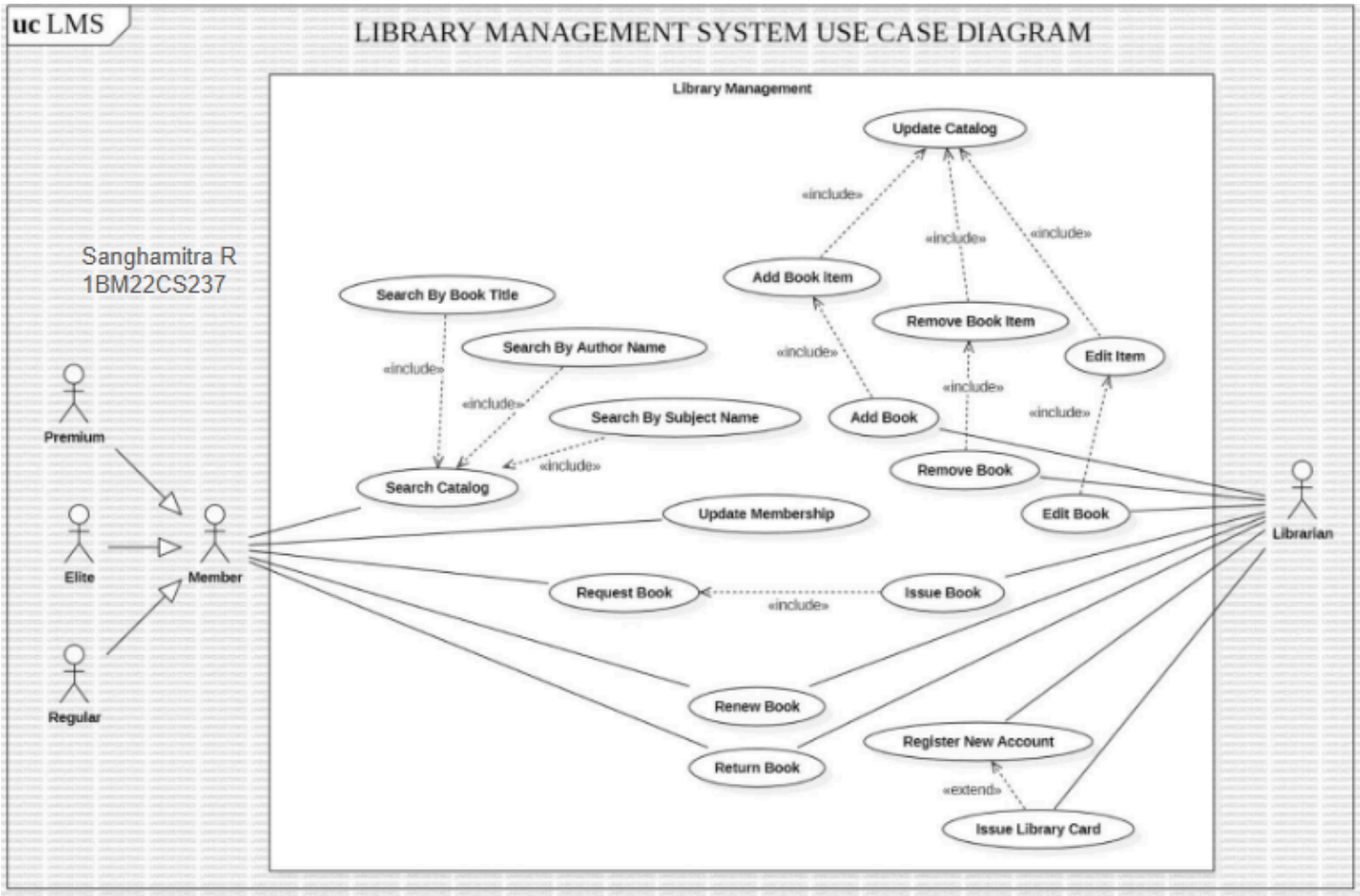
This UML class diagram illustrates the core components of a library management system. It depicts entities such as Library, Book, Member, Librarian, and their relationships. Key features include book reservations, borrowing transactions, and fine management. The diagram also includes enumerations for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

STATE DIAGRAM



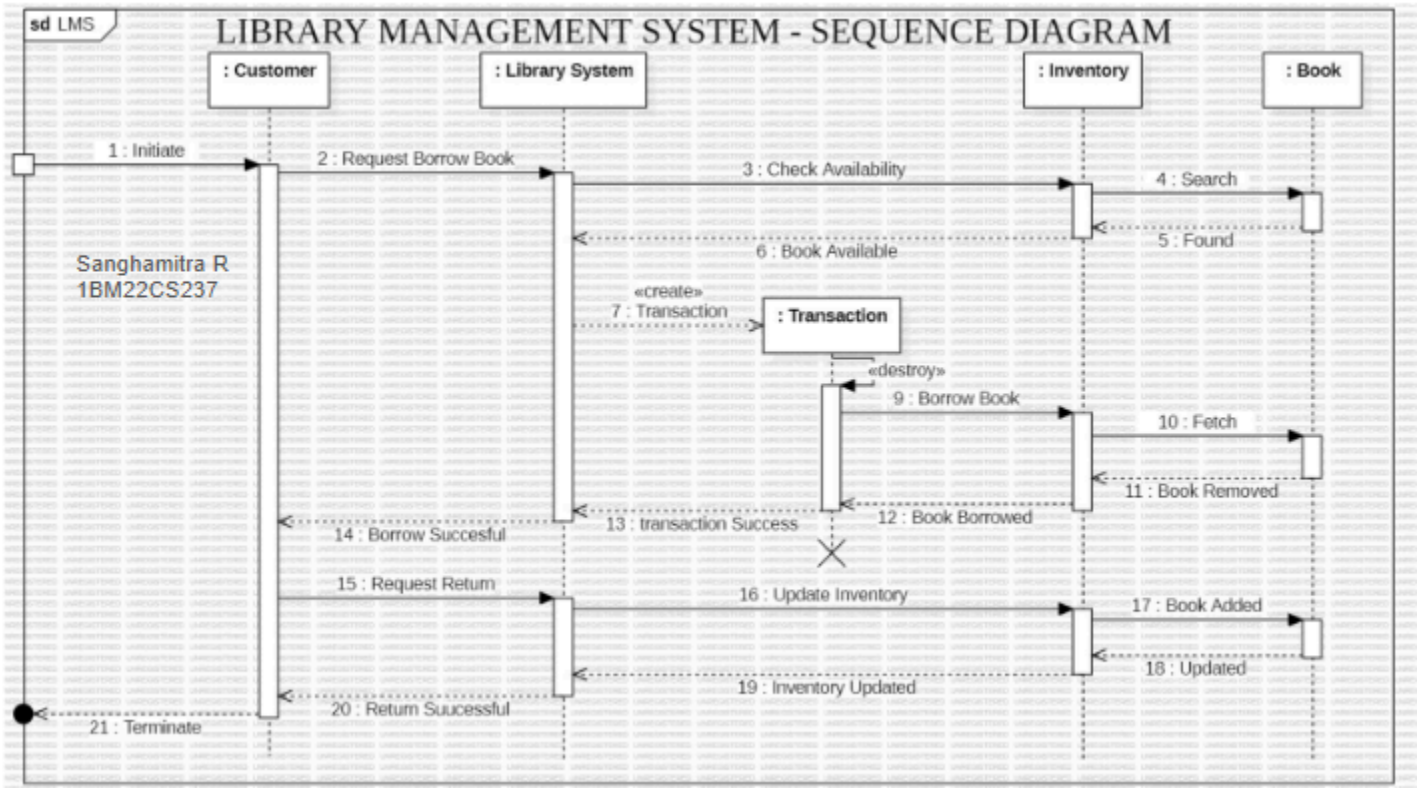
This state diagram illustrates the workflow of a library management system. It starts with the system initializing settings and monitoring user activity. Users can register and log in to access operations like catalog management, book management, borrowing operations, and catalog search. The system handles 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



