**B.M.S. COLLEGE OF ENGINEERING BENGALURU**
Autonomous Institute, Affiliated to VTU


Lab Record

# Object-Oriented Modeling

*Submitted in partial fulfillment for the 5th Semester Laboratory*

Bachelor of Engineering
in
Computer Science and Engineering

*Submitted by:*

**Kathasagaram Aishwarya**

1BM22CS123

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
September-January 2025

# B.M.S. COLLEGE OF ENGINEERING

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING


## *CERTIFICATE*

This is to certify that the **Object-Oriented Modelling** (**23CS5PCOOM**) laboratory has been carried out by Kathasagaram Aishwarya (1BM22CS123) during the 5th Semester Sep 2024-Jan 2025.

Signature of the Faculty Incharge:
Sandhya A Kulkarni
Assisstant Professor
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

# Table of Contents



Github Link: https://github.com/Aish-kathasagaram/OOM-Lab

# 1. Hotel Management System

## 1.1 Problem Statement

The Hotel Management System aims to streamline hotel operations by managing room bookings, check-ins, check-outs, and billing processes. The system includes classes like Guest, Room, Reservation, and Invoice to encapsulate data and behaviors. It should support the creation and modification of reservations, track room availability, and generate invoices. Relationships such as a guest making multiple reservations and a room being associated with multiple reservations over time must be maintained. The system should also include features for managing staff and services to ensure smooth hotel operations.

The proposed system aims to address the following critical areas:

1. Reservation Management: Facilitate online booking with features such as real-time availability checks, secure payment processing, and automated booking confirmations to improve convenience and operational efficiency.

2. Guest Services: Simplify check-in and check-out procedures, efficiently handle guest requests, and provide detailed information about hotel amenities to ensure a seamless and satisfying guest experience.

3. Inventory Management: Provide tools to monitor room availability, manage room rates dynamically, assign rooms, and oversee maintenance activities to ensure optimal resource utilization.

4. Financial Management: Enable accurate invoice generation, secure payment processing, effective account management, and the creation of comprehensive financial reports to support financial oversight and decision-making.

5. Employee Management: Streamline employee scheduling, track attendance records, and manage access permissions to enhance workforce efficiency and maintain operational security.

## 1.2 Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to define the Software Requirements Specification (SRS) for a Hotel Management System (HMS). This system aims to streamline hotel operations, automate reservations,

improve customer experience, and optimize business processes.

### 1.2 Scope

The HMS is designed to serve hotels of all sizes, from boutique establishments to large chains. It provides a centralized platform for managing various hotel operations, including reservations, billing, customer relationships, and staff management. The system includes the following modules:

1. Reservation Management: Real-time room booking and availability tracking.
2. Billing and Payment Processing: Accurate billing and multiple payment methods.
3. Customer Relationship Management (CRM): Guest preferences and feedback management.
4. Staff Management: Task assignment and performance tracking.
5. Reporting and Analytics: Insights into occupancy rates, revenue trends, and customer satisfaction.

The HMS is scalable, adaptable, and user-friendly, ensuring that it evolves with the growing needs of the hospitality industry.

### 1.3 Definitions, Acronyms, and Abbreviations

- HMS: Hotel Management System
- CRM: Customer Relationship Management
- API: Application Programming Interface
- POS: Point of Sale
- GDPR: General Data Protection Regulation
- WCAG: Web Content Accessibility Guidelines
- SRS: Software Requirements Specification

### 1.4 References

- IEEE Std 830-1998: Software Requirements Specification
- GDPR Compliance Standards
- WCAG Accessibility Guidelines

### 1.5 Overview

This document describes the functional and non-functional requirements of the HMS. It includes detailed descriptions of system features, user roles, performance expectations, and design constraints.

## 2. Overall Description

### 2.1 Product Perspective

The HMS is a standalone application with options for API integration to third-party systems. It will consist of:

- A front-end GUI for administrators, staff, and guests.
- A back-end database for secure data storage and management.
- APIs for integrating external systems like payment gateways and CRM tools.

### 2.2 Product Features

The HMS includes the following core features:

1. User Management: Secure access for admins, staff, and guests.
2. Room Inventory Management: Tracks room availability and maintenance.
3. Multi-Language and Multi-Currency Support: Ensures global accessibility.
4. Notifications and Alerts: Real-time updates for bookings, cancellations, and tasks.
5. Compliance and Security: Data protection adhering to industry standards.

### 2.3 User Characteristics

The HMS will cater to the following user types:

1. Hotel Administrators: Manage hotel operations and generate reports.
2. Front Desk Staff: Handle reservations, check-ins, and guest services.
3. Housekeeping Staff: Manage cleaning schedules and room maintenance.
4. Guests: Book rooms, provide feedback, and access services.
5. IT and Support Teams: Maintain system functionality.

### 2.4 Assumptions and Dependencies

- The HMS will require an internet connection for real-time operations.
- Hardware such as desktops, tablets, and POS systems must be supported.
- Compliance with GDPR for data privacy.

## 3. Specific Requirements

### 3.1 Functional Requirements

#### 3.1.1 Reservation Management

- The system shall allow guests to book, modify, and cancel reservations online.
- The system shall track real-time room availability.

#### 3.1.2 Billing and Payment

- The system shall generate invoices automatically.
- The system shall support multiple payment methods (credit card, PayPal, etc.).

#### 3.1.3 Guest Management

- The system shall maintain guest profiles, including preferences and booking history.
- The system shall allow guests to provide feedback and track responses.

#### 3.1.4 Staff Management

- The system shall assign tasks to staff members and monitor their completion.
- The system shall track staff performance metrics.

#### 3.1.5 Reporting and Analytics

- The system shall generate customizable reports on occupancy rates, revenue trends, and performance metrics.
- The system shall provide visual dashboards for quick insights.

### 3.2 Performance Requirements

- The system shall handle up to 500 simultaneous users without performance degradation.
- The response time for any user action shall not exceed 2 seconds under normal load.
- Data backup and recovery processes shall be completed within 30 minutes.

### 3.3 Interface Requirements

#### 3.3.1 Graphical User Interface (GUI)

- The system shall have an intuitive and user-friendly interface with clear navigation.
- Customizable dashboards for administrators, staff, and guests.
- Responsive design for desktops, tablets, and mobile devices.

3.3.2 API Integration

- The system shall support third-party integrations such as payment gateways, booking platforms, and CRM tools.
- The system shall provide RESTful APIs for external applications.

3.3.3 Accessibility

- The system shall comply with WCAG standards to support users with disabilities.
- Multi-language support shall be provided for global accessibility.

3.3.4 Notification System

- The system shall send real-time notifications for bookings, check-outs, and housekeeping tasks.
- Configurable alerts for critical events, such as system errors or low room inventory.

3.3.5 Security Features

- Secure login with multi-factor authentication.
- Role-based access control for data confidentiality and integrity.
- End-to-end encryption for sensitive data.

3.4 Design Constraints

1. Hardware Compatibility: Supports standard hotel hardware, including desktops, tablets, and POS systems.
2. Scalability: Must accommodate increasing user loads and additional features.
3. Industry Standards: Ensure compliance with GDPR and other hospitality regulations.
4. Technology Stack: Utilize widely supported languages and frameworks for development and maintenance.

4. Non-Functional Requirements

4.1 Reliability

- The system shall have 99.9% uptime to ensure continuous availability.

4.2 Usability

- The interface shall be designed for users with varying levels of technical expertise.

4.3 Security

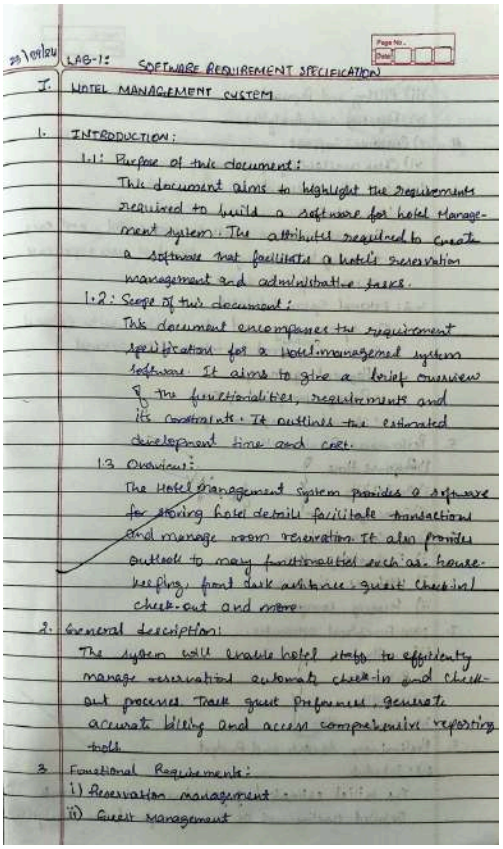
- The system shall use encryption to secure sensitive data.
- Role-based access control shall be implemented.

4.4 Maintainability

- The system shall have a modular architecture to allow easy updates and bug fixes.

4.5 Scalability

- The system shall handle increased user loads and data growth without performance degradation.


28/09/24 LAB-1: SOFTWARE REQUIREMENT SPECIFICATION

I. HOTEL MANAGEMENT SYSTEM

1. INTRODUCTION:

1.1: Purpose of this document:

This document aims to highlight the requirements required to build a software for hotel management system. The attributes required to create a software that facilitates a hotel's reservation management and administrative tasks.

1.2: Scope of this document:

This document encompasses the requirement specification for a Hotel management system software. It aims to give a brief overview of the functionalities, requirements and its constraints. It outlines the estimated development time and cost.

1.3 Overview:

The Hotel Management system provides a software for storing hotel details facilitate transactions and manage room reservation. It also provides outlook to many functionalities such as: housekeeping, front desk assistance, guest check-in/check-out and more.

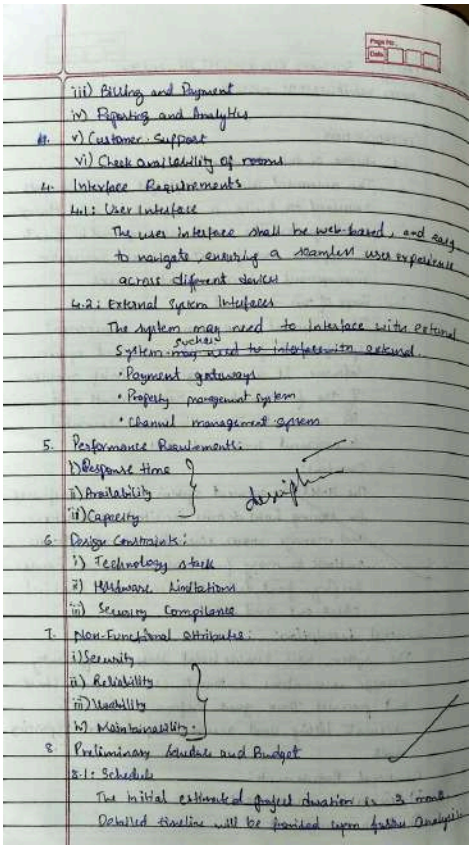
2. General description:

The system will enable hotel staffs to efficiently manage reservation, automate check-in and check-out process, track guest preferences, generate accurate billing and access comprehensive reporting tools.

3. Functional Requirements:

i) Reservation management.

ii) Guest management


iii) Billing and Payment

iv) Reporting and Analytics

v) Customer Support

vi) Check availability of rooms

4. Interface Requirements

4.1: User Interface

The user interface shall be web-based, and easy to navigate, ensuring a seamless user experience across different devices.

4.2: External System Interfaces

The system may need to interface with external systems (such as):

- Payment gateways
- Property management system
- Channel management system

5. Performance Requirements:

i) Response time

ii) Availability

iii) Capacity

6. Design Constraints:

i) Technology stack

ii) Hardware limitations

iii) Security Compliance

7. Non-Functional Attributes:

i) Security

ii) Reliability

iii) Usability

iv) Maintainability.

8. Preliminary Schedule and Budget

8.1: Schedule

The initial estimated project duration is 3 months. Detailed timeline will be provided upon further analysis.

## CLASS DIAGRAM

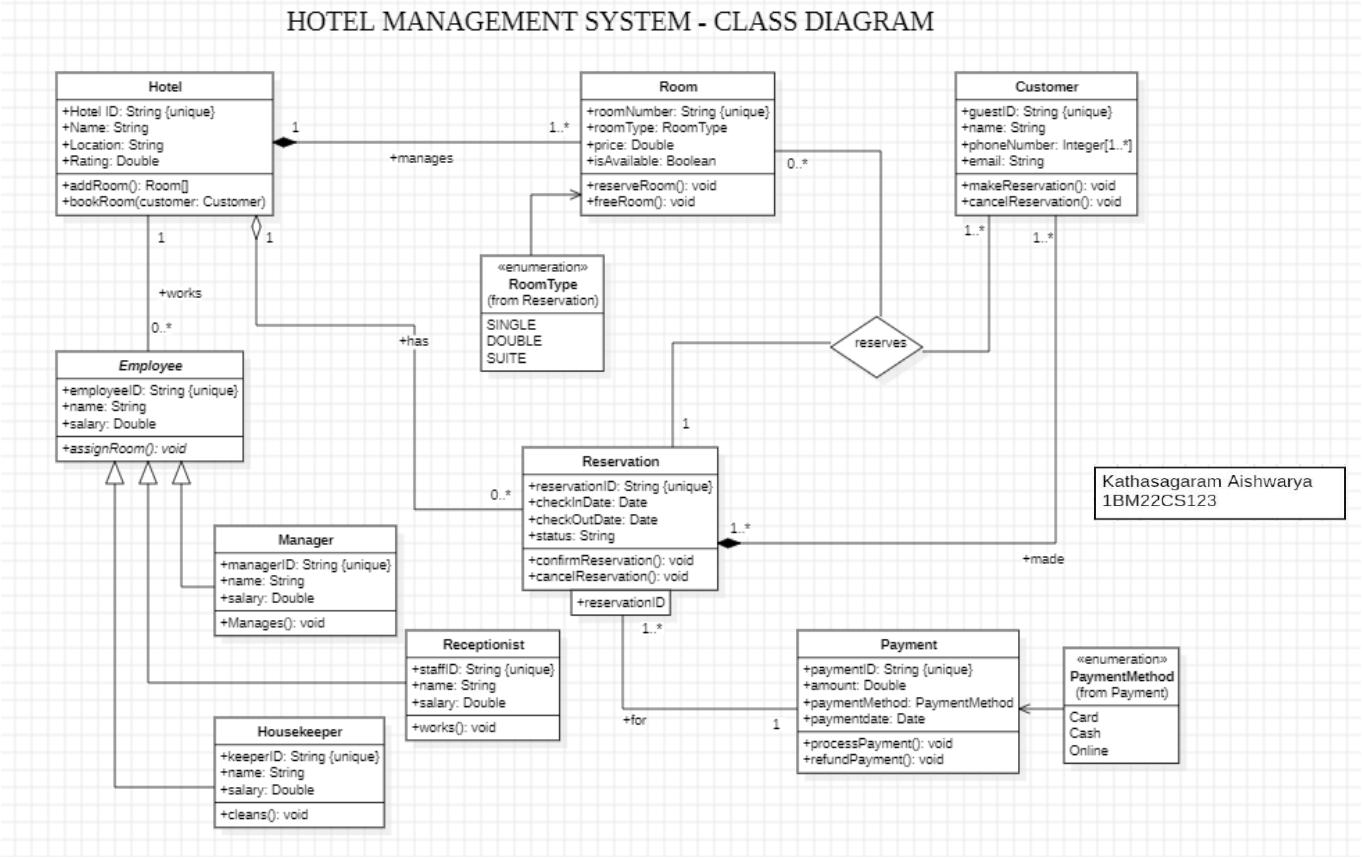



*Figure 1.1: Class Diagram*

The Hotel class represents a hotel with attributes such as Hotel ID, Name, Location, and Rating. It includes methods to manage rooms, such as adding new rooms and booking them for customers. The Room class defines individual rooms in the hotel, each identified by a room number, type, availability status, and price. Rooms can be reserved or marked as available using appropriate methods.

The Customer class holds customer details, including ID, name, phone number, and email. Customers can make or cancel room reservations. The Employee class serves as a base for different roles within the hotel, with attributes like Employee ID, Name, and Salary. It provides a method to assign rooms. Specialized roles include Manager, who oversees hotel operations, Receptionist, responsible for front desk tasks, and Housekeeper, tasked with cleaning rooms.

Reservations are managed by the Reservation class, which includes attributes such as Reservation ID, check-in and check-out dates, and status. It provides methods to confirm or cancel bookings. The Payment class handles payment transactions with attributes like Payment ID, amount, method, and date, along with methods to process or refund payments.

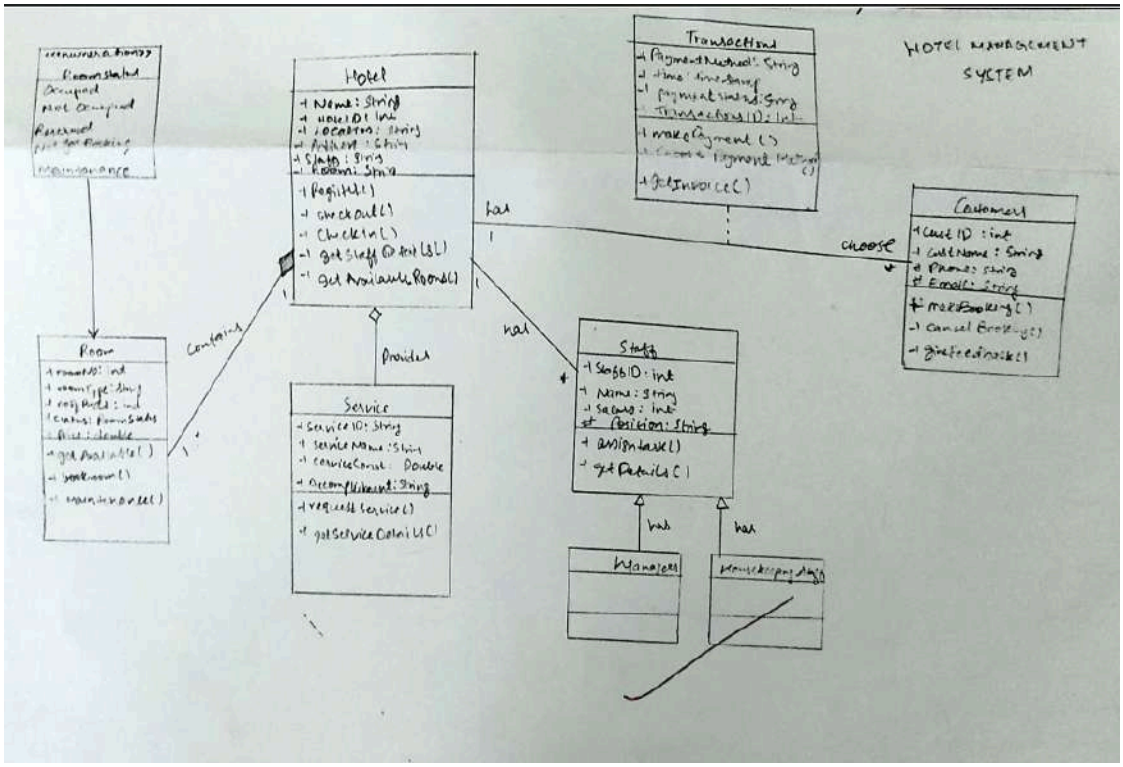
The system features several associations:

- A Hotel manages multiple rooms.
- A Customer can reserve multiple rooms, and a room can be reserved by many customers.
- Employees work for a hotel, with managers overseeing the hotel and specialized roles like receptionists and housekeepers.
- Each Reservation corresponds to one room, and each Payment is tied to one reservation.