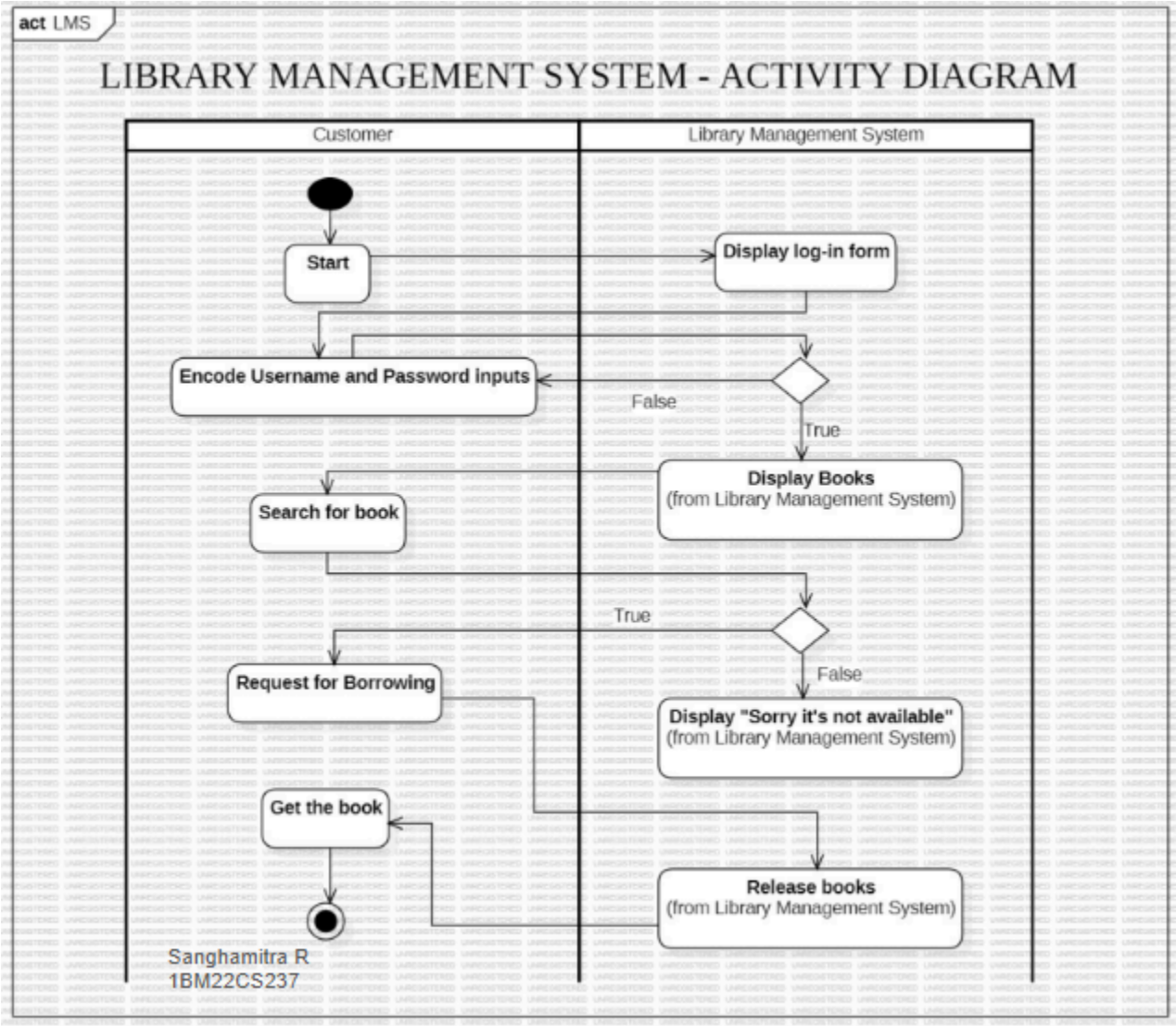
This UML use case diagram represents the functionalities of a Library Management System. It highlights interactions between primary actors such as Librarian, Member, and Administrator. Key use cases include book search, book issue and return, membership management, fine calculation, inventory updates, and report generation. Relationships like <<include>> and <<extend>> are used to illustrate dependencies, such as fine calculation being an extension of the book return process. The diagram provides a clear overview of how the actors and use cases interact to ensure efficient library operations.

SEQUENCE DIAGRAM



This UML sequence diagram illustrates the interactions within a Library Management System. It captures the sequence of messages exchanged between objects such as Member, Librarian, Database, and Notification Service. Key processes include book search, book issue, and return, where the Member requests an operation, the Librarian validates it, and the system updates the database. Additional actions like overdue notifications are handled by the Notification Service. The diagram visually represents the order of interactions, ensuring clarity in understanding the system's dynamic behavior.

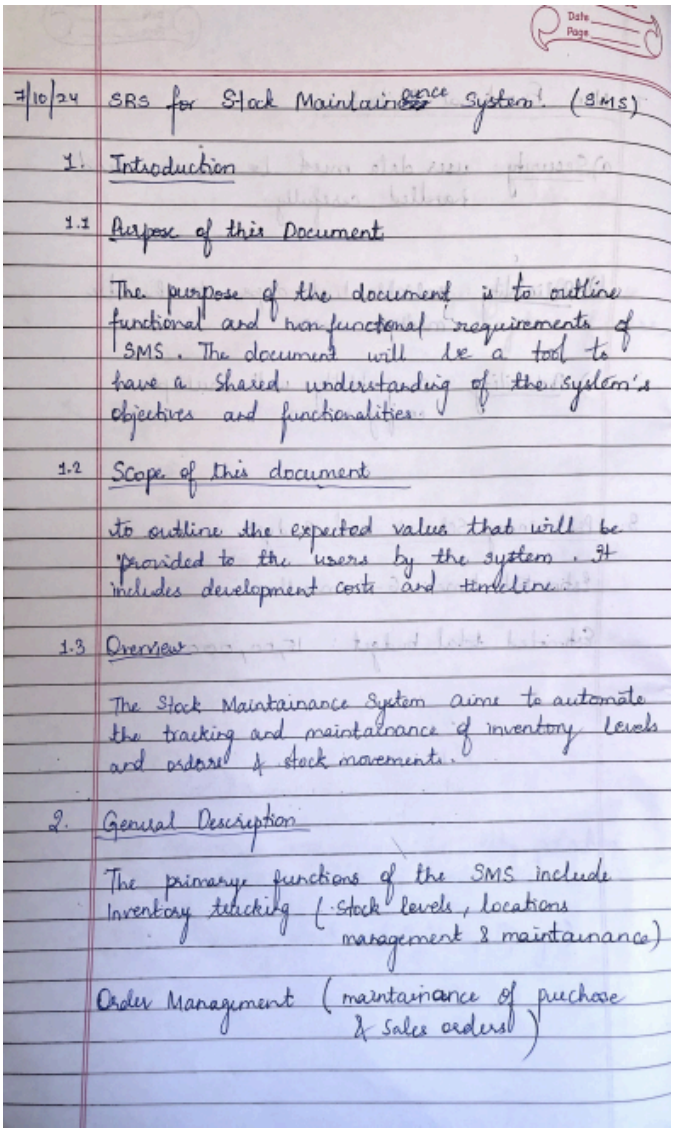
ACTIVITY DIAGRAM



This UML activity diagram illustrates the workflow of a Library Management System. It represents key activities such as searching for a book, verifying membership, issuing or returning a book, and updating the system database. The diagram begins with the user logging into the system and proceeds through decision points like book availability and overdue status. Activities such as fine calculation and sending notifications are also depicted. Control flows and swimlanes are used to clearly show responsibilities between actors like Member, Librarian, and System, ensuring an organized representation of the process.

STOCK MAINTENANCE SYSTEM

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)



7/10/24 SRS for Stock Maintainance System (SMS)

1. Introduction

1.1 Purpose of this Document

The purpose of the document is to outline functional and non-functional requirements of SMS. The document will be a tool to have a shared understanding of the system's objectives and functionalities.

1.2 Scope of this document

to outline the expected value that will be provided to the users by the system. It includes development costs and timeline.

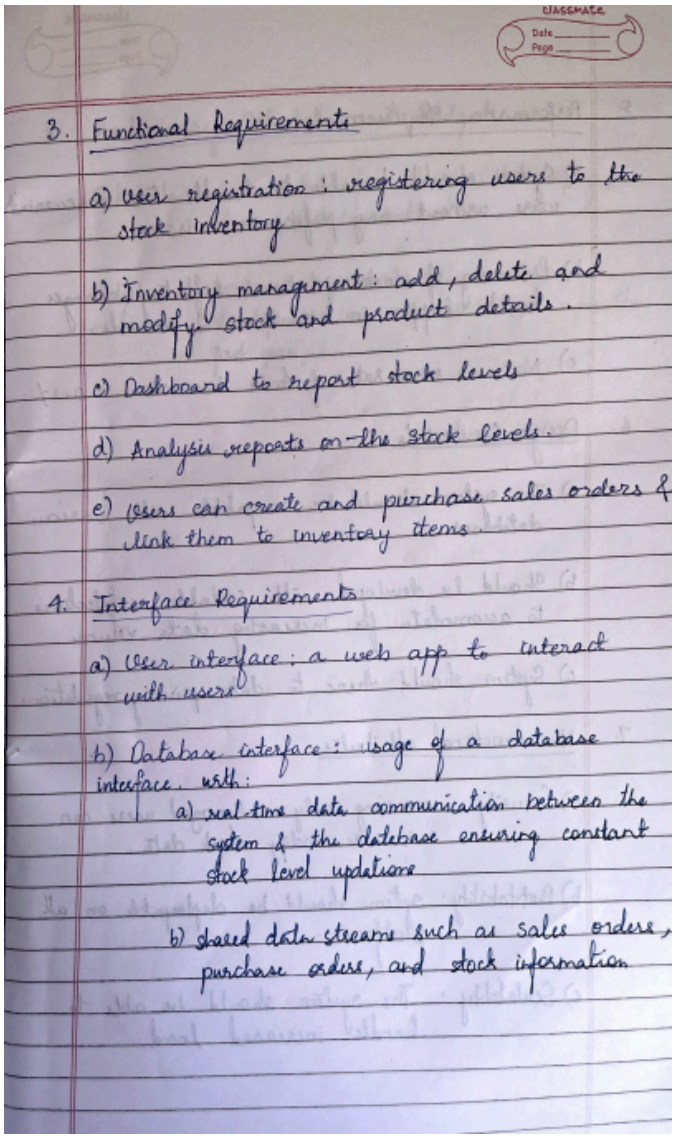
1.3 Overview

The Stock Maintainance System aims to automate the tracking and maintainance of inventory levels and orders & stock movements.

2. General Description

The primary functions of the SMS include Inventory tracking (Stock levels, locations management & maintainance)

Order Management (maintainance of purchase & Sales orders)



3. Functional Requirements

a) User registration: registering users to the stock inventory

b) Inventory management: add, delete and modify stock and product details.

c) Dashboard to report stock levels

d) Analysis reports on the stock levels.

e) Users can create and purchase sales orders & link them to inventory items.

4. Interface Requirements

a) User interface: a web app to interact with users

b) Database interface: usage of a database interface with:

a) real-time data communication between the system & the database ensuring constant stock level updations

b) shared data streams such as sales orders, purchase orders, and stock information

5. Performance Requirements:

a) System should be able to handle 500 concurrent users without any performance issues.

b) Processing of stock updates & reflecting changes should happen in less than 2 seconds.

c) Maximum error rate of stock transaction = 0.1%

6. Design Constraints

a) The system should be compatible with various databases

b) Should be developed with scalable architecture to accomodate for increasing data volume.

c) System should adhere to data privacy regulations

7. Non-Functional attributes

a) Security : ensuring only authorized users can view or modify stock data.

b) Portability: system should be deployable on all platforms.

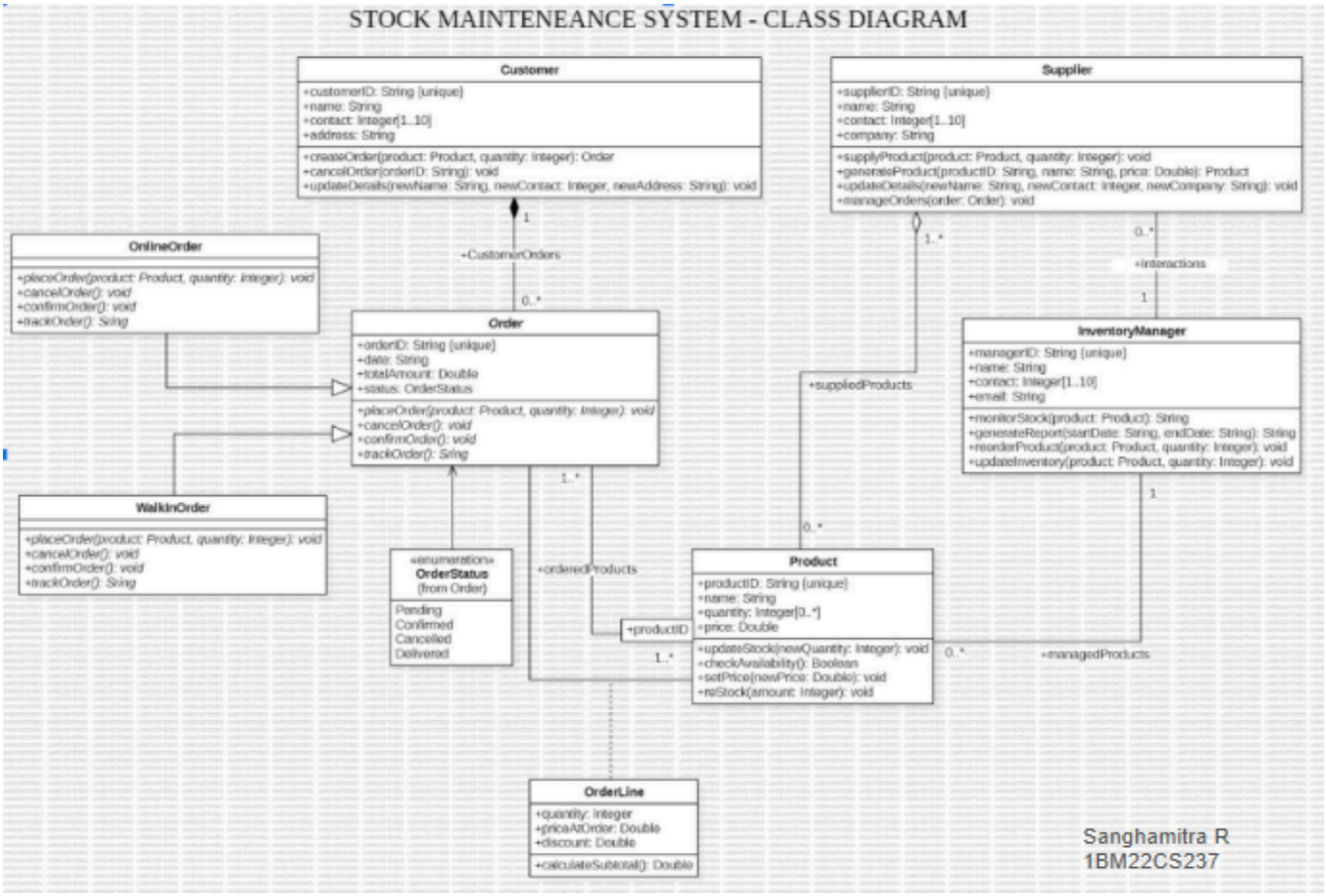
c) Scalability : The system should be able to handle increased load

8. Preliminary Schedule and Budget

Schedule : 6 months for design, developing, testing & deployment.