Enumerations are used to categorize data:

- RoomType: Defines room categories such as SINGLE, DOUBLE, and SUITE.
- PaymentMethod: Specifies payment options, including Cash, Card, and Online.

This system ensures efficient management of hotels, rooms, staff, reservations, and payments.



## STATE DIAGRAM

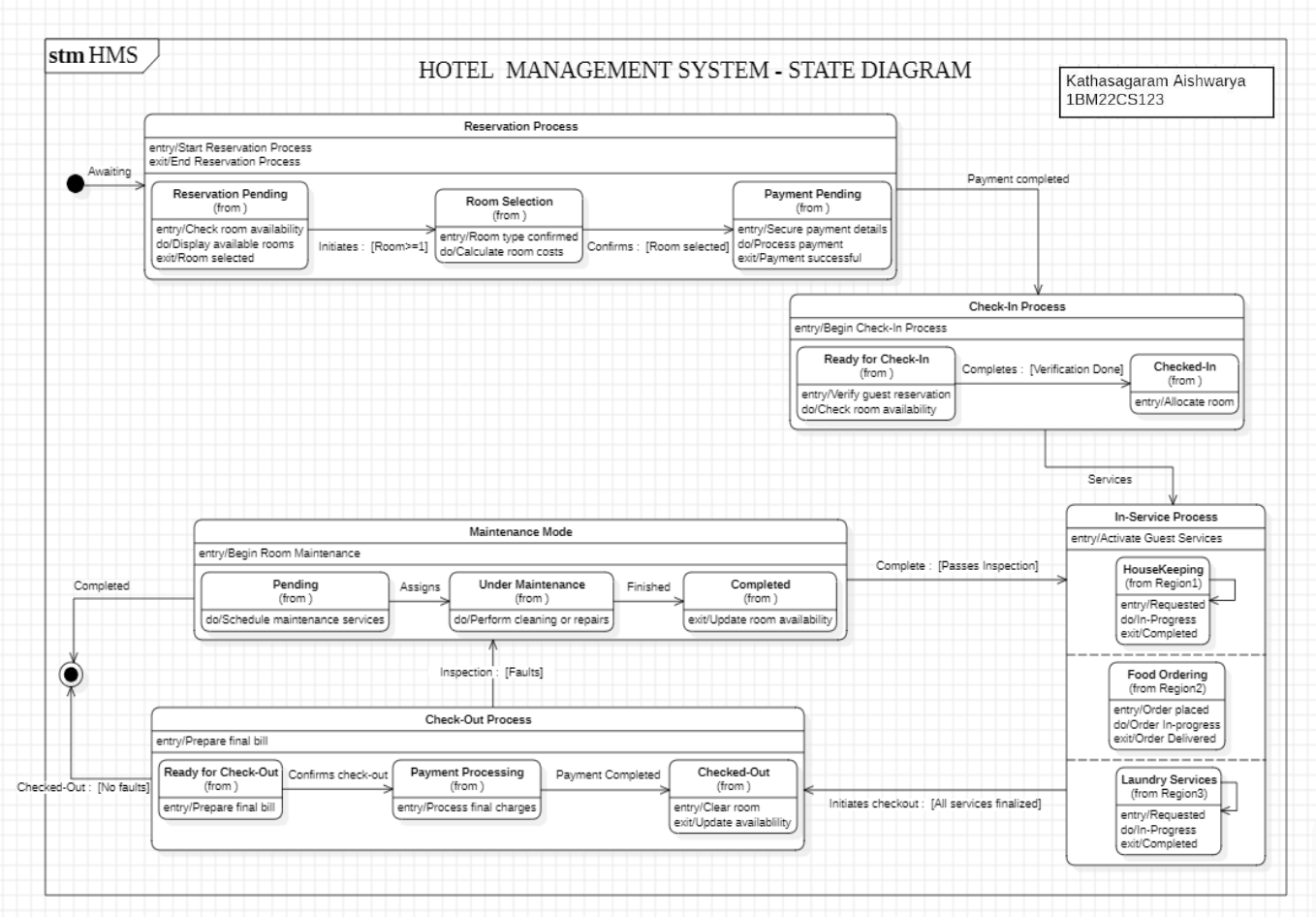
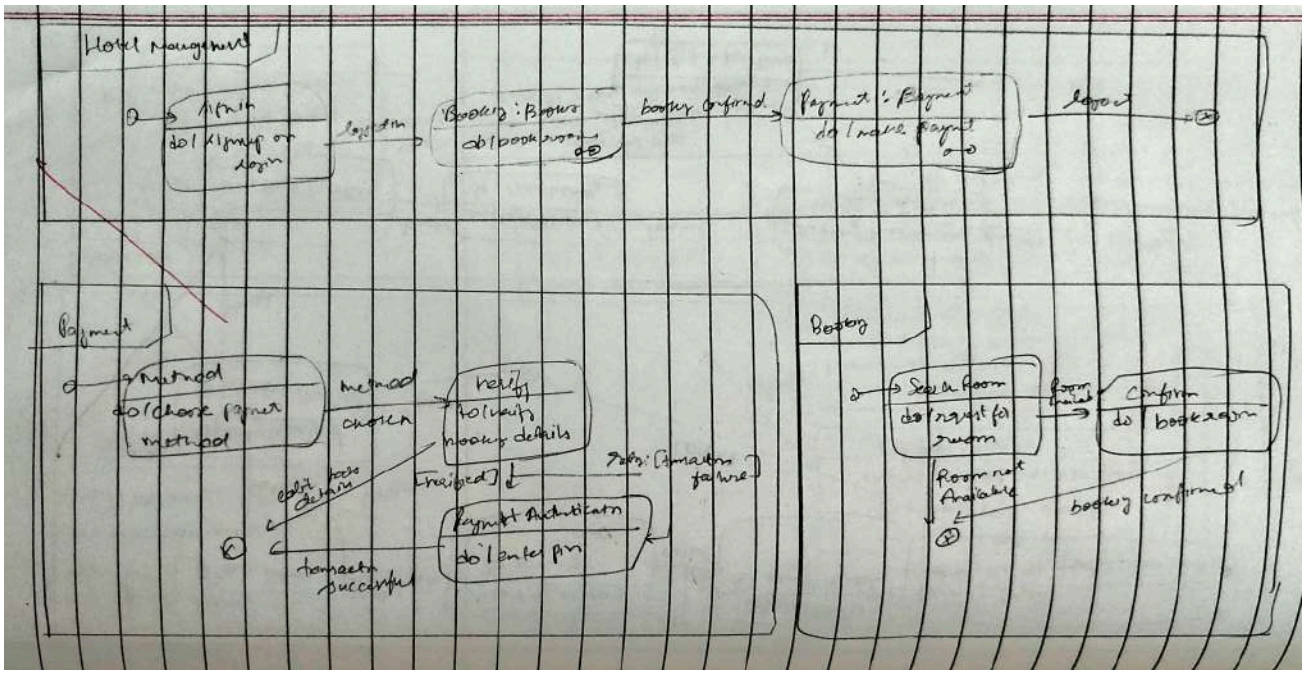



*Figure 1.2: State Diagram*

The reservation process begins in the Awaiting Reservation state, where the system is ready to accept new requests. Once a customer initiates a reservation, the state transitions to Reservation Pending, where room availability is checked. If rooms are available, the system moves to the Room Selection state, allowing the customer to choose their preferred room. After selection, the Payment Pending state prompts the customer to complete the payment. Successful payment transitions the process to Payment Completed, confirming the reservation. Finally, when the customer arrives at the hotel, the process moves to the Checked-In state, where the room is assigned.

For services provided during the stay, the In-Service Process includes nested states such as Housekeeping, Food Ordering, and Laundry Services. Each service follows a cycle starting with Requested, triggered by the guest's request. The service then moves to In-Progress as it is executed and concludes with Completed once finished. These nested states ensure smooth operations and enhance the guest experience.

This state overview highlights the key transitions from reservation to in-service processes, illustrating how the system manages guest interactions efficiently.



## USECASE DIAGRAM

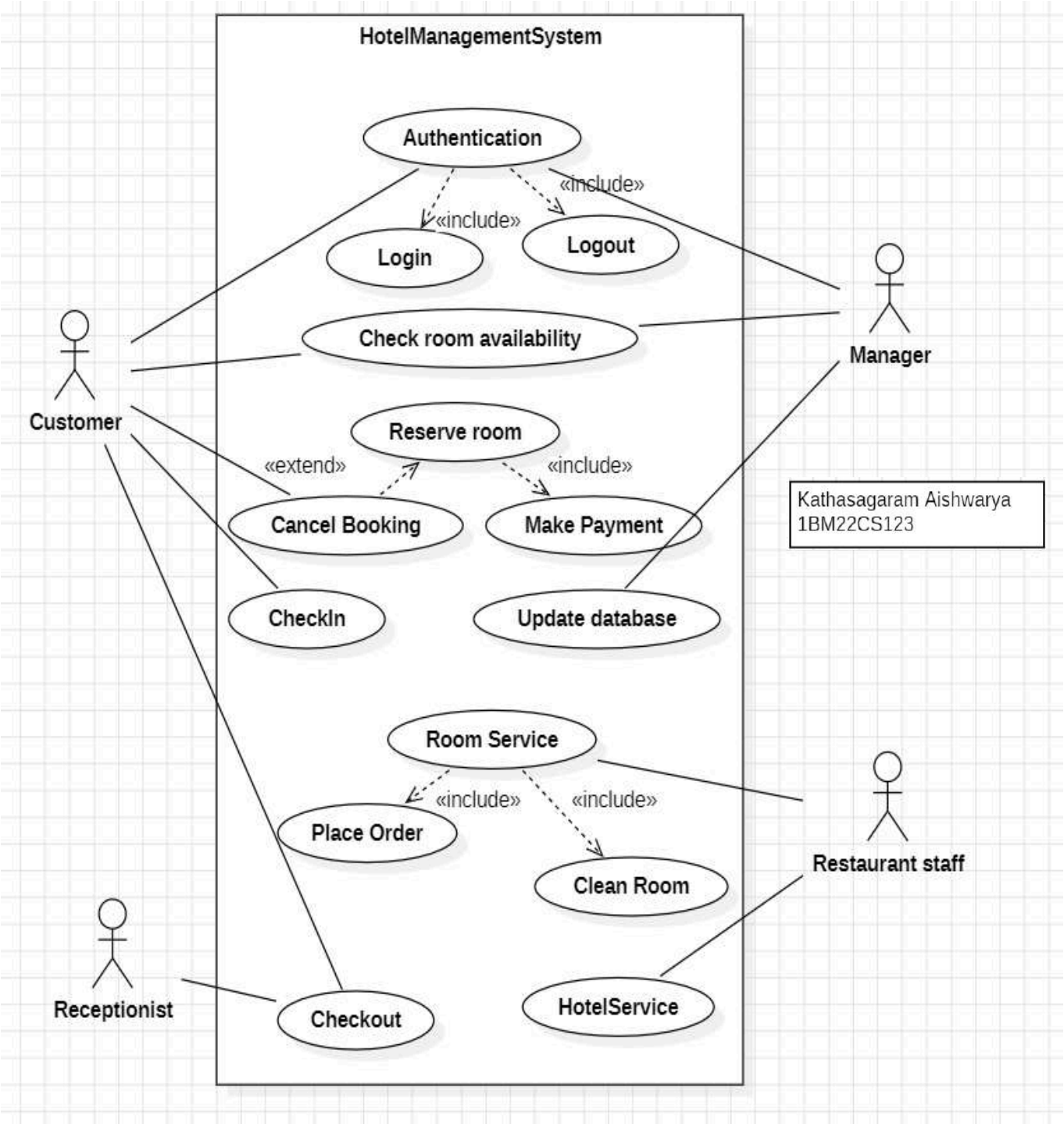
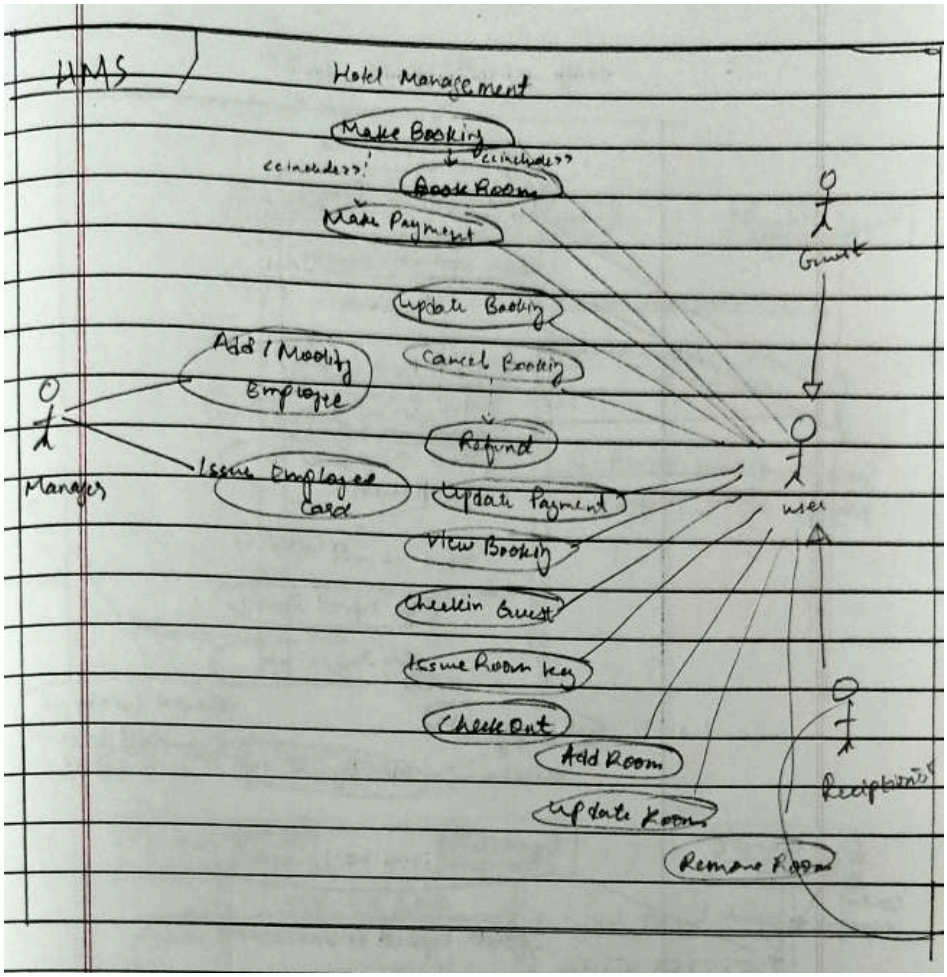



*Figure 1.3: Use Case Diagram*

The use case diagram illustrates the primary interactions within the Hotel Management System, focusing on customer and employee roles. Customers begin by logging into the

system, where they can check room availability, reserve rooms, and cancel bookings. Once a reservation is made, they proceed with payments to confirm their booking. Receptionists play a vital role in facilitating the check-in and check-out processes, ensuring a smooth customer experience.

For room management, the manager is responsible for updating the database to reflect room availability and other operational details. During a guest's stay, additional services like room cleaning, food orders, and general housekeeping are managed. Restaurant staff handle customer food orders, while housekeepers take care of room cleaning and maintenance as requested. This diagram highlights the streamlined interaction between customers, staff, and the system to ensure efficient hotel operations.