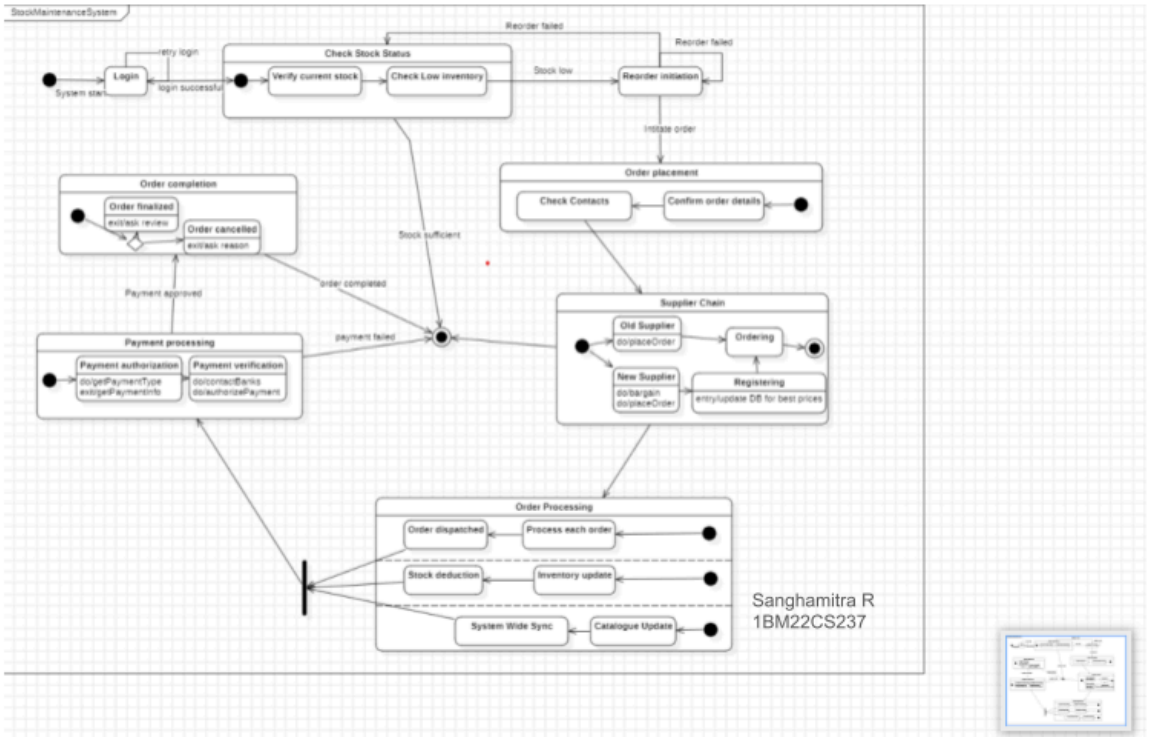
Budget : $50,000 (hardware, software licenses, development time, & maintainance for the first year)

CLASS DIAGRAM



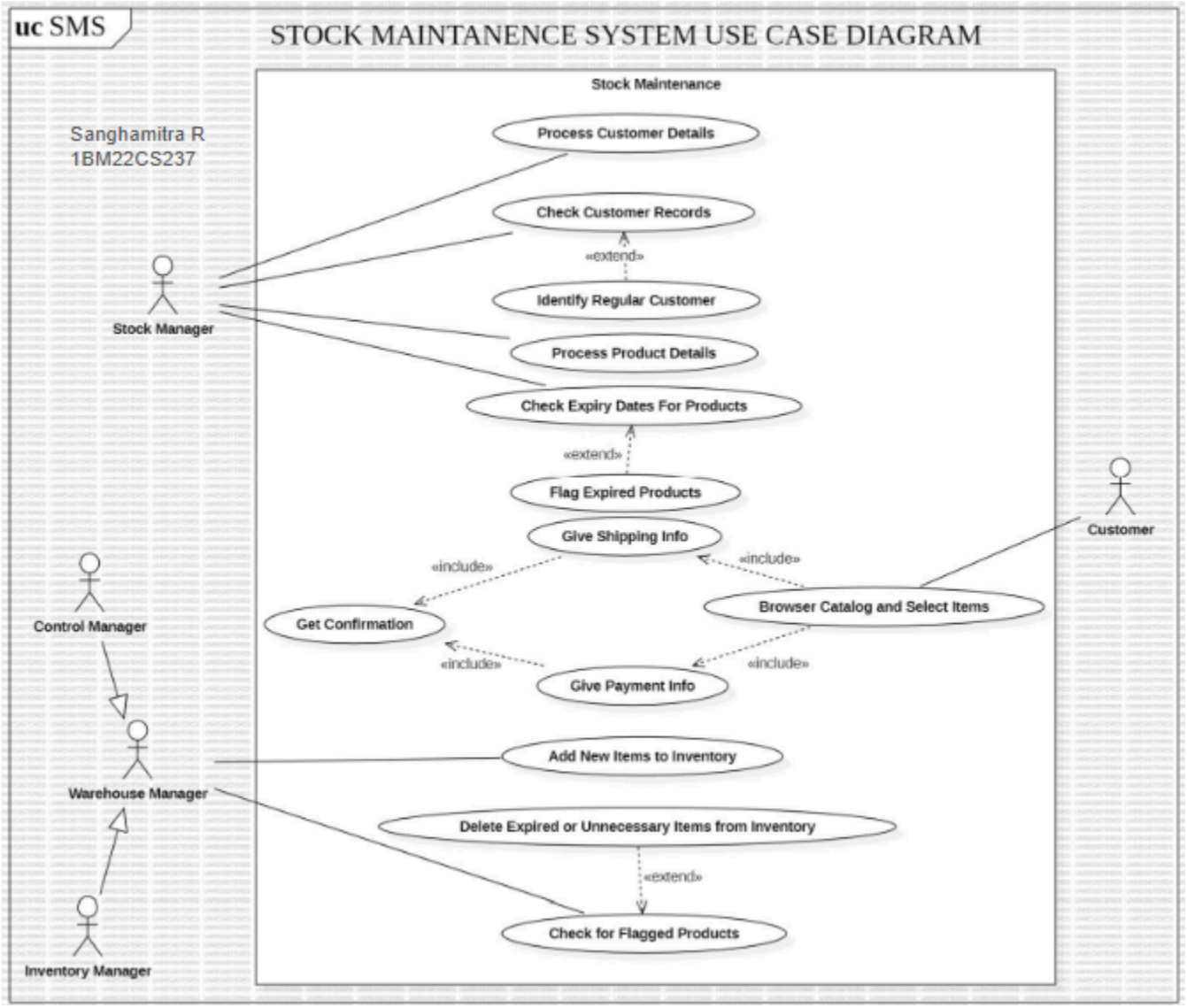
This UML class diagram models the structure of a Stock Management System. It identifies key classes such as Product, Supplier, Stock, Order, and User, along with their attributes and methods. Relationships like associations and aggregations depict how objects interact, such as a Product being linked to Stock and Supplier, and an Order being associated with multiple Products. The diagram provides a detailed blueprint of the system's data model and its relationships.

STATE DIAGRAM



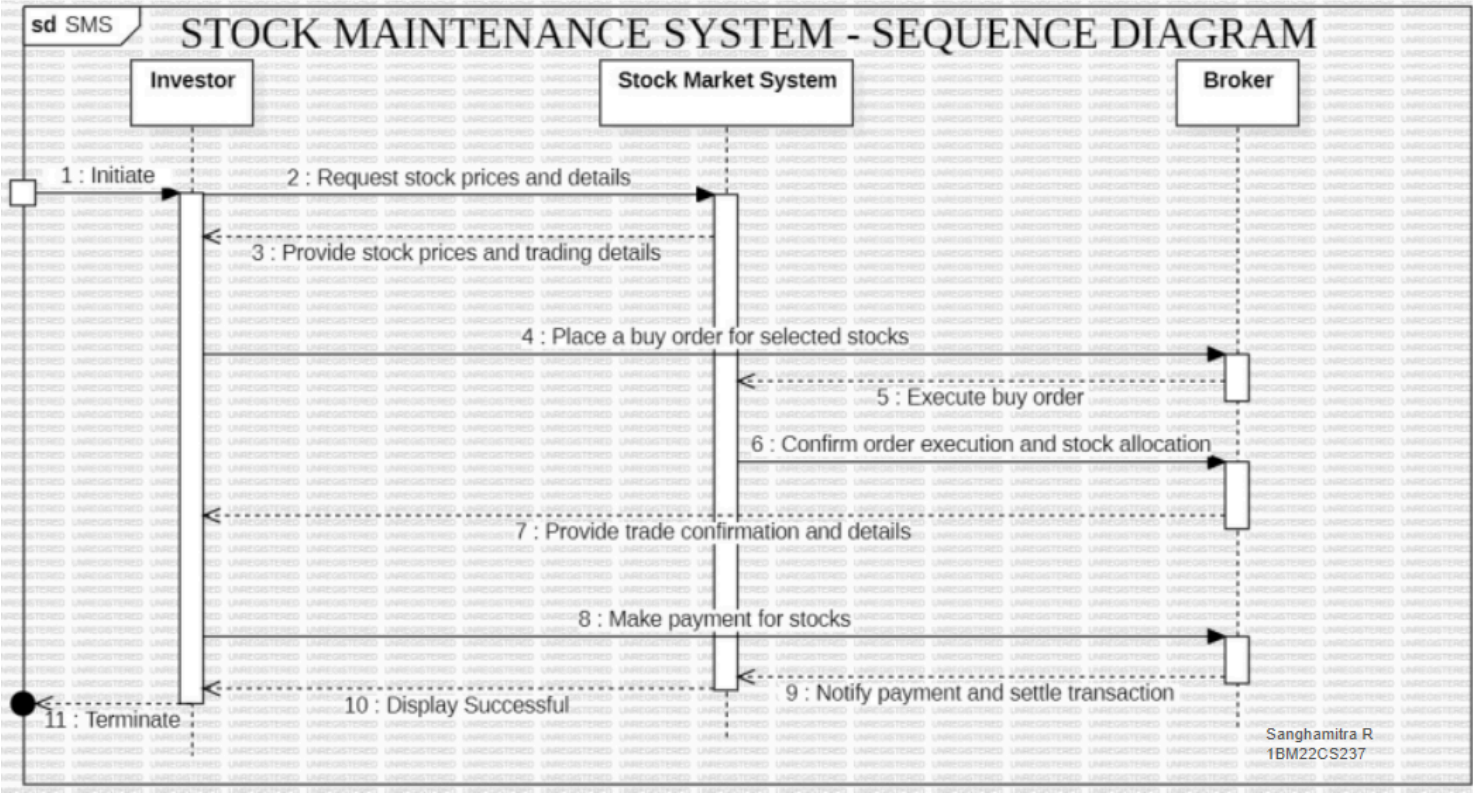
This UML state diagram represents the life cycle of a stock item in a Stock Management System. It transitions through states such as "In Stock," "Low Stock," "Out of Stock," and "Reordered." Events like stock depletion, threshold triggers, and order placements drive transitions between states. Actions like notifying suppliers or updating records accompany state changes, ensuring a clear understanding of the dynamic behavior of stock items.

USECASE DIAGRAM



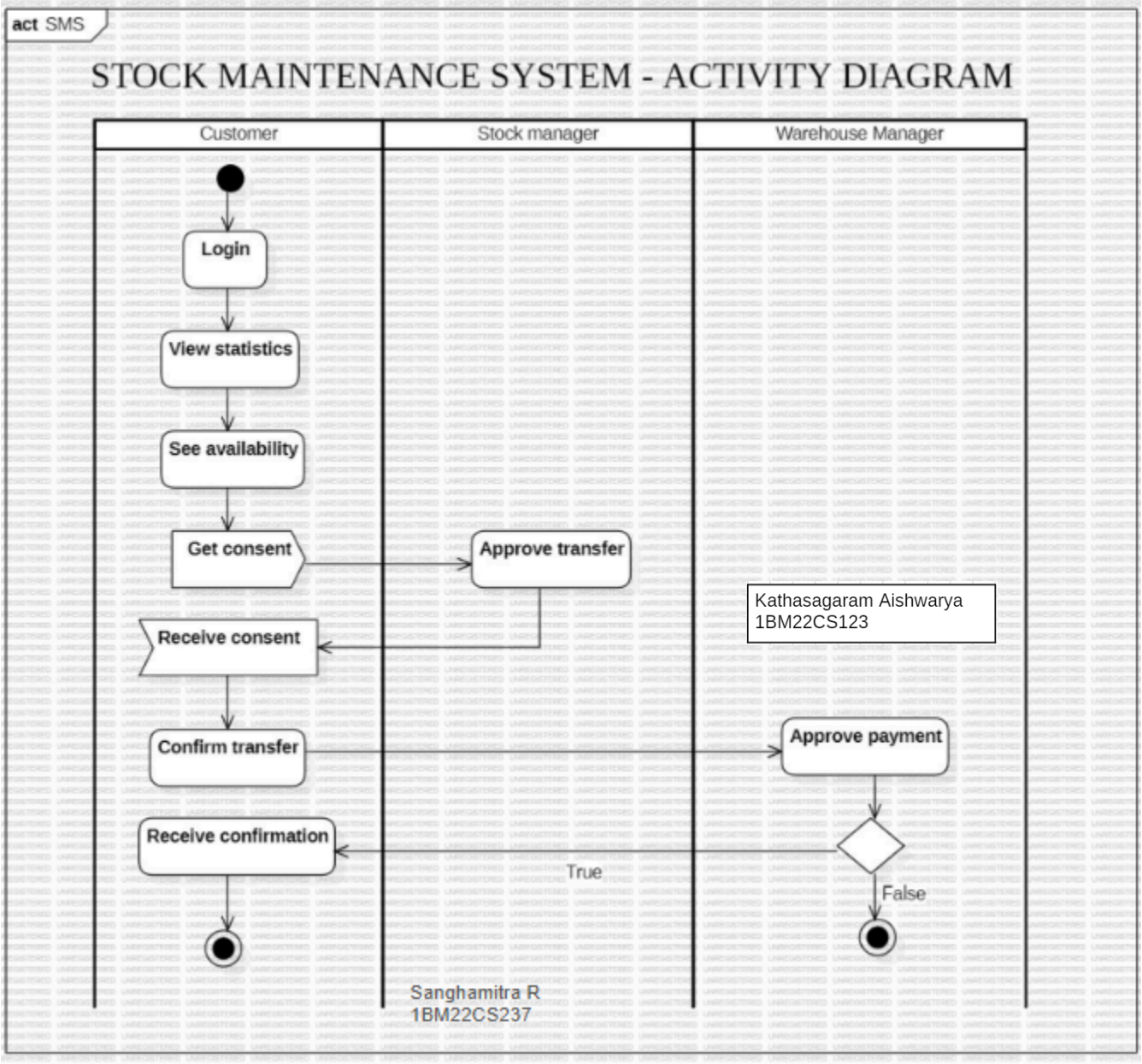
This UML use case diagram outlines the functionalities of a Stock Management System. It highlights interactions between primary actors such as Store Manager, Supplier, and Inventory Staff. Key use cases include stock entry, stock update, supplier management, stock level monitoring, generating reports, and handling low-stock alerts. Relationships like <<include>> and <<extend>> are used to depict dependencies, such as low-stock alerts extending stock level monitoring. The diagram provides a clear overview of system functionalities and actor interactions.

SEQUENCE DIAGRAM



This UML sequence diagram captures the interactions within a Stock Management System. It shows the sequence of messages exchanged between objects like Store Manager, Inventory System, Supplier, and Notification Service. Key processes include stock addition, stock removal, and supplier order placement. The Store Manager initiates actions, the Inventory System processes requests, and notifications are sent for stock reorders. The diagram visually demonstrates the order of interactions, highlighting system workflow and responsibilities.

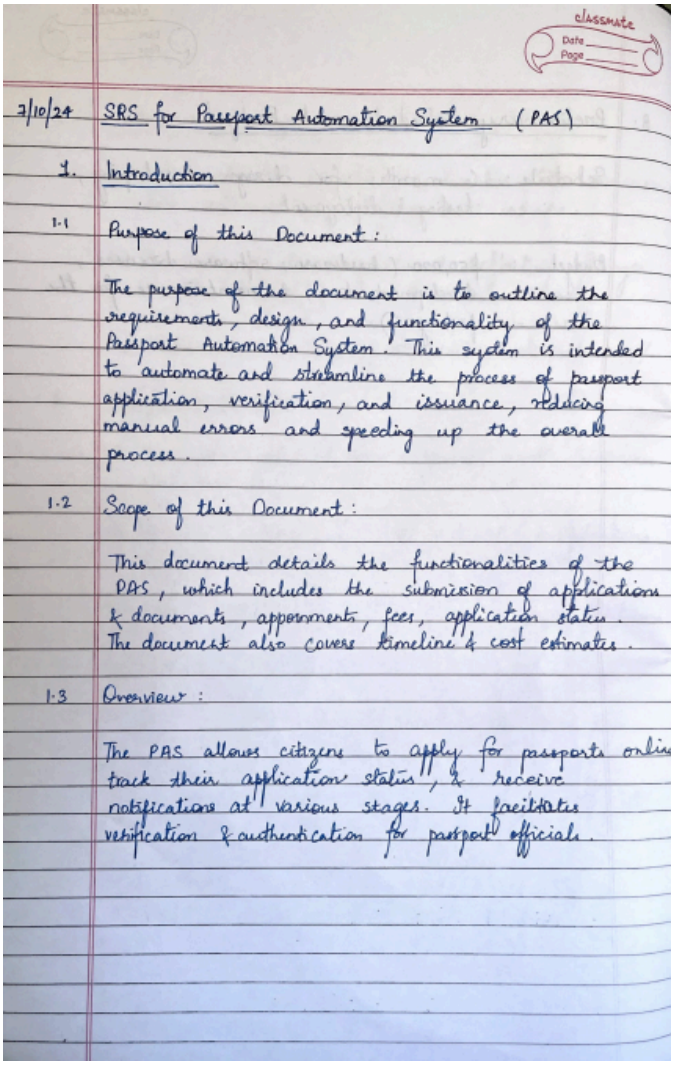
ACTIVITY DIAGRAM



This UML activity diagram represents the workflow of a Stock Management System. It begins with logging into the system and proceeds through activities such as checking stock levels, updating stock records, and generating reports. Decision points like "Is stock below threshold?" lead to actions such as notifying suppliers or placing orders. Swimlanes are used to distinguish responsibilities among Store Manager, Inventory Staff, and System. The diagram ensures a clear representation of the system's operational flow.