## SEQUENCE DIAGRAM

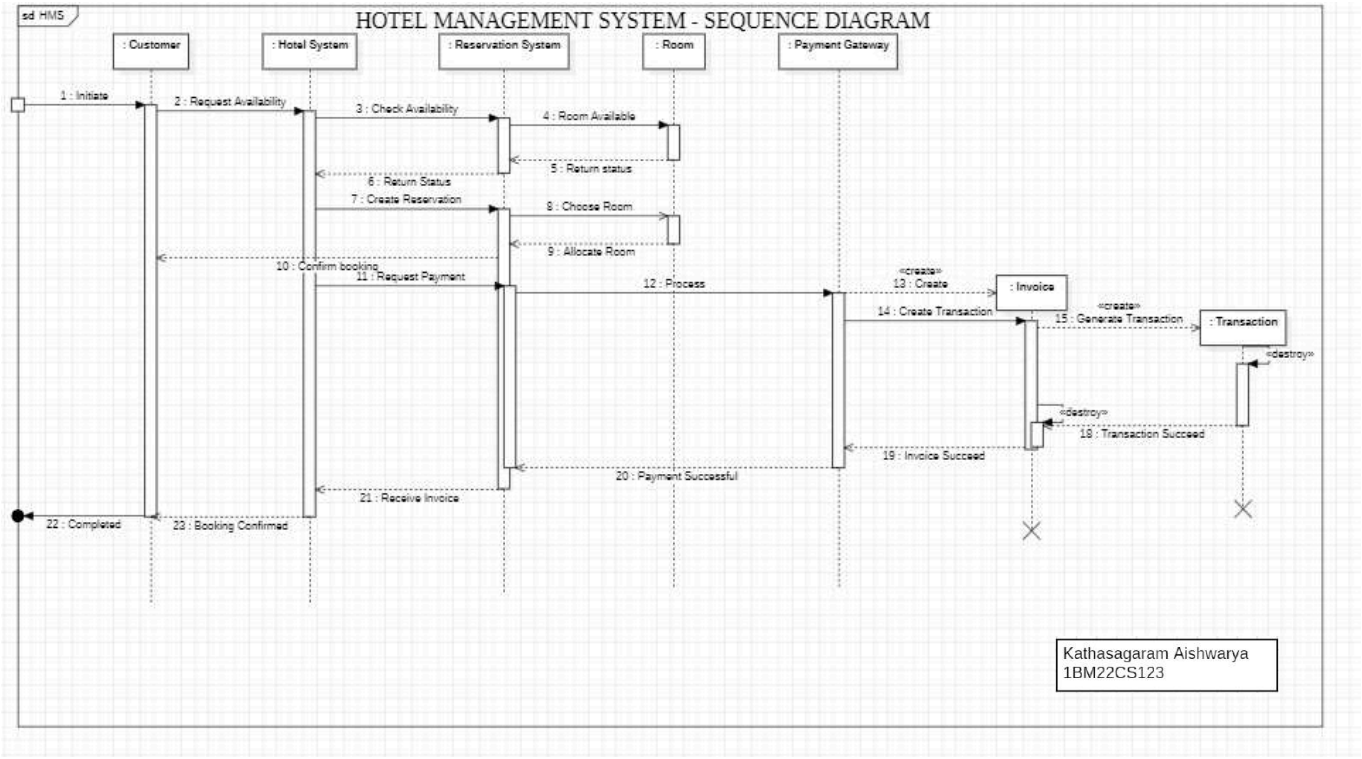



*Figure 1.4: Sequence Diagram*

Scenarios - Customer reservation and Cancellation:

In the Customer Makes a Reservation scenario, the process begins with the customer sending a request to check room availability for specific dates. The hotel system interacts with the reservation system to verify room availability and communicates the options back to the customer. Upon selecting a room, the customer confirms the booking, and the hotel system creates the reservation, allocates the room, and processes payment via a payment gateway. The customer receives confirmation, an invoice, and a transaction receipt, completing the booking process.

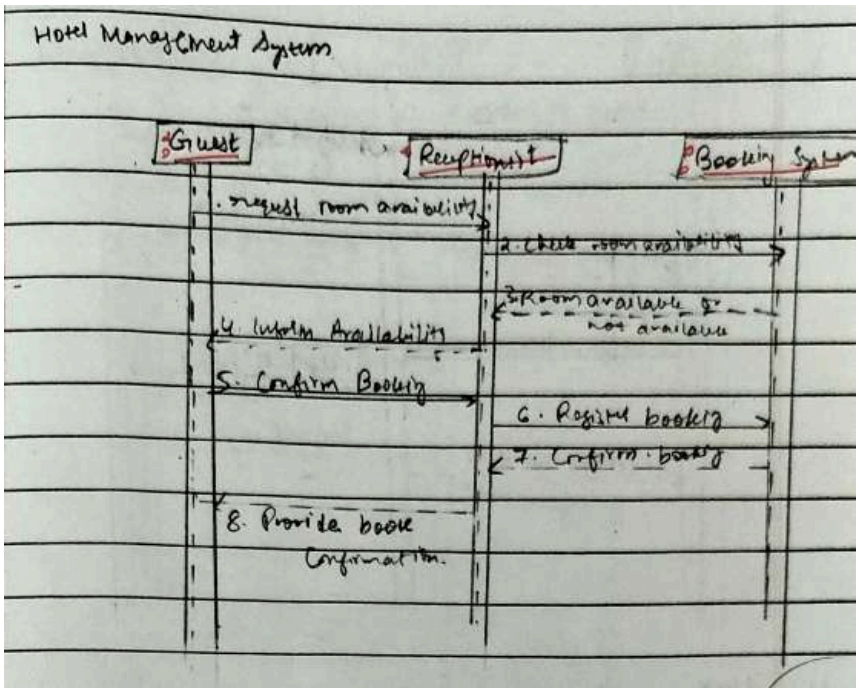
In the Customer Cancels a Reservation scenario, the customer sends a cancellation request. The hotel system verifies the reservation status, updates it to "Cancelled" in the reservation system, and notifies the customer. The room is released back into availability, and, if applicable, the hotel system initiates a refund.

Sequence Diagram:

The Simple Sequence Diagram outlines a user login process. A user submits credentials to the authentication system, which interacts with the database to validate the information. Upon successful authentication, the application displays the user interface for access.

The Advanced Sequence Diagram focuses on an order processing system. When a user places an order, the system creates transient objects like orders and order lines, which interact with passive objects like products and inventory to retrieve details and check availability. Once the order is processed, inventory is updated, and transient objects are destroyed, ensuring efficiency.

These diagrams capture critical workflows for reservations, cancellations, and system interactions, ensuring streamlined hotel operations and efficient order management.



## ACTIVITY DIAGRAM

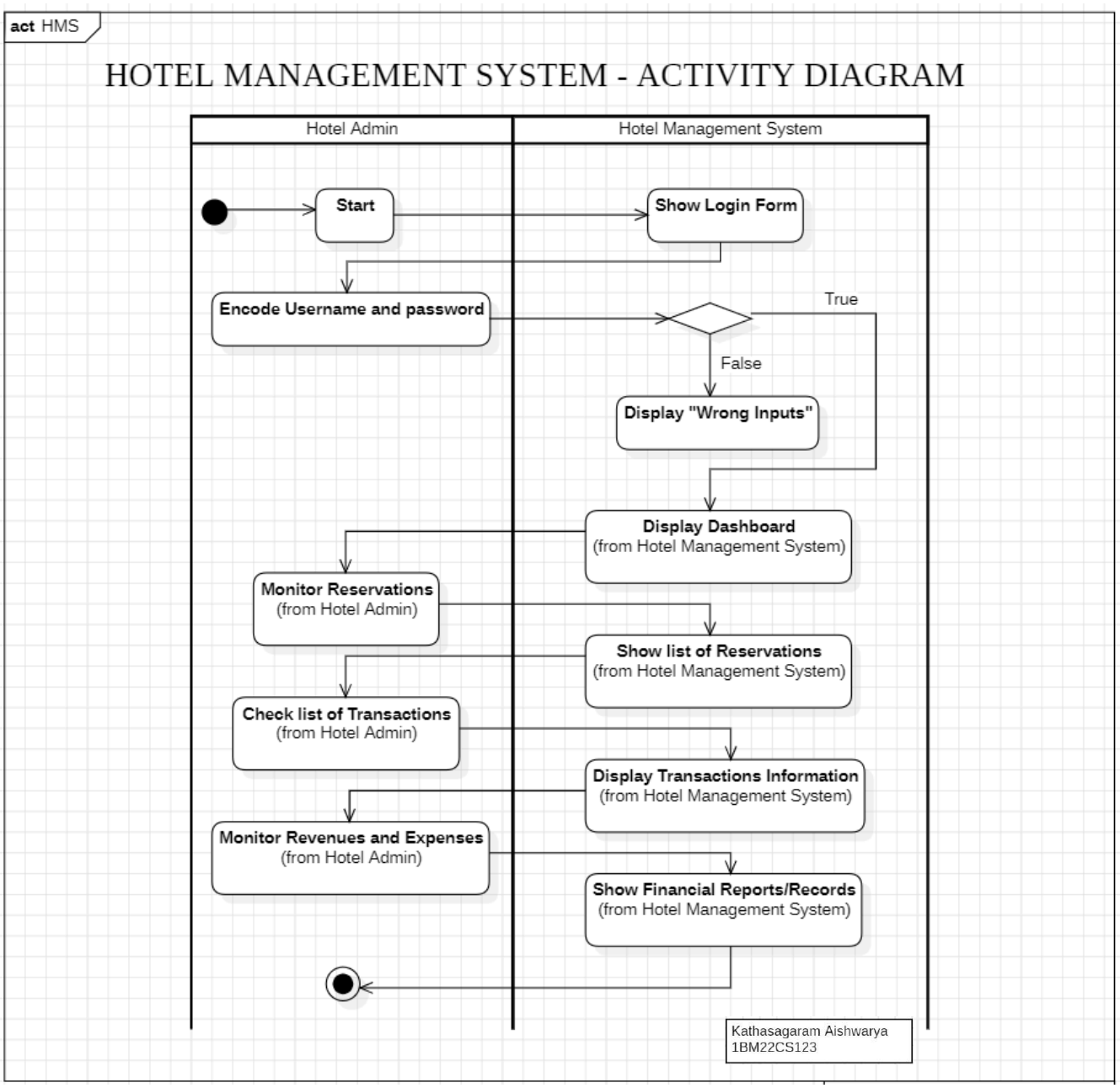



*Figure 1.5: Activity Diagram*

The swimlane diagram for the Hotel Management System divides actions into two key lanes:

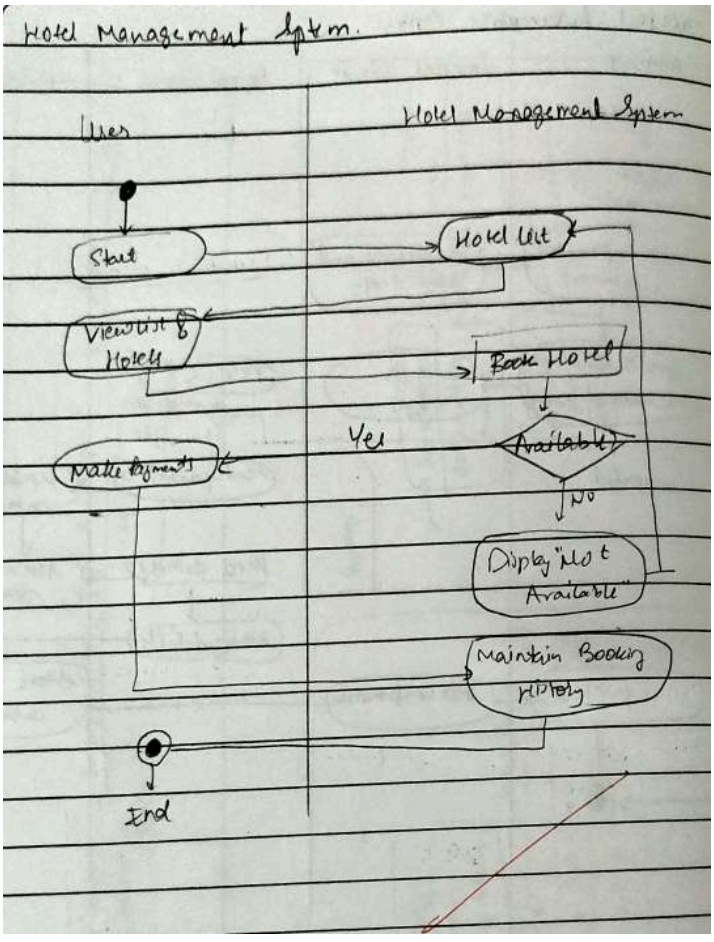
1. Hotel Admin:

   This lane focuses on tasks performed by the hotel administrator. It begins with the

admin logging into the system by entering their username and password. Once logged in, the admin monitors reservations and reviews transaction records to keep track of bookings and payments. Additionally, the admin analyzes financial performance by reviewing revenues and expenses, ensuring effective management of the hotel's financial activities.

2. Hotel Management System:

   This lane represents the system's automated responses to admin actions. The system begins by displaying the login form. If incorrect credentials are entered, it shows an error message. Upon successful login, the system displays the dashboard, which provides an overview of key information. It also facilitates the admin's tasks by showing lists of reservations, detailed transaction records, and financial reports, helping streamline hotel management operations.



# 2. CREDIT CARD PROCESSING

## 2.1 Problem Statement

This lane represents the system's automated responses to admin actions. The system The Credit Card Processing System is designed to securely handle transactions between customers, merchants, and financial institutions. It involves classes like CreditCard, Transaction, Customer, and Merchant, with attributes and methods to verify card details, authorize payments, and process refunds. The system should maintain relationships such as a customer owning multiple credit cards and a merchant handling multiple transactions. Security features, such as encryption and fraud detection, are key components, ensuring the system complies with financial regulations.

The objective of this project is to design and implement a robust Credit Card Processing System to ensure secure, efficient, and accurate handling of credit card transactions. The proposed system aims to address the following critical functionalities:

1. Transaction Processing: Facilitate real-time authorization, capture, and settlement of credit card payments, ensuring fast and reliable transaction handling.

2. Fraud Detection and Prevention: Integrate advanced algorithms to monitor transactions for suspicious activities, implement security protocols, and reduce fraudulent activities effectively.

3. Account Management: Provide tools for managing user accounts, including updating cardholder information, viewing transaction histories, and generating account statements.

4. Dispute Resolution: Support efficient handling of disputes and chargebacks by enabling detailed tracking of transaction records and providing automated workflows for resolution.

5. Compliance and Security: Ensure adherence to industry standards such as PCI DSS (Payment Card Industry Data Security Standard) and implement strong encryption and tokenization mechanisms to safeguard sensitive data.

6. Integration Capabilities: Allow seamless integration with banking systems, payment gateways, and merchant platforms to enhance interoperability and scalability.

This system will centralize and automate credit card processing, minimizing errors, enhancing transaction security, and providing a seamless user experience for both merchants and cardholders.

## 2.2 Software Requirements Specification (SRS)

1.Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for a Credit Card Processing System (CCPS). The system is designed to facilitate secure, reliable, and efficient processing of credit card transactions for businesses and their customers. This document provides a comprehensive outline of the functional and non-functional requirements, design constraints, and system interfaces.

1.2 Scope

The CCPS is a centralized platform that enables businesses to process credit card payments securely and efficiently. The system supports functionalities such as transaction authorization, settlement, fraud detection, and reporting. It integrates with third-party payment gateways and complies with industry security standards such as PCI-DSS.

### 1.3 Definitions, Acronyms, and Abbreviations

- CCPS: Credit Card Processing System
- PCI-DSS: Payment Card Industry Data Security Standard
- API: Application Programming Interface
- GUI: Graphical User Interface

### 1.4 References

1) IEEE Standard for Software Requirements Specifications (IEEE Std 830-1998)
2) PCI-DSS Compliance Guidelines

### 1.5 Overview

This document is organized into sections that describe the overall system functionality, user characteristics, and specific requirements.

## 2. Overall Description

### 2.1 Product Perspective

The CCPS is a modular application designed to handle end-to-end credit card transaction processing. It integrates with merchant systems, card networks, and banking infrastructure to ensure seamless payment workflows.

### 2.2 Product Features

Key features of the CCPS include:

- Real-time transaction authorization and processing
- Fraud detection and prevention
- Automated settlement and reconciliation
- Reporting and analytics dashboards
- Integration with third-party payment gateways

### 2.3 User Characteristics

The system is designed for the following user groups:

- Merchants: Use the system to process customer payments and view transaction reports.
- Financial Institutions: Handle transaction settlement and provide fraud detection insights.
- System Administrators: Manage and maintain system operations.
- Customers: Use the system indirectly through merchant transactions.

2.4 Constraints

- The system must comply with PCI-DSS and other financial regulations.
- It should ensure compatibility with existing merchant and bank hardware/software systems.
- High availability and fault tolerance are required to handle critical payment operations.

2.5 Assumptions and Dependencies

- A reliable internet connection is assumed for real-time processing.
- The system depends on third-party gateways and bank APIs for transaction completion.

3. Specific Requirements

3.1 Functional Requirements

- Transaction Authorization:
  - Validate credit card details and approve transactions in real time.
  - Notify merchants and customers of transaction status.
- Fraud Detection:
  - Identify suspicious activity using rule-based and AI-powered algorithms.
  - Generate alerts for flagged transactions.
- Settlement and Reconciliation:
  - Automate the process of transferring funds between parties.
  - Provide detailed transaction reports for auditing.
- Reporting:

Offer dashboards for transaction volume, revenue trends, and fraud analysis.

Support export of reports in standard formats (e.g., PDF, CSV).

3.2 Performance Requirements

- The system should support up to 1,000 concurrent transactions.
- Response time for transaction authorization must not exceed 3 seconds.
- Data backups must be performed every 15 minutes to minimize data loss.

3.3 Interface Requirements

- GUI:
  - User-friendly design for merchants and administrators.
  - Role-based dashboards with customizable views.
- API Integration:

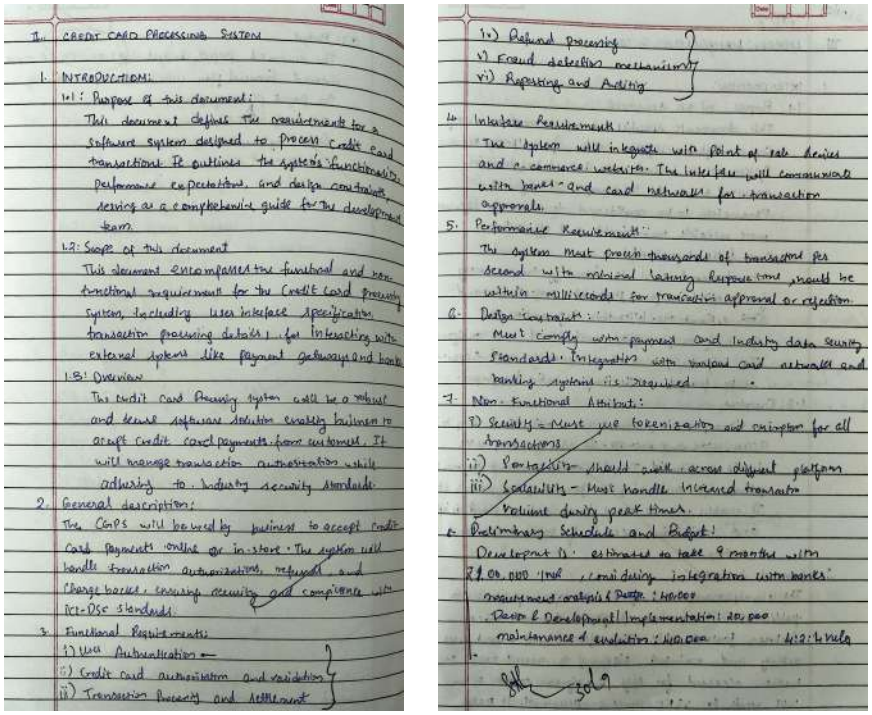
  RESTful APIs for integration with third-party gateways and banking systems.