PASSPORT AUTOMATION SYSTEM

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)



7/10/24 SRS for Passport Automation System (PAS)

1. Introduction

1.1 Purpose of this Document:

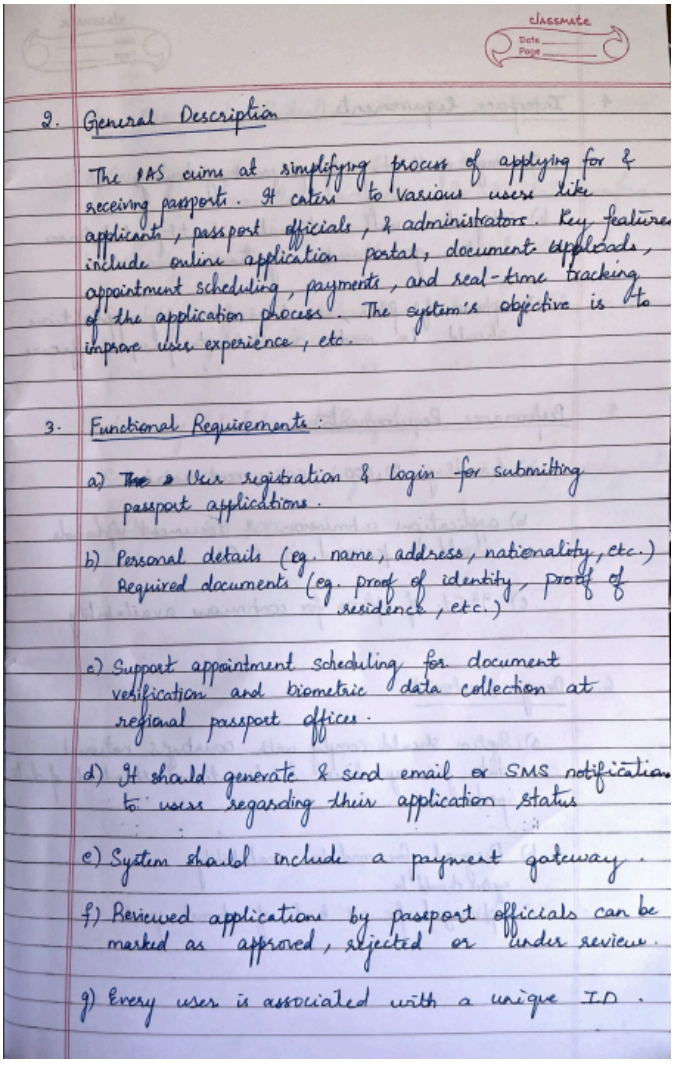
The purpose of the document is to outline the requirements, design, and functionality of the Passport Automation System. This system is intended to automate and streamline the process of passport application, verification, and issuance, reducing manual errors and speeding up the overall process.

1.2 Scope of this Document:

This document details the functionalities of the PAS, which includes the submission of applications & documents, appointments, fees, application status. The document also covers timeline & cost estimates.

1.3 Overview:

The PAS allows citizens to apply for passports online, track their application status, & receive notifications at various stages. It facilitates verification & authentication for passport officials.



2. General Description

The PAS aims at simplifying process of applying for & receiving passports. It caters to various users like applicants, passport officials, & administrators. Key features include online application portal, document uploads, appointment scheduling, payments, and real-time tracking of the application process. The system's objective is to improve user experience, etc.

3. Functional Requirements:

a) User registration & login for submitting passport applications.

b) Personal details (eg. name, address, nationality, etc.) Required documents (eg. proof of identity, proof of residence, etc.)

c) Support appointment scheduling for document verification and biometric data collection at regional passport offices.

d) It should generate & send email or SMS notifications to users regarding their application status

e) System should include a payment gateway.

f) Reviewed applications by passport officials can be marked as approved, rejected or under review.

g) Every user is associated with a unique ID.

2. General Description

The PAS aims at simplifying process of applying for & receiving passports. It caters to various users like applicants, passport officials, & administrators. Key features include online application portal, document uploads, appointment scheduling, payments, and real-time tracking of the application process. The system's objective is to improve user experience, etc.

3. Functional Requirements:

a) The system should allow User registration & login for submitting passport applications.

b) Personal details (eg. name, address, nationality, etc.) Required documents (eg. proof of identity, proof of residence, etc.)

c) Support appointment scheduling for document verification and biometric data collection at regional passport offices.

d) It should generate & send email or SMS notifications to users regarding their application status

e) System should include a payment gateway.

f) Reviewed applications by passport officials can be marked as approved, rejected or under review.

g) Every user is associated with a unique ID.

7. Non-Functional Attributes:

a) Security: implement multi-factor authentication & secure data encryption for all data transfers

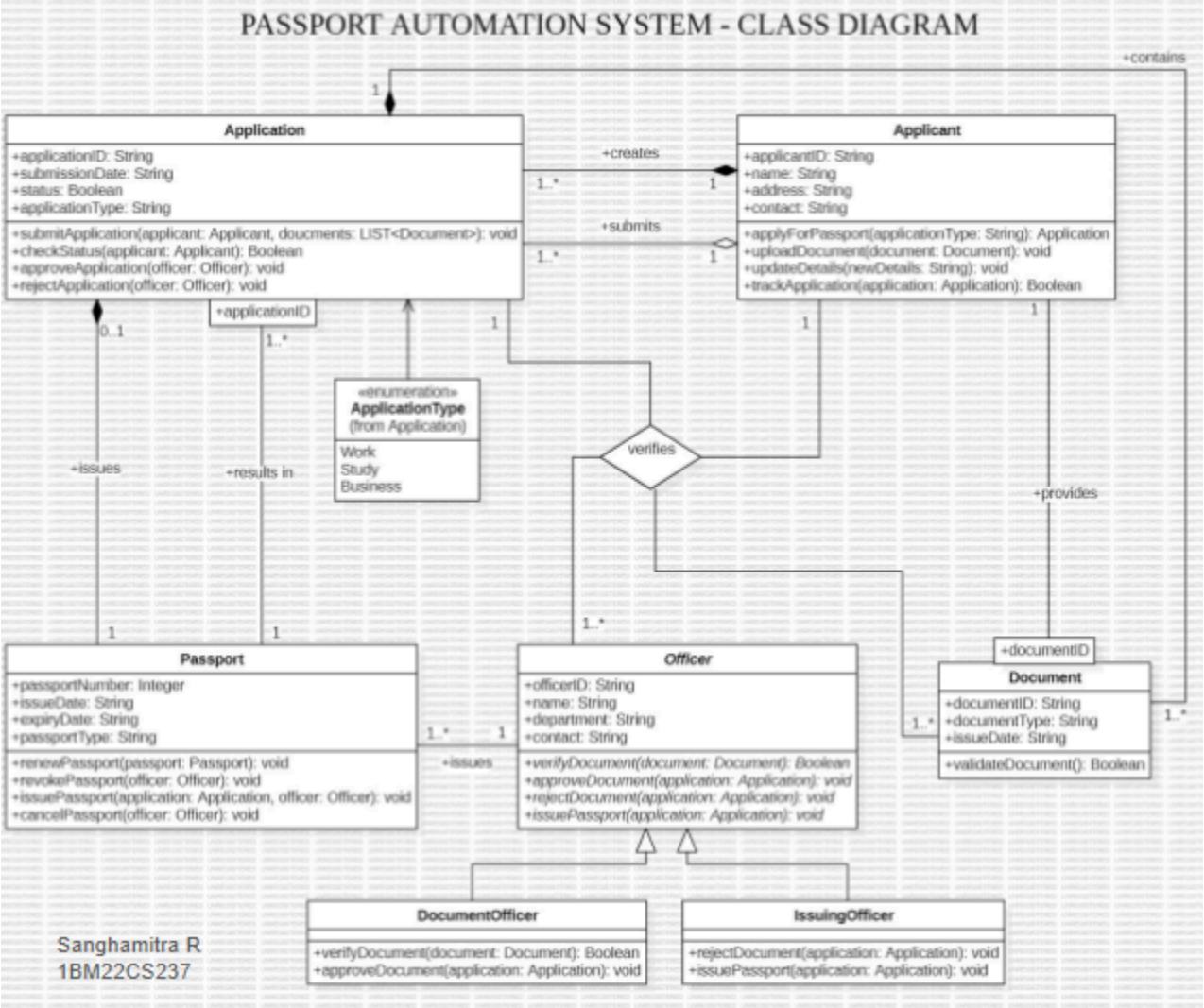
b) Portability: should be accessible on various devices.

c) Reusability: different modules should be reusable for other government services.

8. Preliminary Schedule and Budget

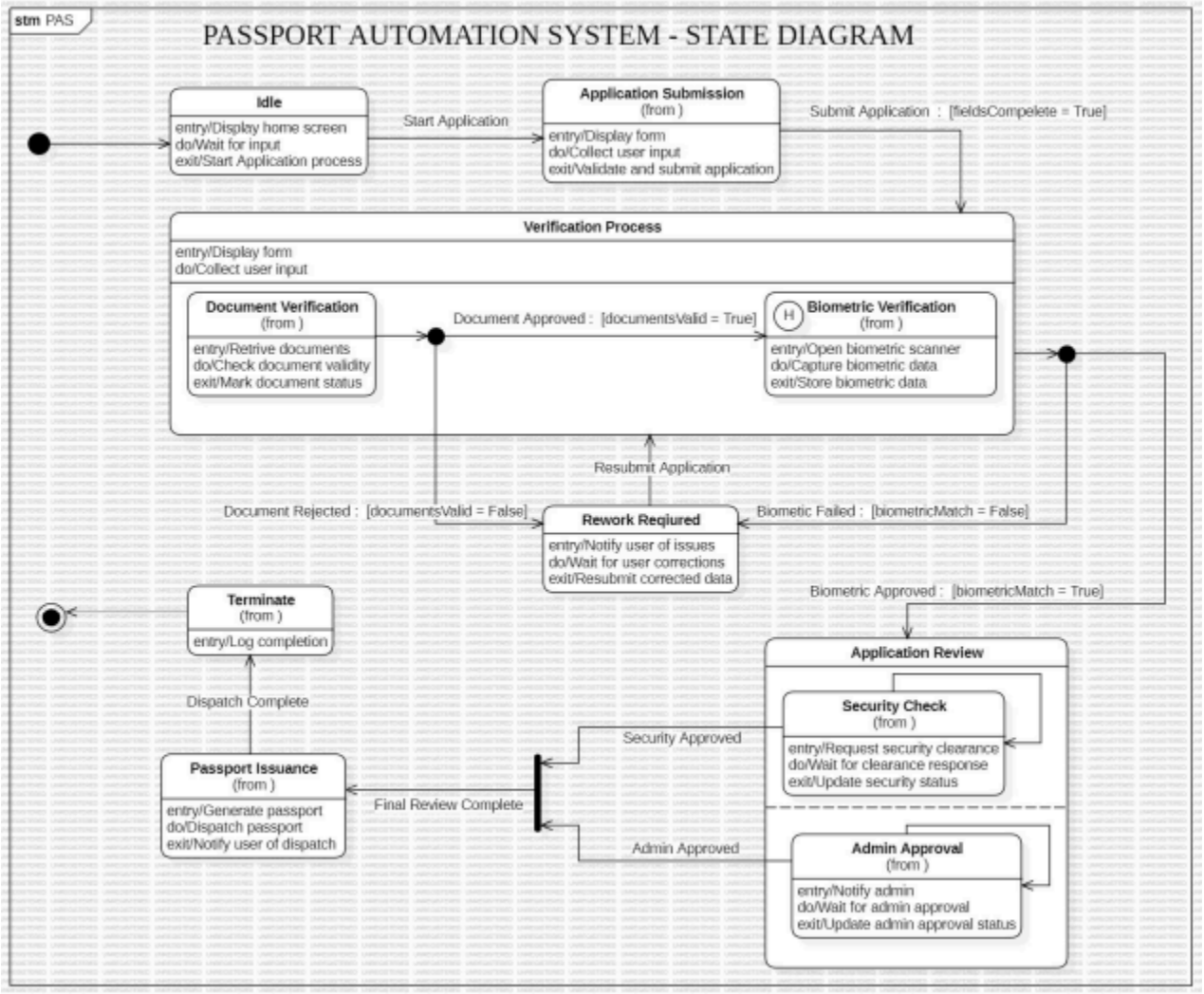
- Schedule: 9 months (design, development, testing, deployment)
- Budget: $75,000

CLASS DIAGRAM



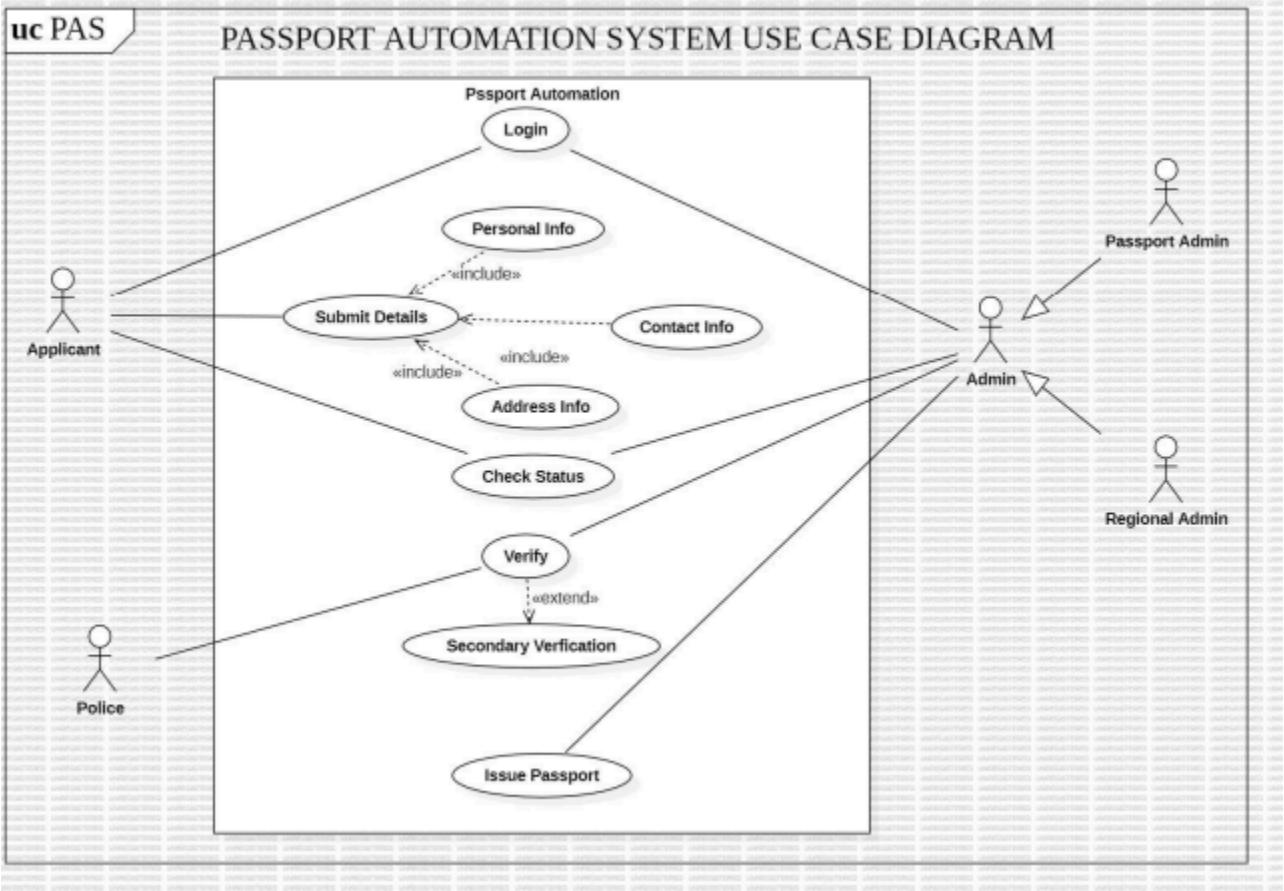
This UML class diagram models the structure of a Passport Automation System. It identifies key classes such as Applicant, Passport, BiometricData, Document, VerificationAgency, and Notification. Each class includes attributes like name, address, and status, along with methods such as submitApplication(), verifyDocuments(), and issuePassport(). Relationships such as associations and dependencies depict interactions, such as an Applicant being linked to Documents and BiometricData. The diagram provides a structural blueprint of the system.