3.4 Design Constraints

- The system must be developed using secure, scalable technologies.
- Hardware requirements should be minimal to ensure widespread adoption.

3.5 Non-Functional Attributes

- Reliability: 99.99% uptime to support critical financial operations.
- Security: Adherence to PCI-DSS standards, including encryption and tokenization.
- Usability: Simple and intuitive user interfaces for all stakeholders.
- Scalability: Capability to handle increasing transaction volumes.
- Maintainability: Modular architecture for ease of updates and debugging.



# CLASS DIAGRAM

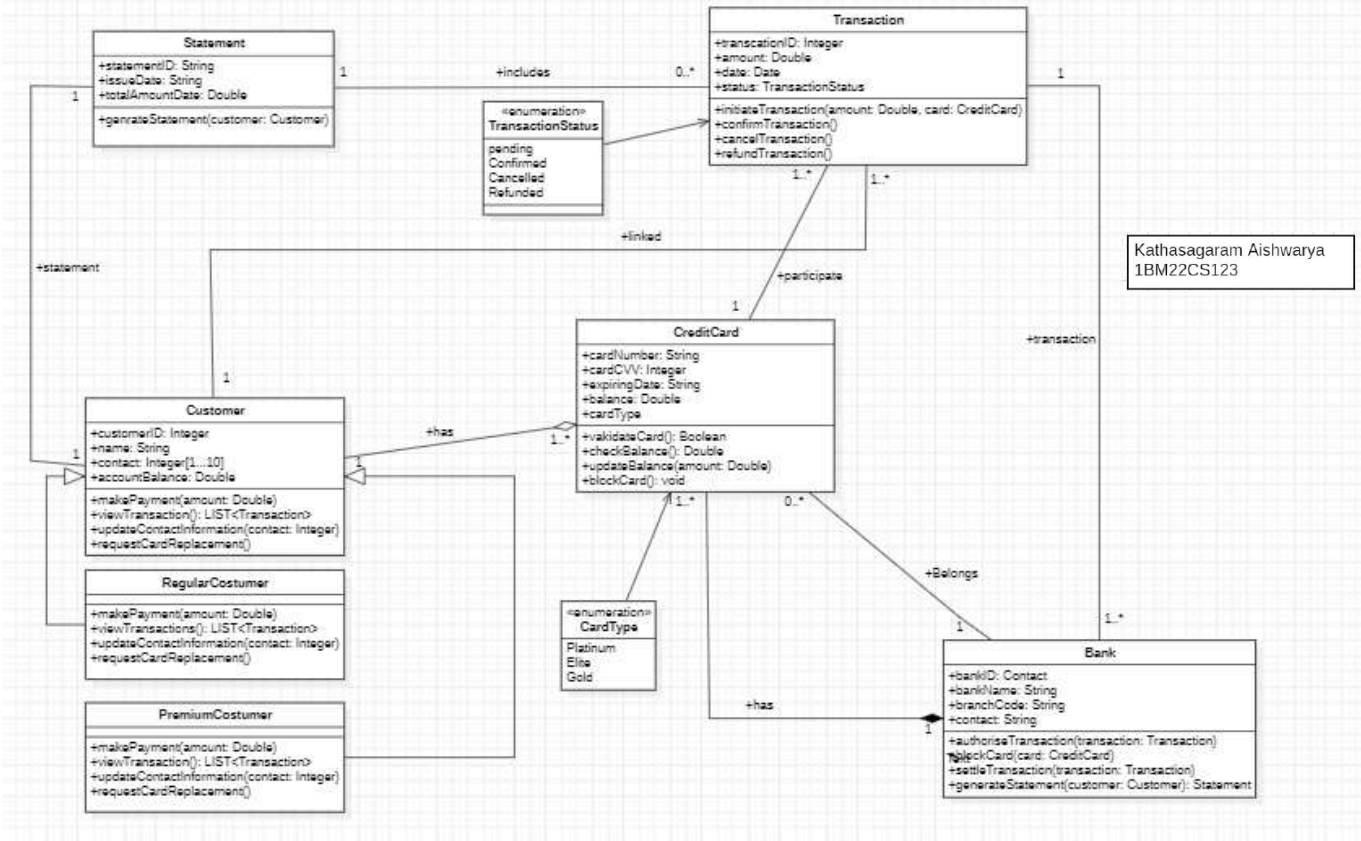
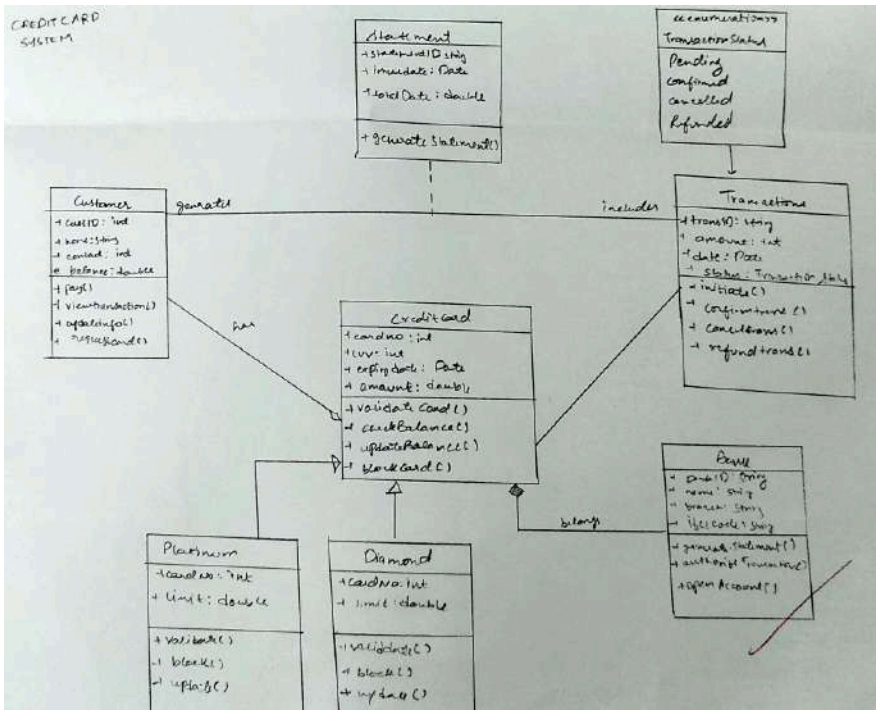



*Figure 2.1: Class Diagram*

The system models the interaction between customers, credit cards, transactions, and banks. At its core are several classes, each with distinct roles. The Statement class tracks the details of a customer's financial statement, including the total amount due and the transactions it contains. The Transaction class represents individual transactions with attributes such as transaction ID, date, amount, and status. It is closely linked to the CreditCard class, which manages information about credit cards, such as card number, type, expiration date, and balance. The Customer class represents an individual customer, with the ability to make payments, view transactions, and manage account details. Regular and premium customers are distinguished by the RegularCustomer and PremiumCustomer subclasses, with premium customers often having additional benefits.

The Bank class plays a crucial role in managing the overall financial transactions, including authorizing, settling, and blocking transactions. Banks also generate statements for customers and issue credit cards. Relationships between these classes define how they interact; a statement can include multiple transactions, and a transaction is linked to one credit card. Customers can hold multiple credit cards, while banks can issue many credit cards and participate in many transactions. Two key enumerations, TransactionStatus and CardType, help define the states of transactions (such as Pending, Confirmed, or Refunded) and categorize cards (such as Platinum, Elite, or Regular).

In summary, the system integrates the management of financial statements, transactions, credit cards, and customer accounts, ensuring smooth interaction between customers, their banks, and the transactions they conduct.




## STATE DIAGRAM

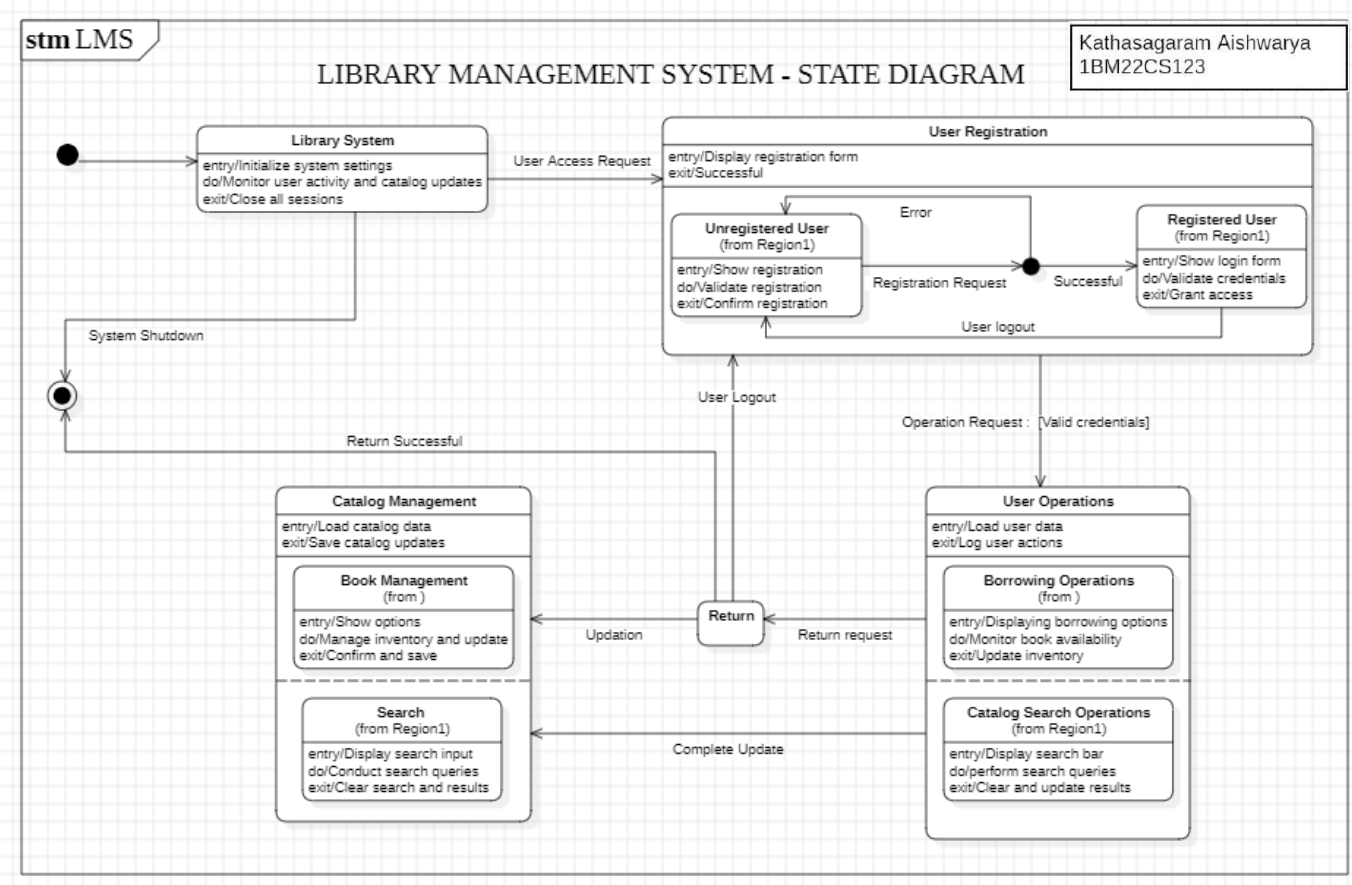



*Figure 2.2: State Diagram*

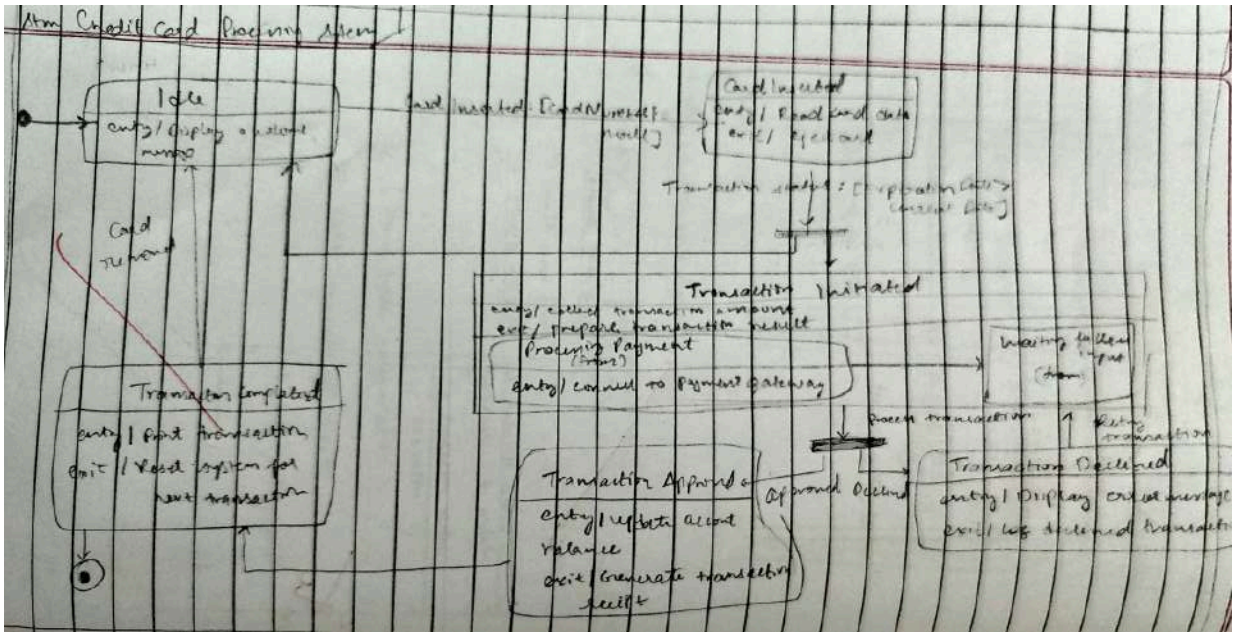
This state diagram represents the flow of a credit card processing system, outlining the sequence of states, transitions, events, and activities involved in handling a transaction. The system begins in the Idle state, where it waits for a card to be inserted. When the card is inserted, the system transitions to the Card Inserted state, where it reads the card data.

Next, the Transaction Initiated state is triggered when the user enters the transaction amount. At this point, the system starts to communicate with the payment gateway, transitioning into the Processing Payment state. If additional input is required, such as PIN entry or approval, the system moves into the Waiting for User Input state.

After processing, the system waits for a response from the payment gateway. If the transaction is successful, the system transitions to the Transaction Approved state, updating the account balance accordingly. If the transaction is declined due to insufficient funds or other reasons, it enters the Transaction Declined state, displaying an error message.

In either case, once the transaction status is determined, the system moves to the Transaction Completed state. This state signifies the end of the transaction process, and the system is reset, ready for the next transaction.

The state transitions are triggered by specific events such as Card Inserted, Transaction Starts, and Response from Gateway. Activities, like displaying a welcome message or printing a transaction summary, are performed at different stages, enhancing the user experience and ensuring smooth operation throughout the process.




## USECASE DIAGRAM

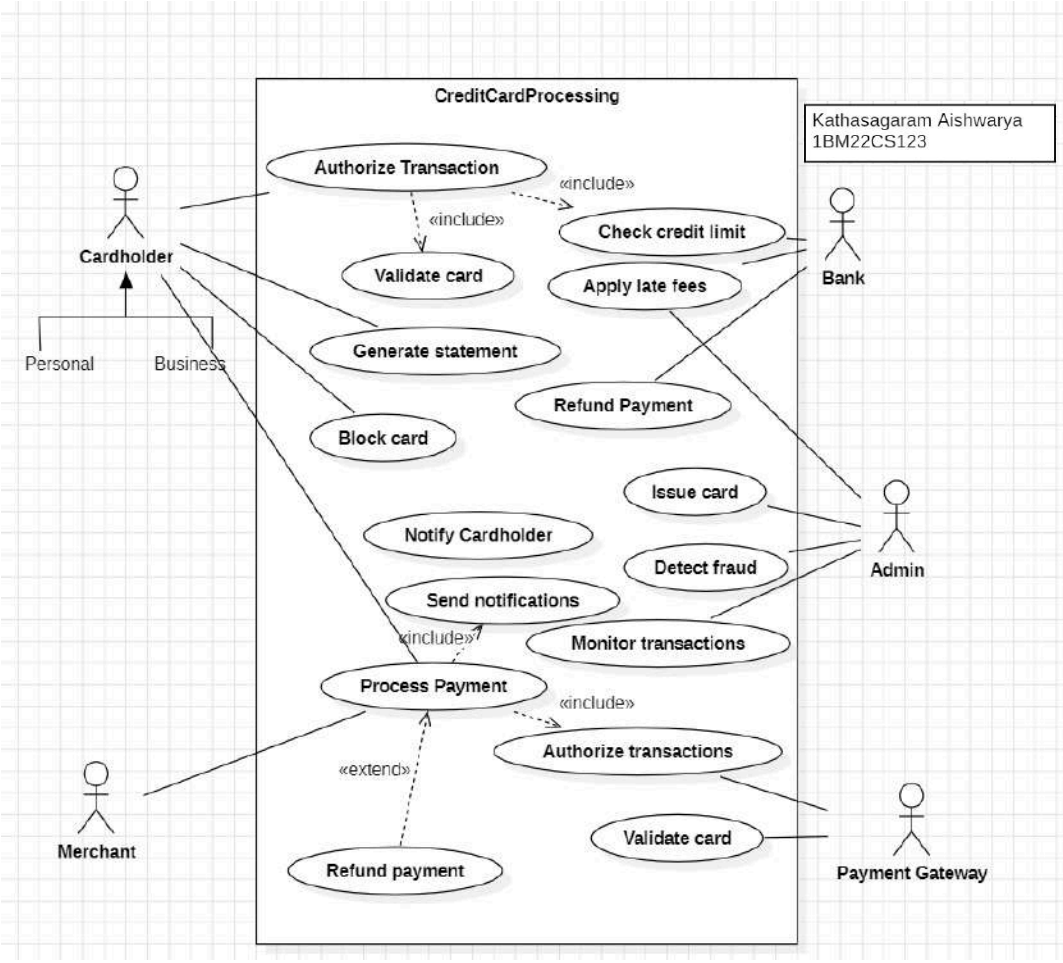
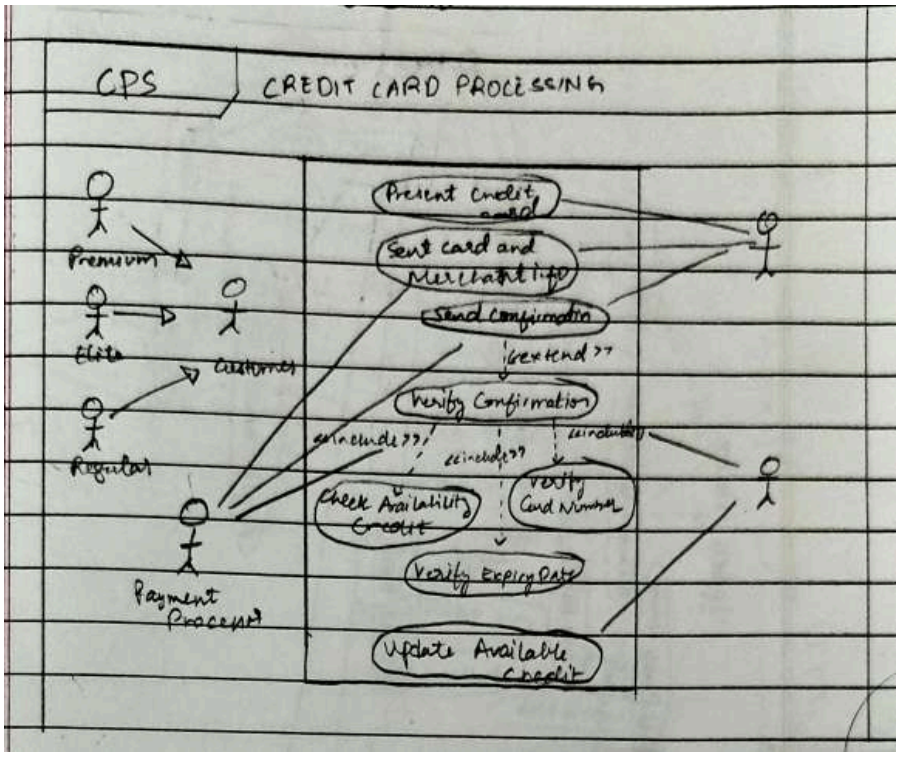