STATE DIAGRAM



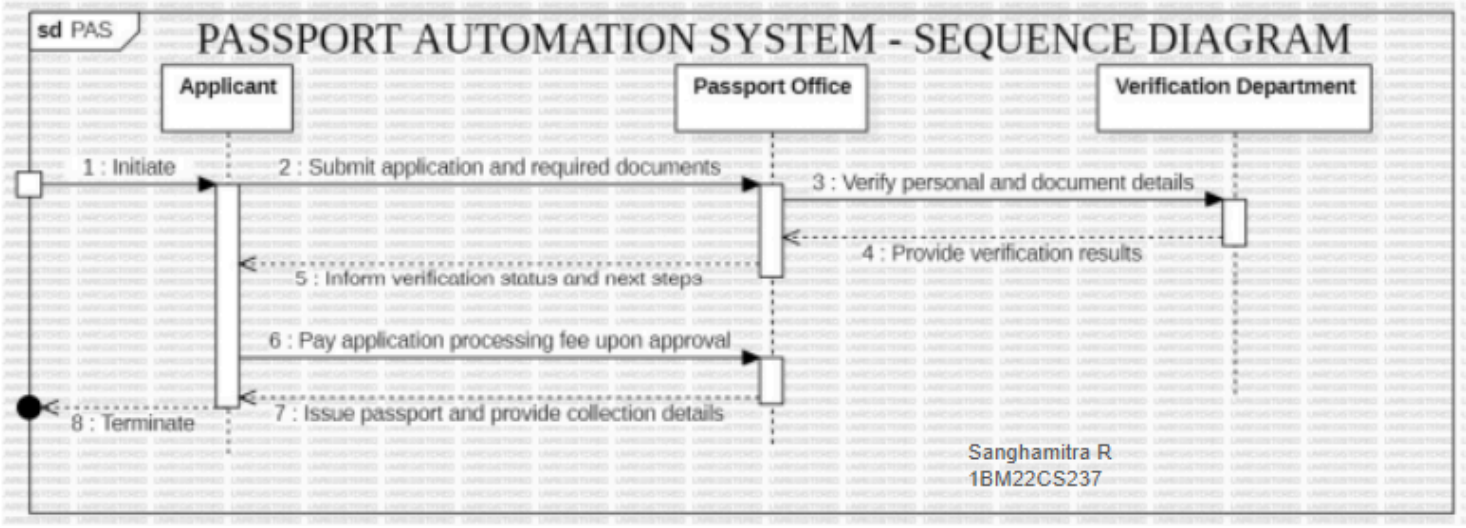
This UML state diagram represents the life cycle of a passport application in a Passport Automation System. It transitions through states such as "Application Submitted," "Documents Under Review," "Biometric Data Captured," "Verification In Progress," "Approved," and "Passport Issued." Events like document validation, biometric verification, and external record checks trigger transitions. Actions like notifying the applicant or issuing the passport accompany state changes, ensuring a clear representation of the application process dynamics.

USE CASE DIAGRAM



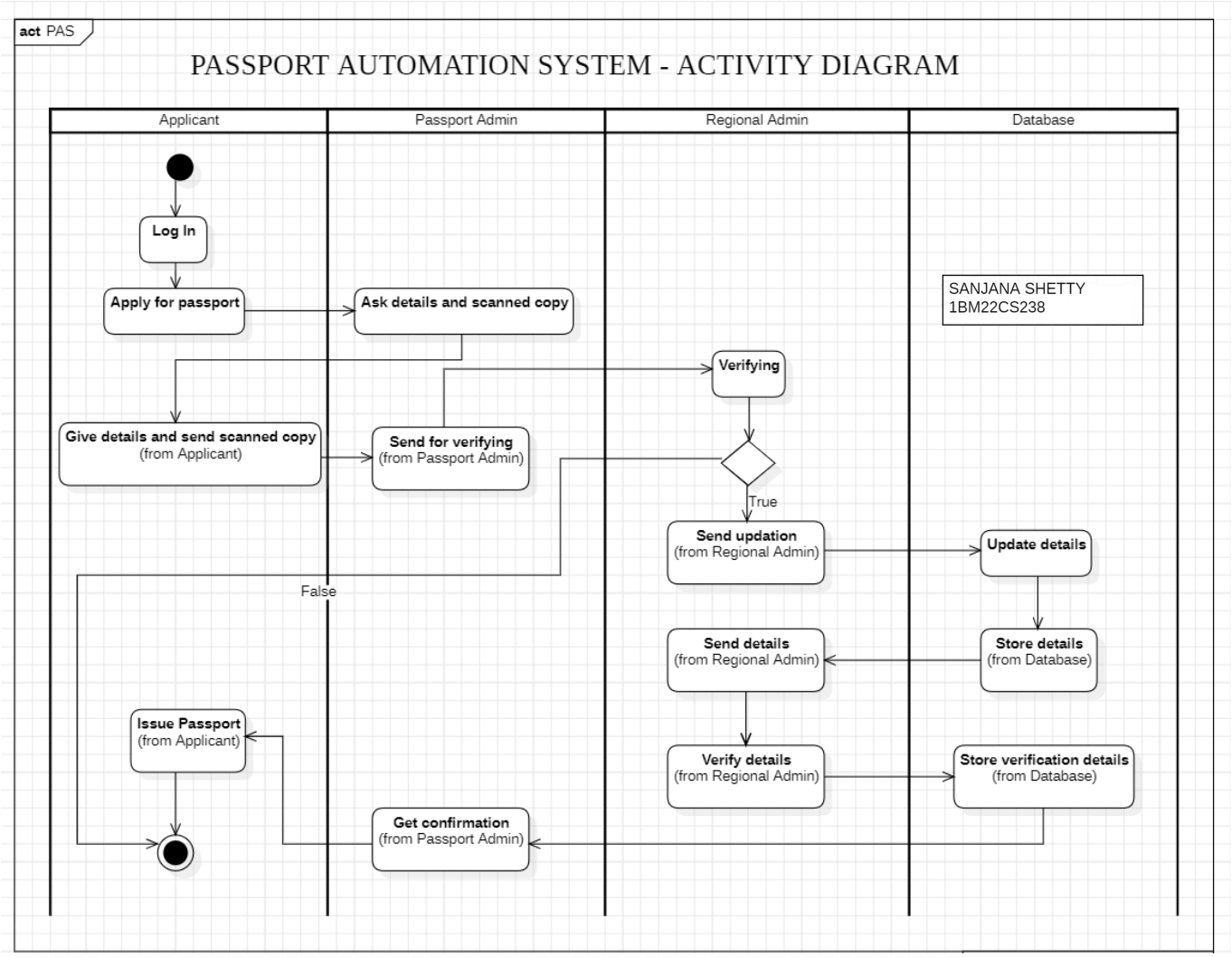
This UML use case diagram outlines the functionalities of a Passport Automation System. The key actors include Applicant, Passport Officer, and External Verification Agency. Key use cases include application submission, document verification, biometric data capture, application processing, external record verification, status notification, and passport issuance. Relationships like <<include>> and <<extend>> illustrate dependencies, such as biometric data capture being included in application processing. The diagram provides a clear overview of the automated processes and actor interactions.

SEQUENCE DIAGRAM



This UML sequence diagram captures the interactions within a Passport Automation System. It details the sequence of messages between objects like Applicant, Passport System, Biometric System, Verification Agency, and Notification System. Key processes include submitting an application, verifying documents, capturing biometric data, forwarding data for external verification, and notifying the applicant about the application status. The diagram demonstrates the orderly flow of actions and system interactions.

ACTIVITY DIAGRAM



This UML activity diagram represents the workflow of a Passport Automation System. It begins with the applicant logging into the system and submitting the application. Subsequent activities include uploading documents, biometric data capture, and external verification. Decision points like "Are documents valid?" and "Is biometric data verified?" lead to further actions, such as re-uploading documents or approving the application. Swimlanes clearly define the responsibilities of the Applicant, System, and Passport Office. The diagram highlights the automation of passport processing tasks.