*Figure 2.3: Use Case Diagram*

The credit card processing system involves several key actors: the Customer, the Merchant, the Payment Processor, and the Bank. The Customer is the cardholder who initiates the transaction, which can be a Regular, Elite, or Premiumcardholder. Regular customers have standard privileges, while Elite and Premium customers enjoy higher spending limits and additional benefits. The Merchant is the business receiving the payment from the customer and interacts with the Payment Processor to process the transaction.

The transaction begins when the Customer presents their credit card to the Merchant, who then sends the card details and transaction information to the Payment Processor. The Payment Processor verifies the card number and expiry date, checks the customer's available credit, and confirms whether the transaction can proceed. If approved, the processor updates the customer's available credit and sends a confirmation back to the merchant. If the transaction is declined, the customer is notified of the failure. The Bank is responsible for issuing the credit card and managing the customer's account, ensuring funds are available for the transaction.

This system ensures that each transaction is processed efficiently, updating the customer's account balance in real-time while maintaining secure and accurate payment processing.




## SEQUENCE DIAGRAM

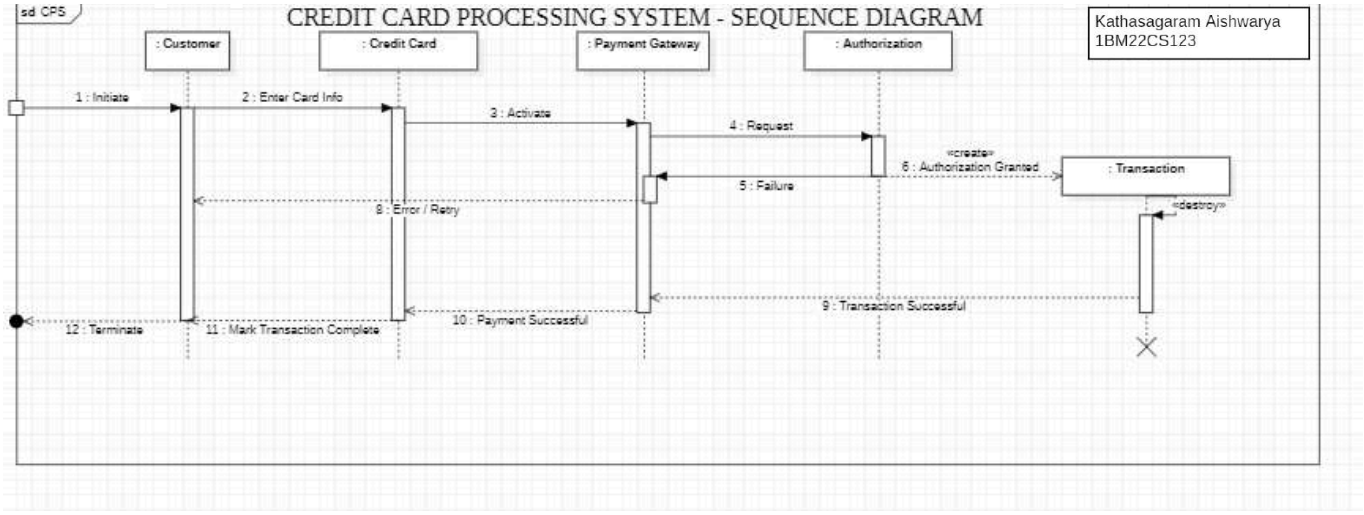


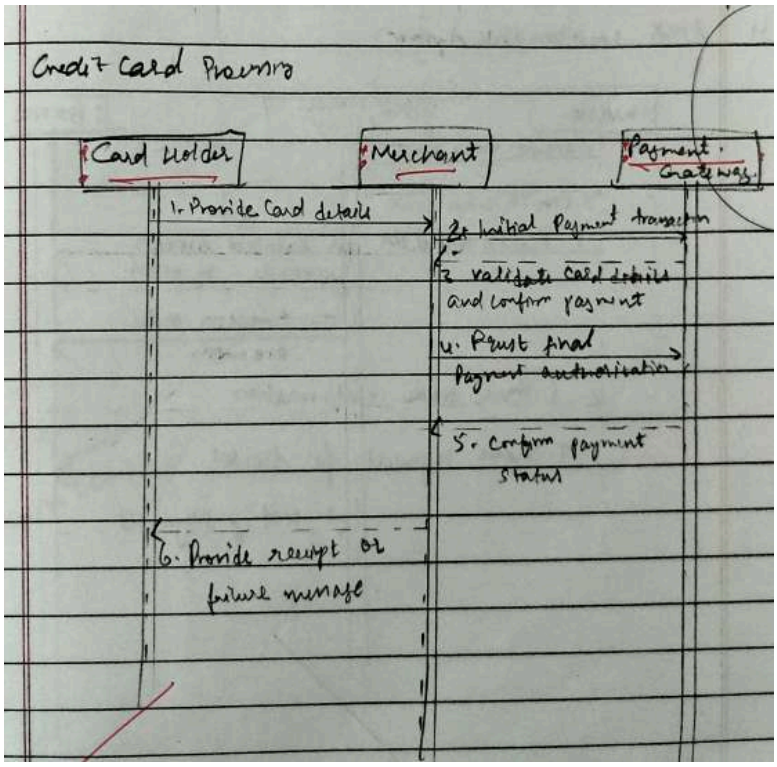

*Figure 2.4: Sequence Diagram*

This Sequence Diagram outlines the interactions between key objects in a credit card transaction. The process begins when the Customer initiates the transaction. The Customer then enters the credit card information, such as the card number, expiry date, and CVV. Once the card details are entered, the Credit Card is activated for use in the transaction.

The next step involves the Credit Card sending a request for authorization to the Payment Gateway. If there is an issue, such as invalid card details or insufficient funds, the Payment Gateway sends a failure response back to the Credit Card, and the process may prompt the Customer to retry by entering the information again. If the authorization is successful, the Payment Gateway grants authorization to the Credit Card.

Once authorization is granted, a Transaction object is created to record the successful transaction. The Payment Gateway then informs the Credit Card that the transaction was successful, and the Credit Card notifies the Customerof the successful payment. The Credit Card then marks the transaction as complete, and the process is terminated.

This sequence diagram illustrates the flow of messages between the entities, showing how each step leads to the next, and highlights the possibility of failure and retry in the event of an error. It captures the stages of transaction processing, from initiation to completion, ensuring clarity in the system's behavior and interactions.



## ACTIVITY DIAGRAM

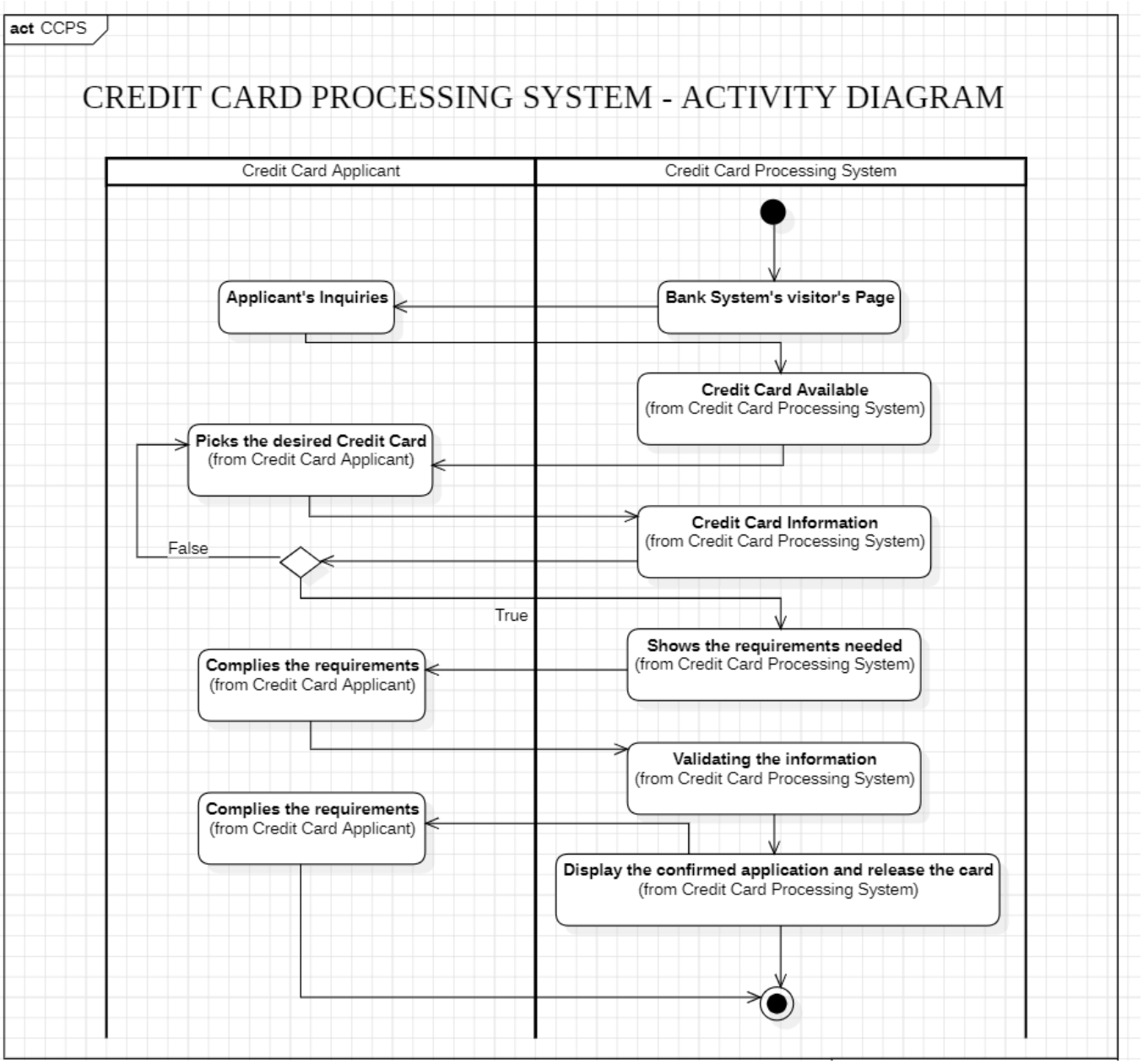


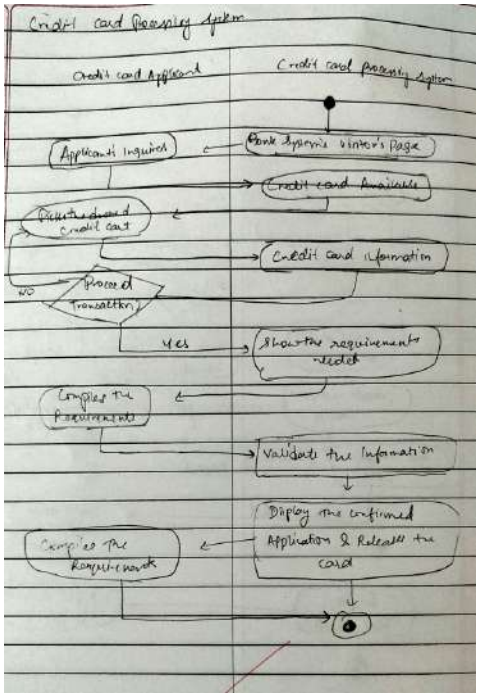

*Figure 2.5: Activity Diagram*

The Swimlane diagram for the credit card application process illustrates the roles and actions of both the Credit Card Applicant and the Credit Card Processing System. The process begins with the Credit Card Applicant who inquires about the available credit card options. Once the applicant has reviewed the options, they select the desired credit card based on their preferences. The applicant then proceeds to comply with the requirements by

submitting necessary documents and information to support the application.

On the other hand, the Credit Card Processing System plays a key role in facilitating the application process. It starts by presenting a page showing all available credit card options to the applicant. Once the applicant selects a card, the system displays detailed information about the chosen credit card, including its features and benefits. The system then presents the requirements that the applicant must meet to be eligible for the card. After the applicant submits the necessary information, the system validates the details to ensure they meet the criteria for approval. If the validation is successful, the system confirms the application and initiates the card issuance process.

This Swimlane diagram clearly distinguishes the tasks performed by the Credit Card Applicant and the Credit Card Processing System, helping to visualize the flow of actions from initial inquiry to the final approval and release of the credit card.



# 3. LIBRARY MANAGEMENT SYSTEM

## 3.1 Problem Statement

The Library Management System aims to automate the management of library resources, including books, journals, and digital media. Classes like Member, Book, Loan, and Catalog are central to the system. The system should allow members to borrow and return items, manage overdue fines, and search the catalog by title, author, or category. Relationships include members borrowing multiple books and books being reserved or loaned to multiple members over time. The system must also maintain a history of transactions for audit and reporting purposes.

The proposed system aims to address the following critical functionalities:

1. Book Management: Facilitate the addition, removal, and categorization of books, along with real-time tracking of book availability to ensure efficient inventory management.

2. User Management: Enable registration and management of users, including librarians, members, and administrators, with appropriate access permissions and roles.

3. Borrowing and Return Management: Provide seamless processes for issuing and returning books, including automated due date tracking, fine calculation, and reminders for overdue books.

4. Catalog Search: Implement advanced search functionalities, such as keyword-based, genre-based, and author-based search, to help users locate books quickly.

5. Reservation System: Allow users to reserve books currently unavailable, with automated notifications when the reserved items become available.

6. Reports and Analytics: Generate detailed reports on library usage, borrowing trends, and book inventory to aid in decision-making and resource allocation.

7. Security and Data Integrity: Ensure secure access to the system, maintain data accuracy, and implement backup mechanisms to protect against data loss.

This system will integrate these functionalities into a cohesive platform, reducing manual effort, improving operational efficiency, and enhancing the overall user experience for library members and staff.

## 3.2 Software Requirements Specification (SRS)

1.Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for a Library Management System (LMS). The system aims to automate and streamline library operations, including book cataloging, borrowing, returns, and user management. This document provides a comprehensive outline of functional and non-functional requirements, design constraints, and system interfaces.

1.2 Scope

The LMS is a centralized platform designed to manage library resources efficiently. It caters to the needs of librarians, students, and faculty by offering features such as book search, reservation, inventory management, and overdue notifications. The system is adaptable for libraries of varying sizes and integrates with third-party digital libraries and catalog systems.

1.3 Definitions, Acronyms, and Abbreviations

- LMS: Library Management System
- API: Application Programming Interface
- GUI: Graphical User Interface

1.4 References

- IEEE Standard for Software Requirements Specifications (IEEE Std 830-1998)
- Library of Congress Cataloging Guidelines

1.5 Overview

This document is structured to detail the overall system description and specific requirements. Section 2 outlines the product features and user characteristics, while Section 3 elaborates on the functional and non-functional requirements.

## 2. Overall Description

### 2.1 Product Perspective

The LMS is a web-based application designed to manage library operations efficiently. It integrates with existing library systems, third-party catalog services, and digital libraries to offer seamless management and user experiences.

### 2.2 Product Features

Key features of the LMS include:

- Real-time book availability tracking
- User account and borrowing history management
- Automated notifications for due dates and overdue items
- Reporting and analytics dashboards
- Integration with digital libraries and e-book platforms

### 2.3 User Characteristics

The system is designed for the following user groups:

- Librarians: Manage book inventories, user accounts, and borrowing records.
- Students and Faculty: Search for, borrow, and reserve books.
- Administrators: Oversee overall library operations and generate reports.

### 2.4 Constraints

- The system must comply with data privacy regulations.
- High availability is required to ensure continuous access.
- Must support multilingual interfaces for diverse user bases.

### 2.5 Assumptions and Dependencies

- Reliable internet connectivity is assumed for accessing online resources.
- The system depends on third-party APIs for digital library integration.

## 3. Specific Requirements

3.1 Functional Requirements

- Book Management:
  - Add, update, and remove book entries in the catalog.
  - Track book availability in real time.
- User Management:
  - Allow users to create accounts and view borrowing history.
  - Manage user roles (e.g., librarian, student, faculty).
- Borrowing and Returning:
  - Enable users to borrow and return books seamlessly.
  - Send automated notifications for due dates and overdue items.
- Search and Reservation:
  - Provide advanced search capabilities for books by title, author, and genre.
  - Allow users to reserve unavailable books.
- Reporting:
  - Generate reports on inventory, borrowing trends, and overdue items.

3.2 Performance Requirements

- The system should support up to 1,000 simultaneous users.
- Response time for book search queries must not exceed 2 seconds.
- Daily data backups must be automated and verified.

3.3 Interface Requirements

- GUI:
  - Intuitive design for librarians and users.
  - Role-specific dashboards with relevant functionalities.
- API Integration:
  - RESTful APIs for third-party catalog and digital library services.

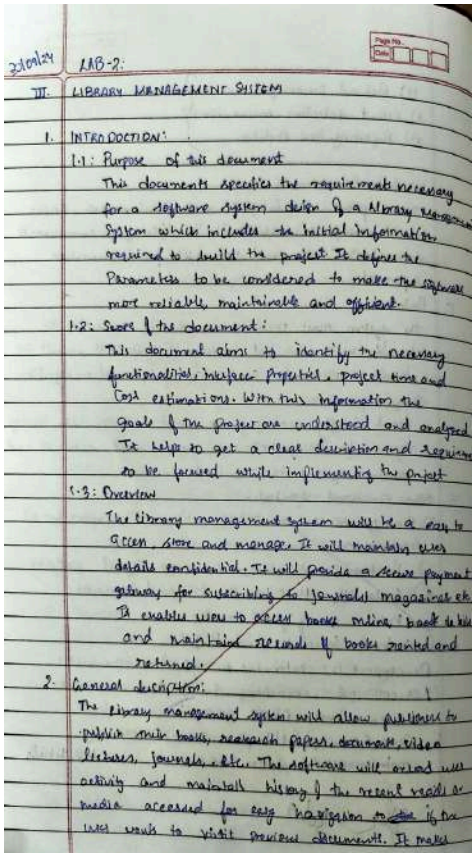
3.4 Design Constraints

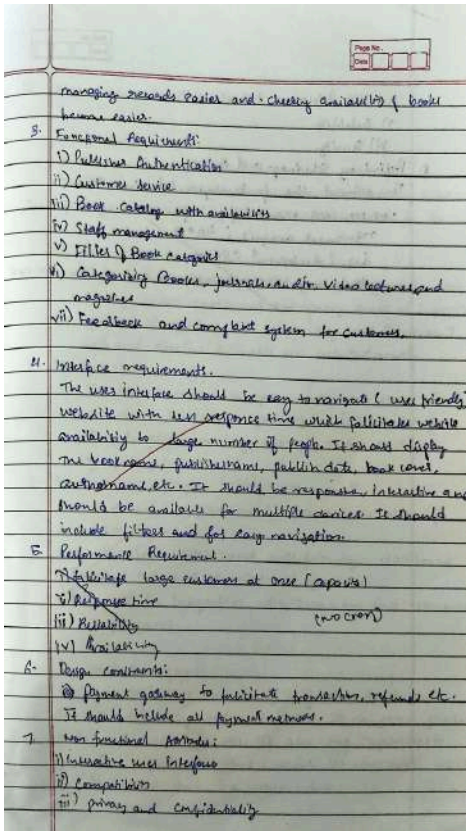
- The system must be developed using open-source technologies to reduce costs.

- Compatibility with standard desktop and mobile browsers is required.

3.5 Non-Functional Attributes

- Reliability: 99.9% uptime to ensure continuous library operations.
- Security: Ensure secure user authentication and data protection.
- Usability: Provide an accessible interface for users of varying technical expertise.
- Scalability: Handle increasing numbers of books and users efficiently.
- Maintainability: Modular design for ease of updates and feature enhancements.



## CLASS DIAGRAM

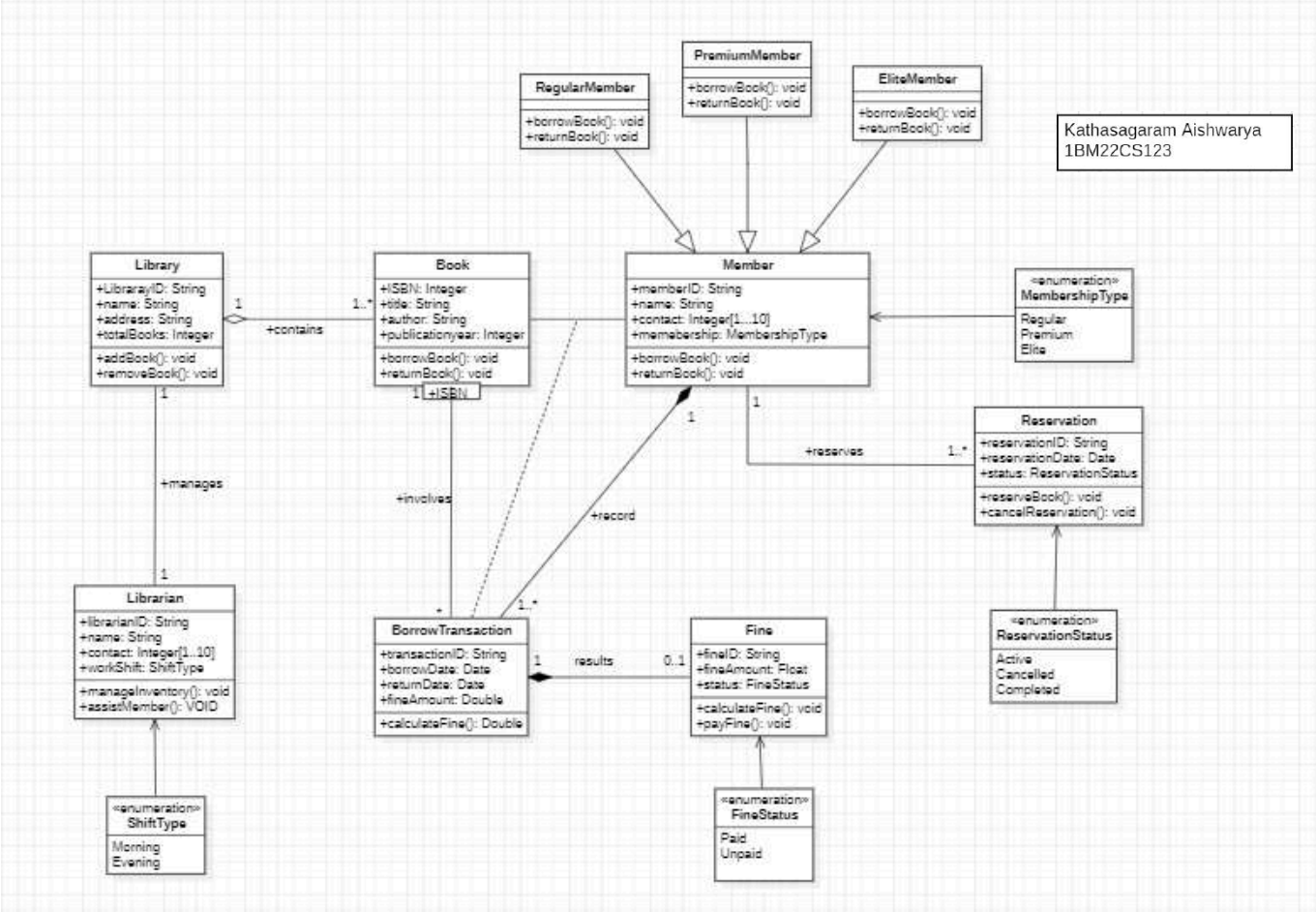
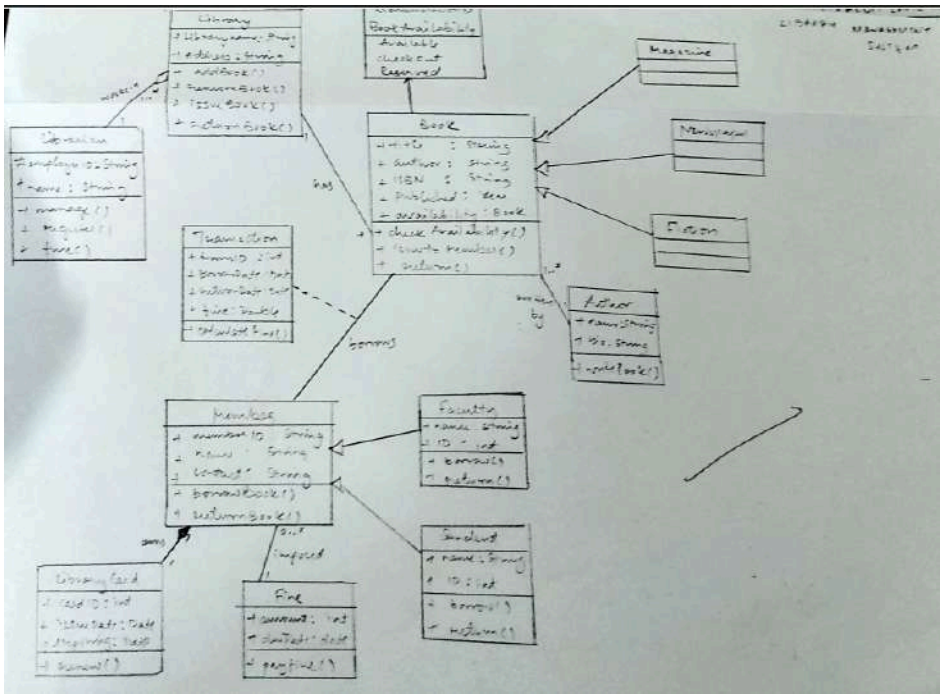



*Figure 3.1: Class Diagram*

The library system consists of several key classes for managing books, members, and staff. The Library class holds details like the library ID, name, address, and books, with methods to add or remove books. The Book class represents individual books with attributes like ISBN, title, and author, and allows borrowing and returning.

The Member class tracks library members, with attributes such as ID, name, contact information, and membership type (Regular, Premium, Elite). Each type offers different benefits. Librarians manage the library's operations, including inventory and assisting members, based on their shifts.

The BorrowTransaction class records book loans, while the Reservation class handles book reservations. The Fine class manages fines for late returns. Enumerations like MembershipType, ShiftType, and FineStatushelp define the system's operations and workflows. Relationships between these classes, such as members borrowing or reserving books, ensure efficient library management.



## STATE DIAGRAM

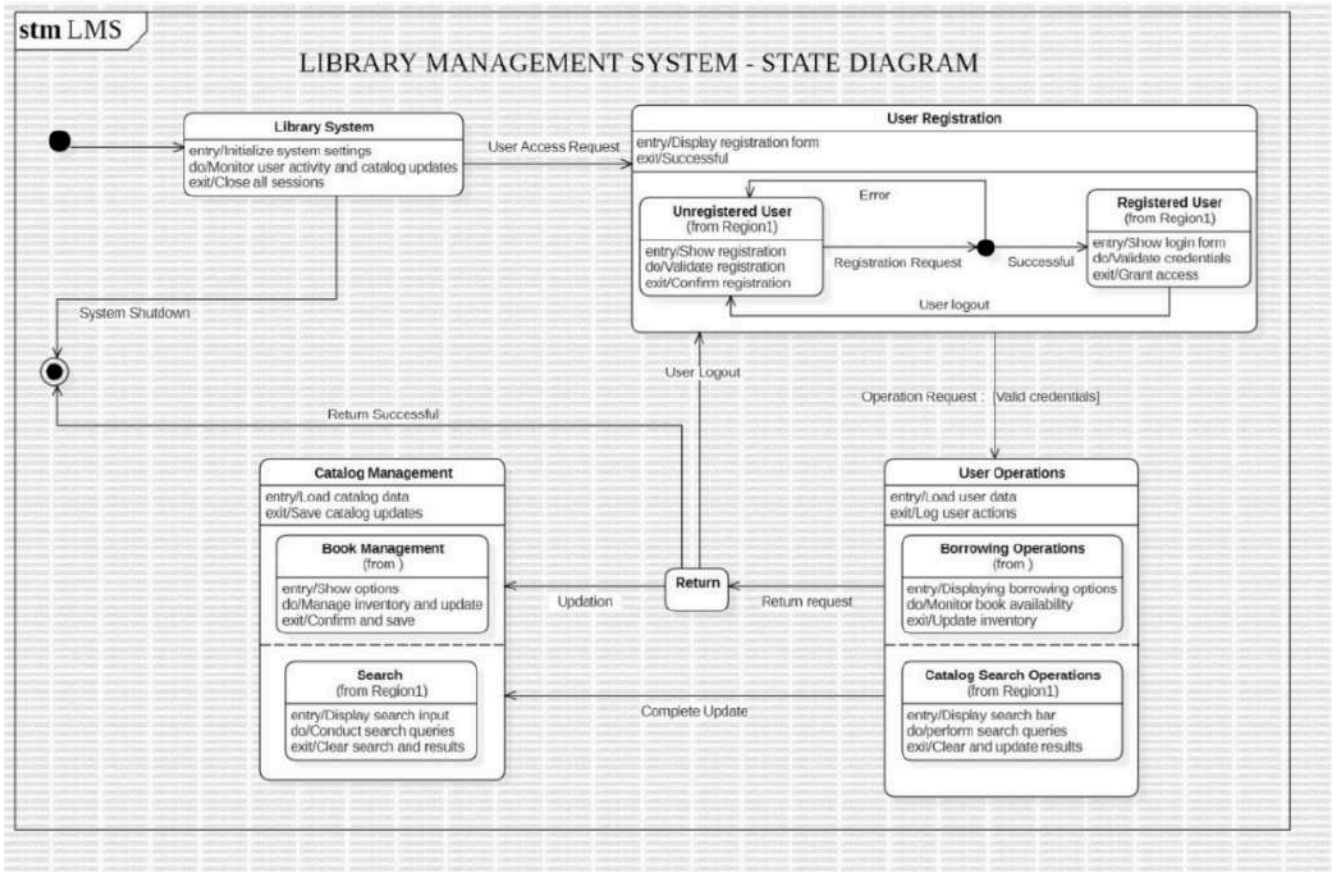
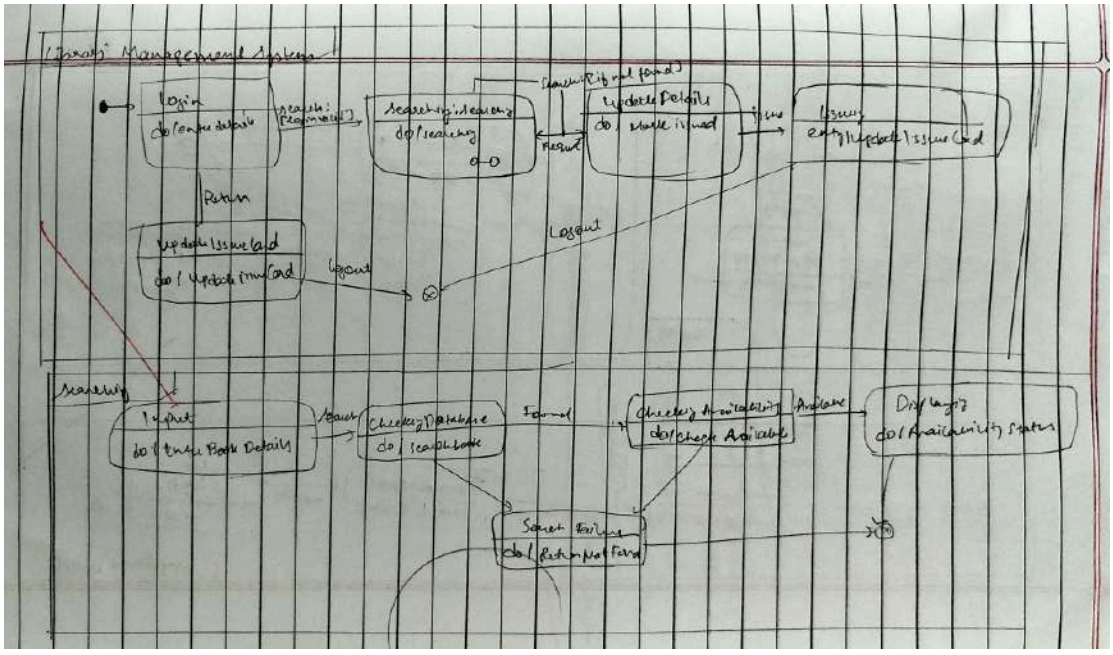



*Figure 3.2: State Diagram*

The state diagram of the Library Management System illustrates the key states and transitions in the system. It starts with the Library System state, where the system is initialized and ready for user interaction. Users either remain in the Unregistered User state or transition to User Registration if they are new. After successful registration, they move to the Registered User state to log in and access the system.

Key states include Catalog Management for managing the library's catalog, Book Management for borrowing, returning, and reserving books, and Search for searching the catalog. The User Operations state allows users to perform actions like borrowing and returning books. The Return state is temporary during the return process, while the Errorstate handles issues like invalid credentials.

Transitions are triggered by actions such as requesting to access the system, register, or perform operations. Activities in each state include displaying forms, validating data, managing inventory, and updating records. This state diagram provides an overview of how the system manages user interactions and library functions.



## USECASE DIAGRAM

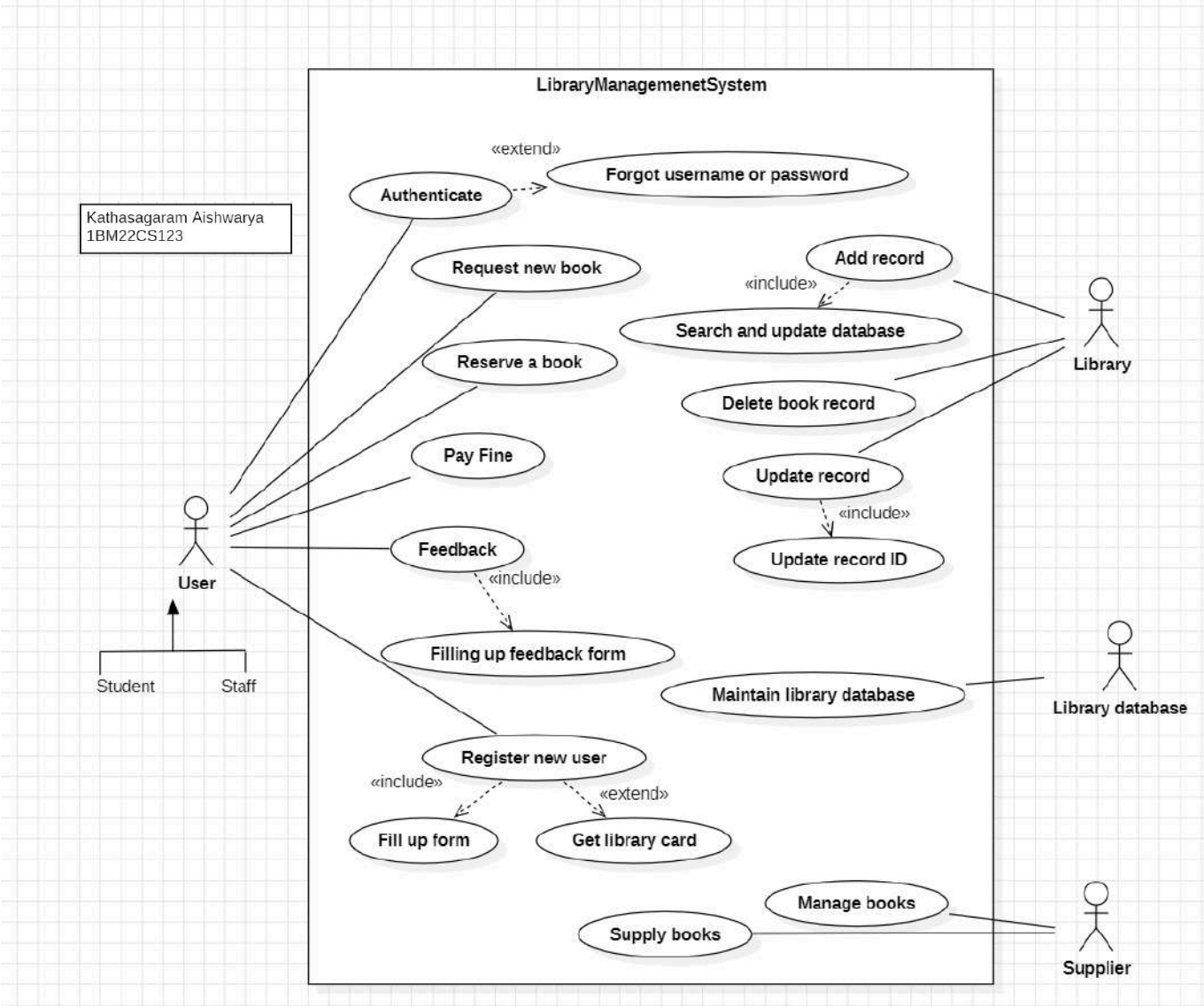



*Figure 3.3: Use Case Diagram*

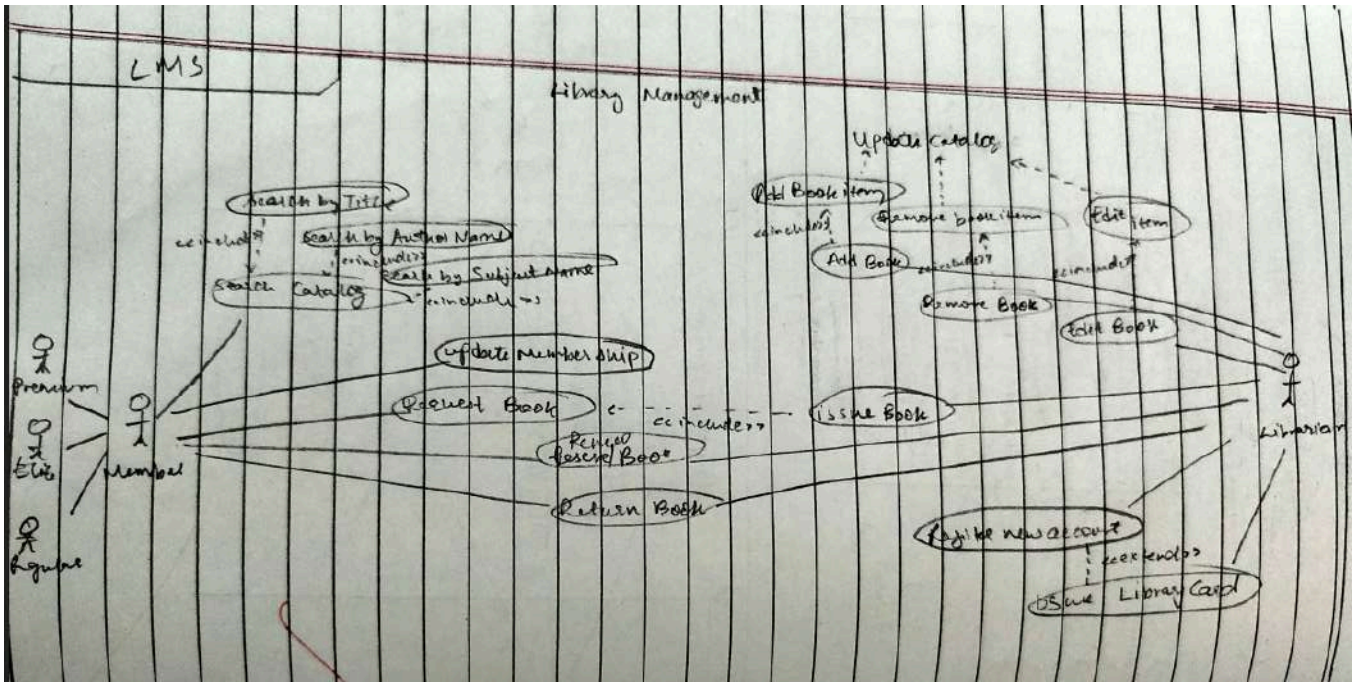
The use case diagram for the Library Management System outlines key functionalities and interactions within the system. Members are categorized into different types: Regular, Elite, and Premium, each with varying levels of borrowing privileges. The Librarian manages library operations, such as catalog updates, issuing books, and registering new accounts.

The main use case, Library Management, includes tasks like Update Catalog, which encompasses adding, removing, and editing book items. Add Book and Remove Book are specific actions related to catalog management. Update Membership allows for the updating

of member information or membership types.

The system also allows members to Search Catalog by title, author, or subject, request unavailable books, renew borrowed books, and return them. Issue Book is a use case for librarians to issue books to members, while Register New Account and Issue Library Card are used to register new members and provide them with cards.

The diagram highlights Includes relationships, showing that actions like updating the catalog include adding, removing, and editing books. It also demonstrates Inheritance, with Elite and Premium members inheriting the characteristics of Regular members. This use case diagram provides an overview of the key services, member interactions, and librarian responsibilities within the library system.



## SEQUENCE DIAGRAM

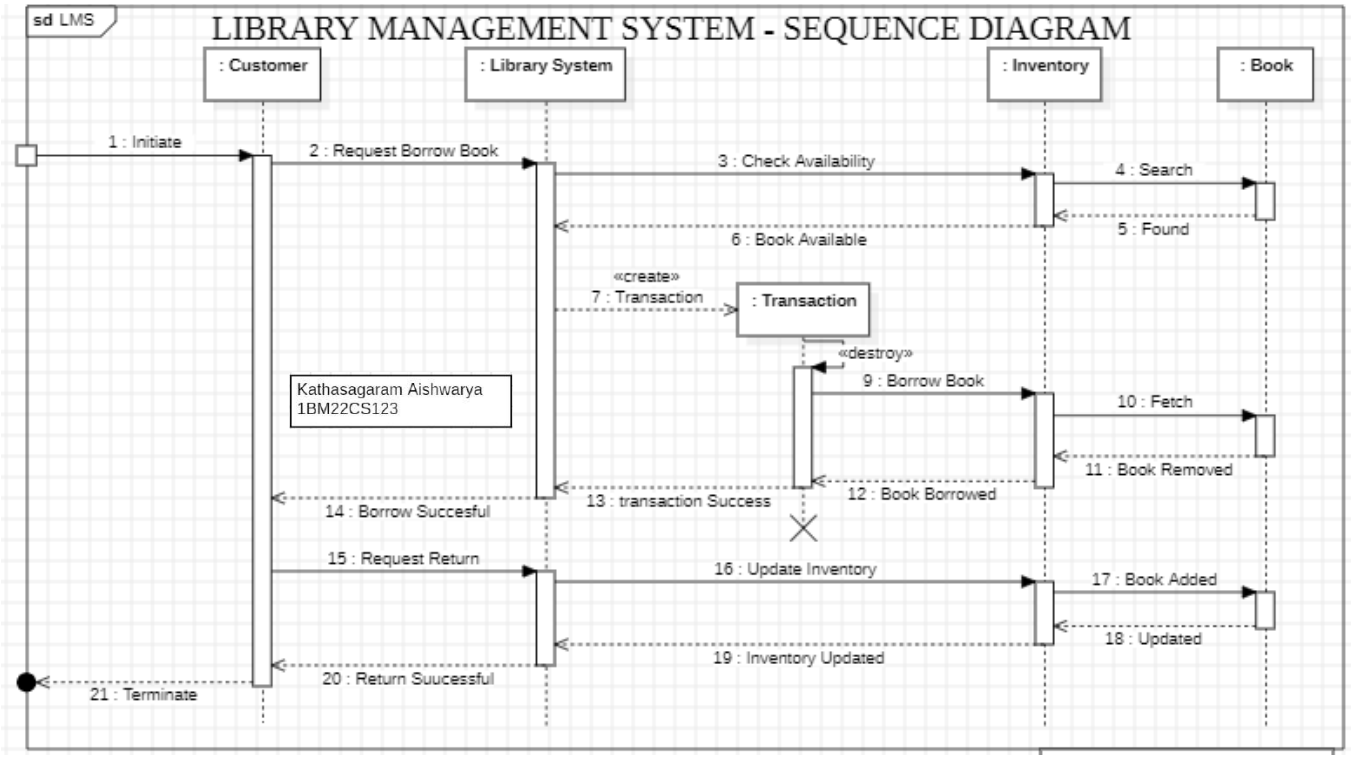


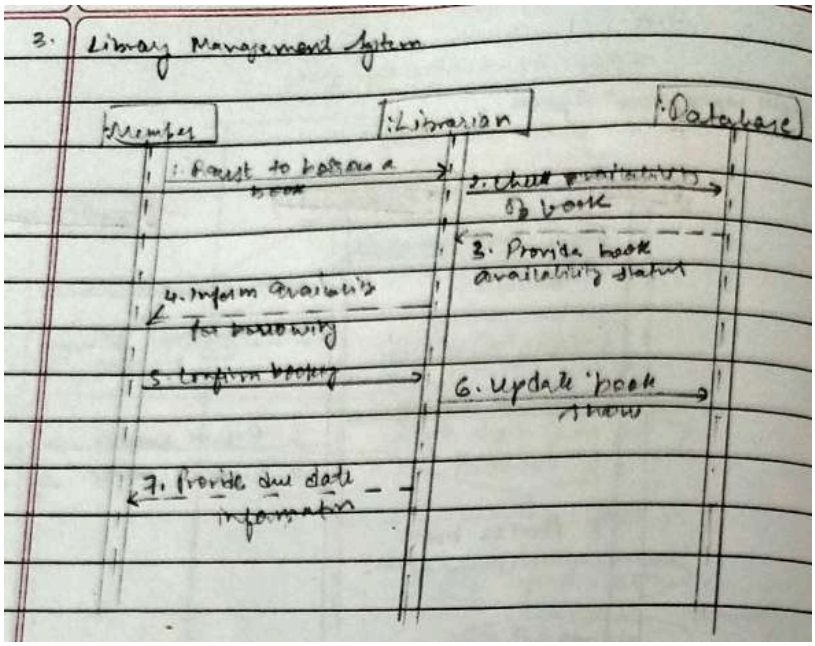

*Figure 3.4: Sequence Diagram*

The Sequence Diagram for the Library System illustrates the interactions between the Customer, Library System, Inventory, and Book objects during the borrowing and returning process. The Customer initiates the process by requesting to borrow a book. The Library System checks the Inventory for availability by searching for the book, and once found, informs the customer that the book is available.

A Transaction object is created to manage the borrowing process. The Library System requests the book from the Inventory, which fetches the book, removes it from the inventory, and confirms the borrowing process. The Customer is then informed that the book has been successfully borrowed.

When the Customer returns the book, they send a Request Return to the Library System, which updates the Inventoryand adds the book back to the stock. The Inventory confirms the update, and the Library System informs the Customerthat the return was successful. Finally, the interaction is terminated.

This diagram showcases the flow of actions, including the creation and destruction of the Transaction object, that governs the borrowing and returning of books in the library system.




## ACTIVITY DIAGRAM

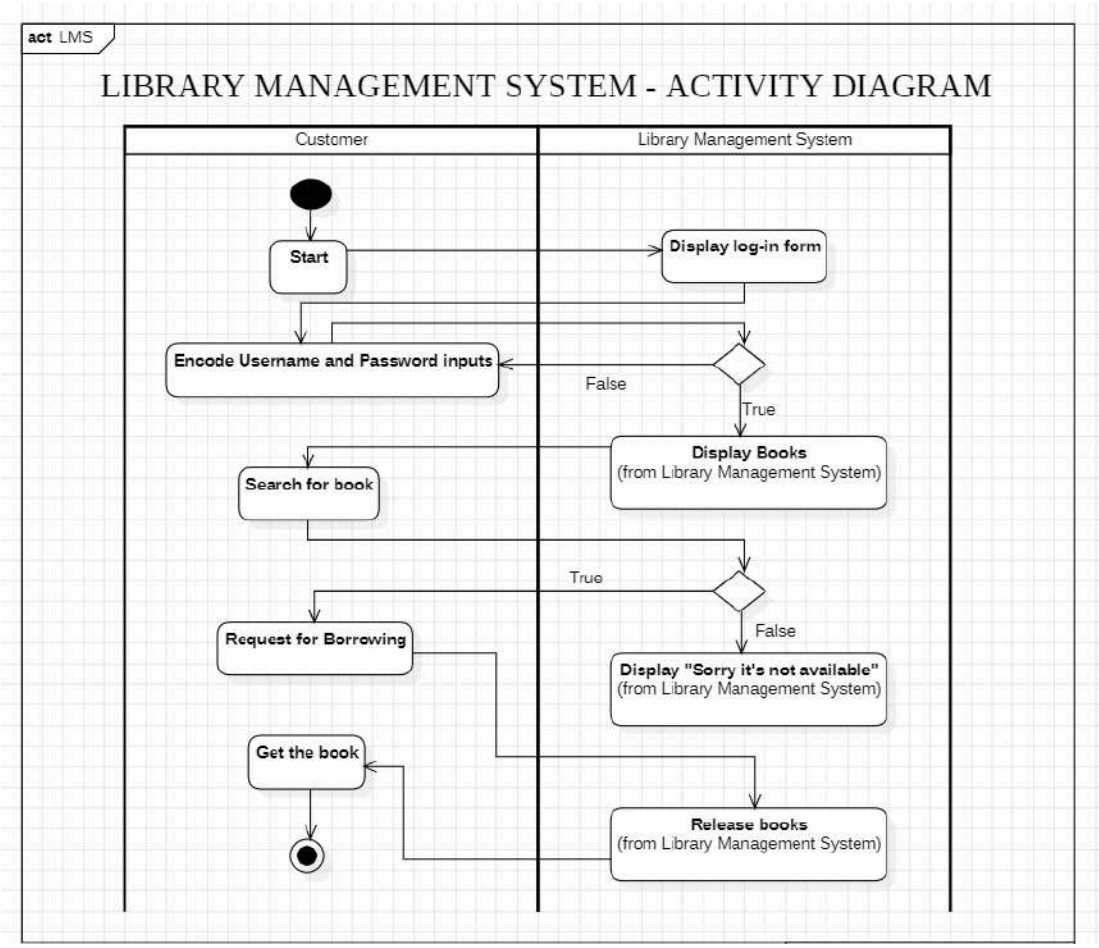
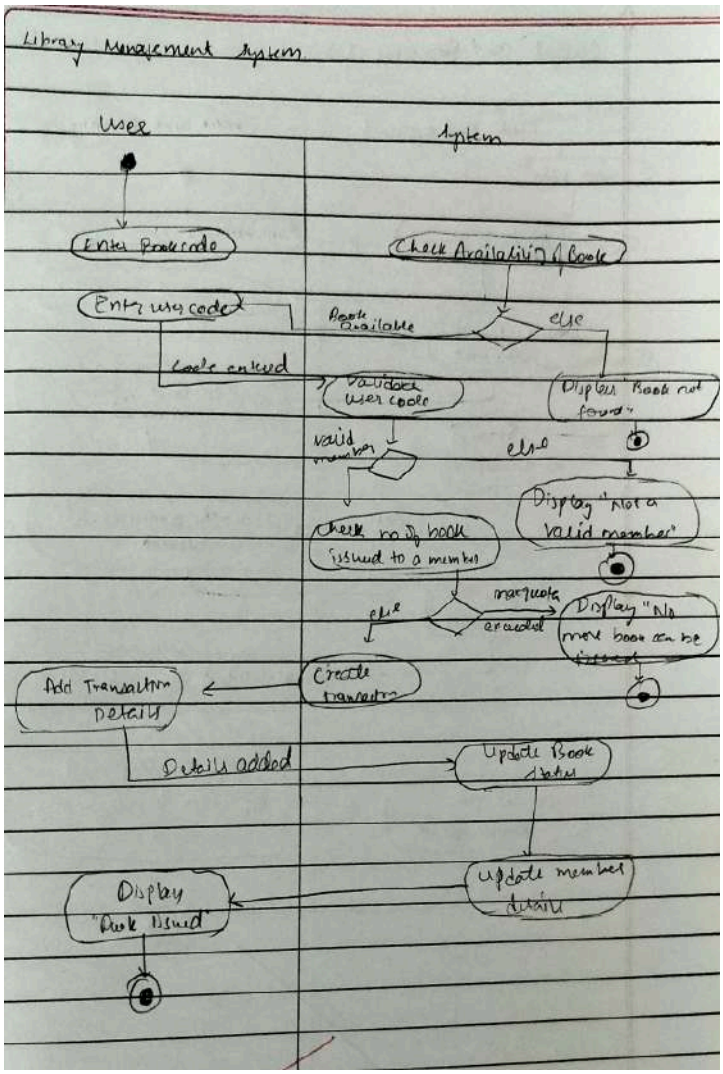



*Figure 3.5: Activity Diagram*

This Swimlane diagram outlines the roles of two entities: the Customer and the Library Management System. The Customer begins by logging into the system with their credentials. After successful login, they search for a book in the catalog. Upon finding the desired book, they request to borrow it and ultimately receive the book. These steps represent the customer's interaction with the library system during the borrowing process.

The Library Management System is responsible for managing the entire flow of actions. It starts by displaying the login form to the customer. After login, the system shows the available books and, if the requested book is unavailable, it notifies the customer. The system also handles the release of books to customers and updates the inventory once a book is borrowed. This clear division of responsibilities between the customer and the system ensures a smooth borrowing experience.



# 4. STOCK MAINTENANCE SYSTEM

## 4.1 Problem Statement

The Stock Maintenance System helps businesses manage inventory by tracking stock levels, sales, and orders. Key classes include Product, Stock, Order, and Supplier. The system should facilitate adding and updating stock quantities, generating purchase orders, and alerting users about low stock levels. Relationships, such as suppliers providing multiple products and products being part of multiple orders, need to be modeled. The system should provide reports on stock trends and help optimize inventory levels for operational efficiency.

The objective of this project is to develop a reliable and efficient Stock Maintenance System to streamline inventory management and enhance operational efficiency. The proposed system aims to address the following critical functionalities:

1. Inventory Management: Enable real-time tracking of stock levels, categorize items, and manage stock additions, deletions, and adjustments to ensure accurate inventory records.

2. Supplier Management: Provide tools to maintain supplier information, manage purchase orders, and track delivery schedules to ensure seamless stock replenishment.

3. Stock Monitoring: Implement mechanisms to monitor stock usage, set reorder thresholds, and generate alerts for low stock levels to prevent shortages.

4. Sales and Distribution Tracking: Facilitate tracking of stock movement related to sales and distribution, maintaining accurate records of outgoing stock and sales performance.

5. Reporting and Analytics: Generate detailed reports on inventory levels, stock turnover, and supplier performance, along with analytics to support strategic decision-making.

6. Integration with Financial Systems: Allow integration with financial systems to streamline cost management, calculate stock value, and manage accounts related to stock transactions.

7. Security and Access Control: Ensure secure access to the system, implement role-based access

permissions, and maintain data integrity to prevent unauthorized actions.

This system will centralize and automate stock maintenance processes, reducing manual effort, improving accuracy, and supporting efficient stock management for businesses.

## 4.2 Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for a Stock Maintenance System (SMS). The system is designed to automate and streamline stock management processes, including inventory tracking, stock level monitoring, purchase order generation, and reporting. This document provides a comprehensive outline of the functional and non-functional requirements, design constraints, and system interfaces.

1.2 Scope

The SMS is a centralized platform developed to efficiently manage inventory for businesses of various sizes. It enables users to monitor stock levels, track item movements, manage suppliers, and generate analytical reports. The system integrates seamlessly with existing enterprise resource planning (ERP) tools and supports scalability to accommodate growing business needs.

1.3 Definitions, Acronyms, and Abbreviations

- SMS: Stock Maintenance System
- API: Application Programming Interface
- ERP: Enterprise Resource Planning
- GUI: Graphical User Interface

1.4 References

- IEEE Standard for Software Requirements Specifications (IEEE Std 830-1998)
- Industry Best Practices for Inventory Management

### 1.5 Overview

This document details the overall description and specific requirements of the SMS. Section 2 provides a high-level overview of the system's features and user characteristics, while Section 3 outlines the functional and non-functional requirements.

## 2. Overall Description

### 2.1 Product Perspective

The SMS is a web-based application designed to handle end-to-end inventory management tasks. It integrates with existing accounting and ERP systems to provide real-time insights into stock levels, item movements, and supplier data.

### 2.2 Product Features

Key features of the SMS include:

- Real-time inventory tracking
- Automated alerts for low stock levels
- Purchase order management
- Detailed reporting and analytics dashboards
- Integration with ERP systems and supplier databases

### 2.3 User Characteristics

The system is designed for the following user groups:

- Inventory Managers: Oversee stock levels, supplier interactions, and order processing.
- Warehouse Staff: Track item movements and update stock statuses.
- Administrators: Configure system settings, user roles, and permissions.
- Business Owners: Access analytics for decision-making.

### 2.4 Constraints

- The system must comply with data protection regulations.
- Must ensure compatibility with standard hardware, including barcode scanners.
- High availability is required to support critical inventory operations.

### 2.5 Assumptions and Dependencies

- Reliable internet connectivity is assumed for real-time operations.
- The system relies on third-party APIs for ERP integration.

3. Specific Requirements

3.1 Functional Requirements

- Inventory Tracking:
    - Monitor stock levels in real time across multiple locations.
    - Track item movements, including additions, removals, and transfers.
- Alerts and Notifications:
    - Generate alerts for low stock levels and pending purchase orders.
    - Notify users of discrepancies in stock levels.
- Purchase Order Management:
    - Create, approve, and track purchase orders.
    - Maintain supplier information and order history.
- Reporting and Analytics:
    - Generate detailed reports on inventory trends, stock usage, and supplier performance.
    - Provide visual dashboards for quick insights.

3.2 Performance Requirements

- The system should support up to 1,000 simultaneous users.
- Response time for inventory queries must not exceed 2 seconds.
- Daily automated backups must be performed to ensure data security.

3.3 Interface Requirements

- GUI:
    - User-friendly design with role-specific dashboards.
    - Support for desktop and mobile browsers.
- API Integration:
    - RESTful APIs for ERP and supplier database integration.

3.4 Design Constraints

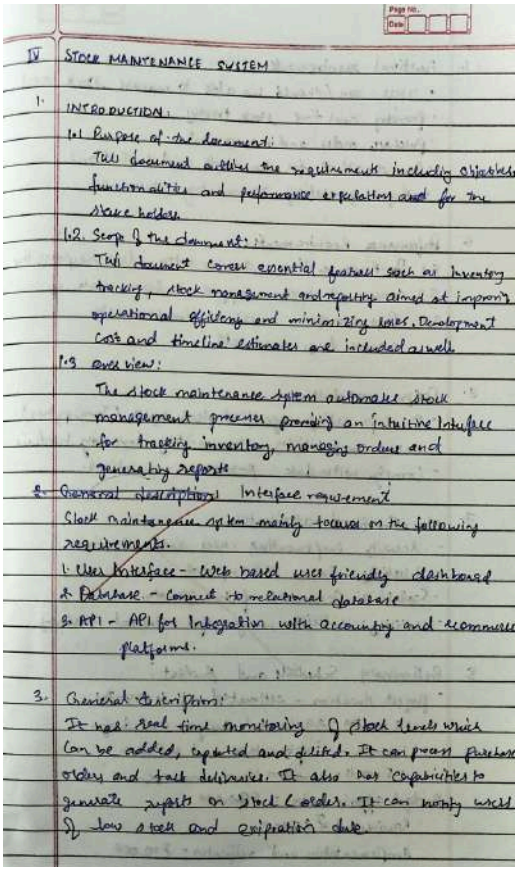
- The system must be developed using scalable and secure technologies.
- Compatibility with barcode scanning hardware is required.

3.5 Non-Functional Attributes

- Reliability: 99.9% uptime to ensure uninterrupted inventory management.
- Security: Data encryption and secure user authentication are mandatory.
- Usability: Provide an intuitive interface for users with varying technical expertise.
- Scalability: Efficiently handle increasing inventory sizes and locations.
- Maintainability: Modular architecture to support easy updates and debugging.

3.6 Performance, Schedule, and Budget

- Performance: Handle peak inventory updates during high-demand periods without delays.
- Schedule: Development and deployment to be completed within 15 months.
- Budget: Estimated at $200,000, including development, testing, and integration costs.


IV STOCK MAINTENANCE SYSTEM

1. INTRODUCTION:

1.1 Purpose of the document:

This document outlines the requirement including objectives functionalities and performance expectations and for the stake holders.

1.2. Scope of the document:

This document covers essential features such as inventory tracking, stock management and reporting aimed at improving operational efficiency and minimizing losses. Development cost and timeline estimates are included as well.

1.3 overview:

The stock maintenance system automates stock management process providing an intuitive interface for tracking inventory, managing orders and generating reports.

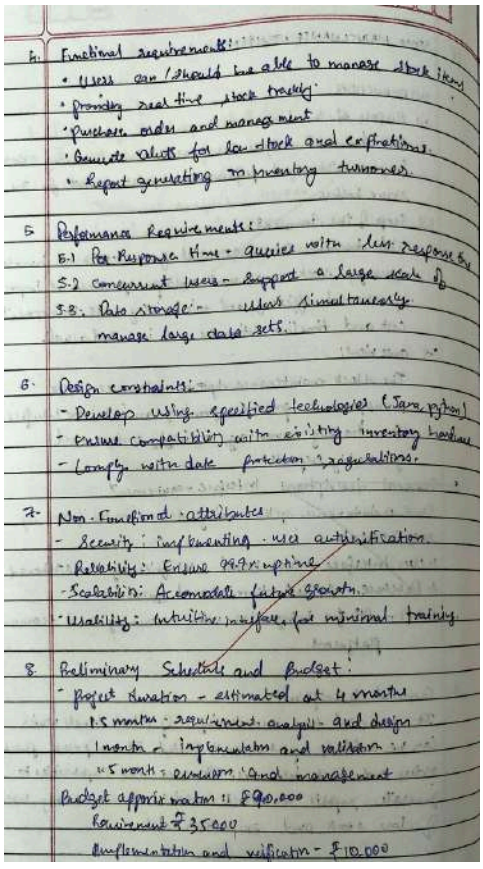
2. General description: Interface requirement

Stock maintenance system mainly focuses on the following requirements.

1. User interface - Web based user friendly dashboard
2. Database - Connect to relational database
3. API - API for integration with accounting and ecommerce platforms.

3. General description:

It has: real time monitoring of stock levels which can be added, updated and deleted. It can process purchase orders and track deliveries. It also has capabilities to generate reports on stock order. It can notify users of low stock and expiration date.


4. Functional requirements:

- Users can / should be able to manage stock items
- Provide real time stock tracking
- Purchase orders and management
- Generate alerts for low stock and expirations
- Report generating on inventory turnovers

5 Performance Requirements:

5.1 For Response time - queries with less response time

5.2 Concurrent users - Support a large scale of

5.3. Data storage - users simultaneously manage large data sets.

6. Design constraints:

- Develop using specified technologies (Java, python)
- Ensure compatibility with existing inventory hardware
- Comply with data protection regulations.

7. Non-Functional attributes

- Security: Implementing user authentication.
- Reliability: Ensure 99.9% uptime
- Scalability: Accomodate future growth.
- Usability: Intuitive interface, for minimal training

8 Preliminary Schedule and Budget:

- Project duration - estimated at 4 months
  1.5 months - requirement analysis and design
  1 month - implementation and validation
  0.5 month - evaluation and management

Budget approximation: $90,000

Requirement $35000

Implementation and verification - $10,000

## CLASS DIAGRAM

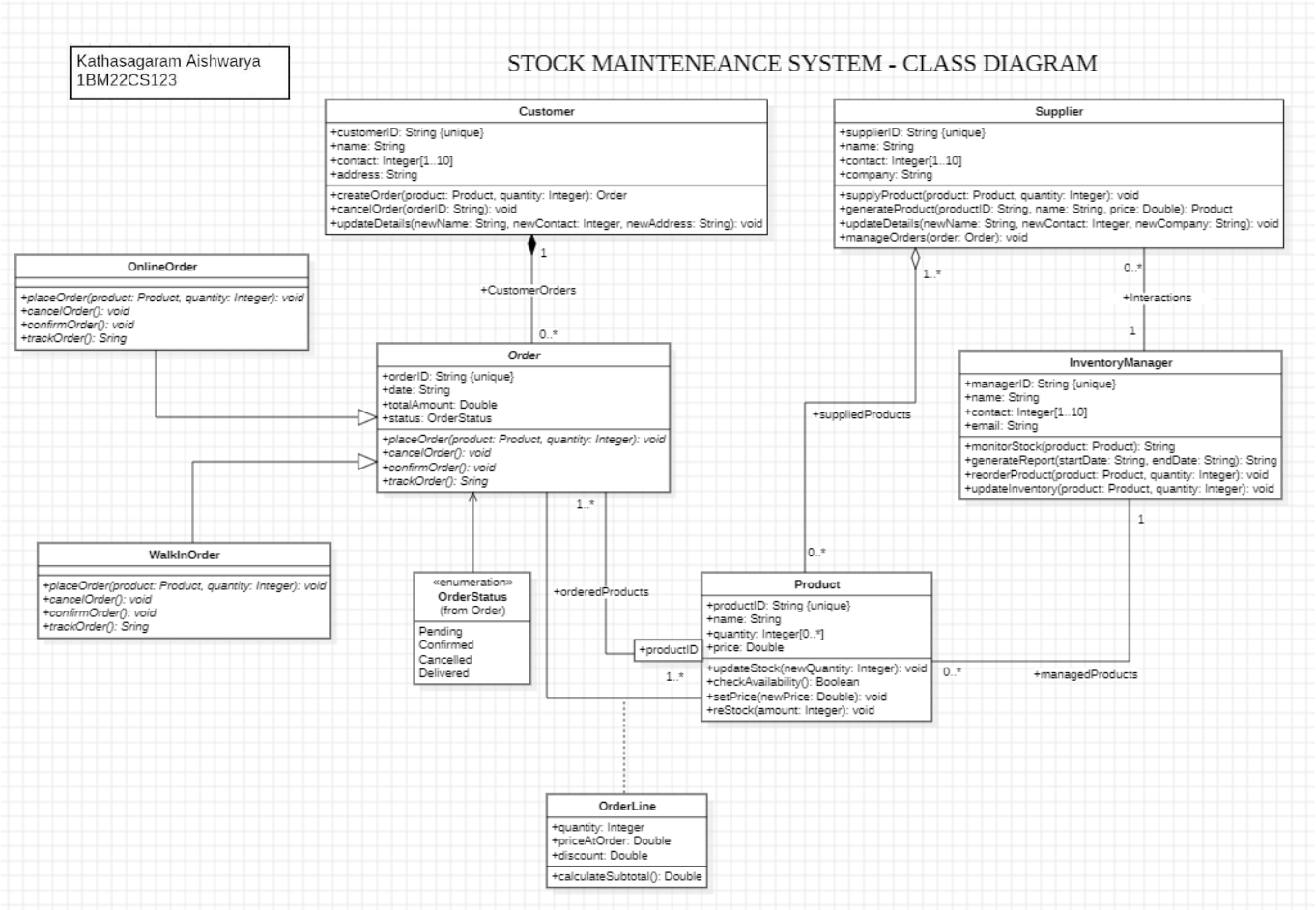
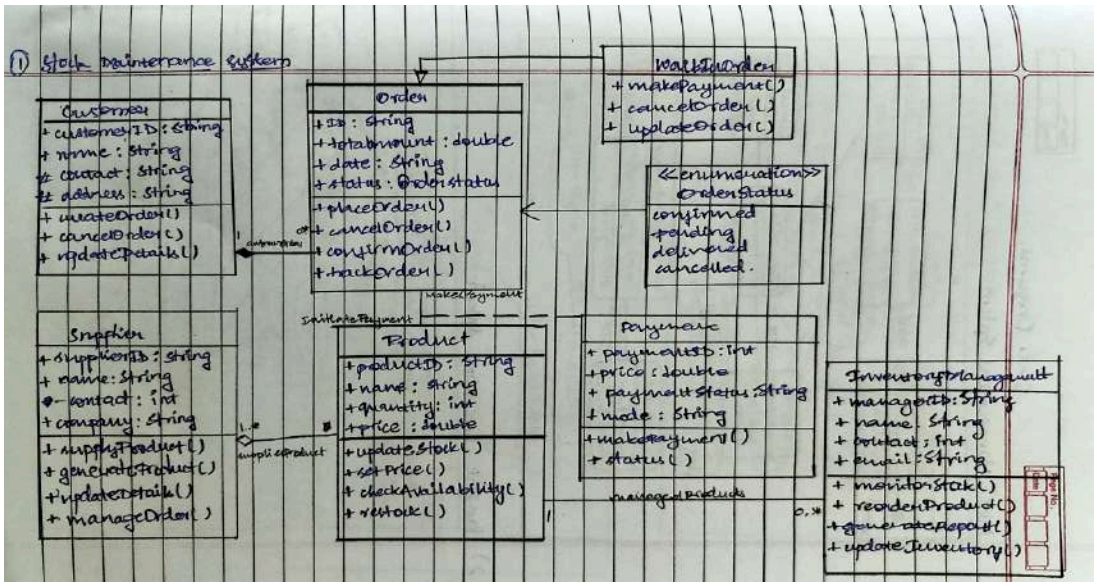



*Figure 4.1: Class Diagram*

This system involves multiple classes representing different aspects of an order management process. The key classes include Customer, who places orders; Supplier, who supplies products; and Product, which represents the items available for purchase. Order is the central class, with its subclasses OnlineOrder and WalkinOrder, representing different types of orders. Each order can contain multiple OrderLine items, which reference specific products and their quantities. InventoryManager oversees product management, ensuring stock levels are monitored and orders are processed efficiently. CustomerOrders represents all orders placed by a particular customer, linking back to the Customer.

The system operates with several associations that manage relationships between entities. A customer can place multiple orders, each of which contains multiple order lines referencing products. Suppliers provide products, and an inventory manager handles product stock, manages orders, and generates reports. Additionally, the OrderStatus enumeration tracks the status of each order, with statuses like Pending, Confirmed, Cancelled, and Delivered. The multiplicity between the entities allows for flexibility, where

one customer can place multiple orders, and one inventory manager manages multiple products and orders.




**STATE DIAGRAM**

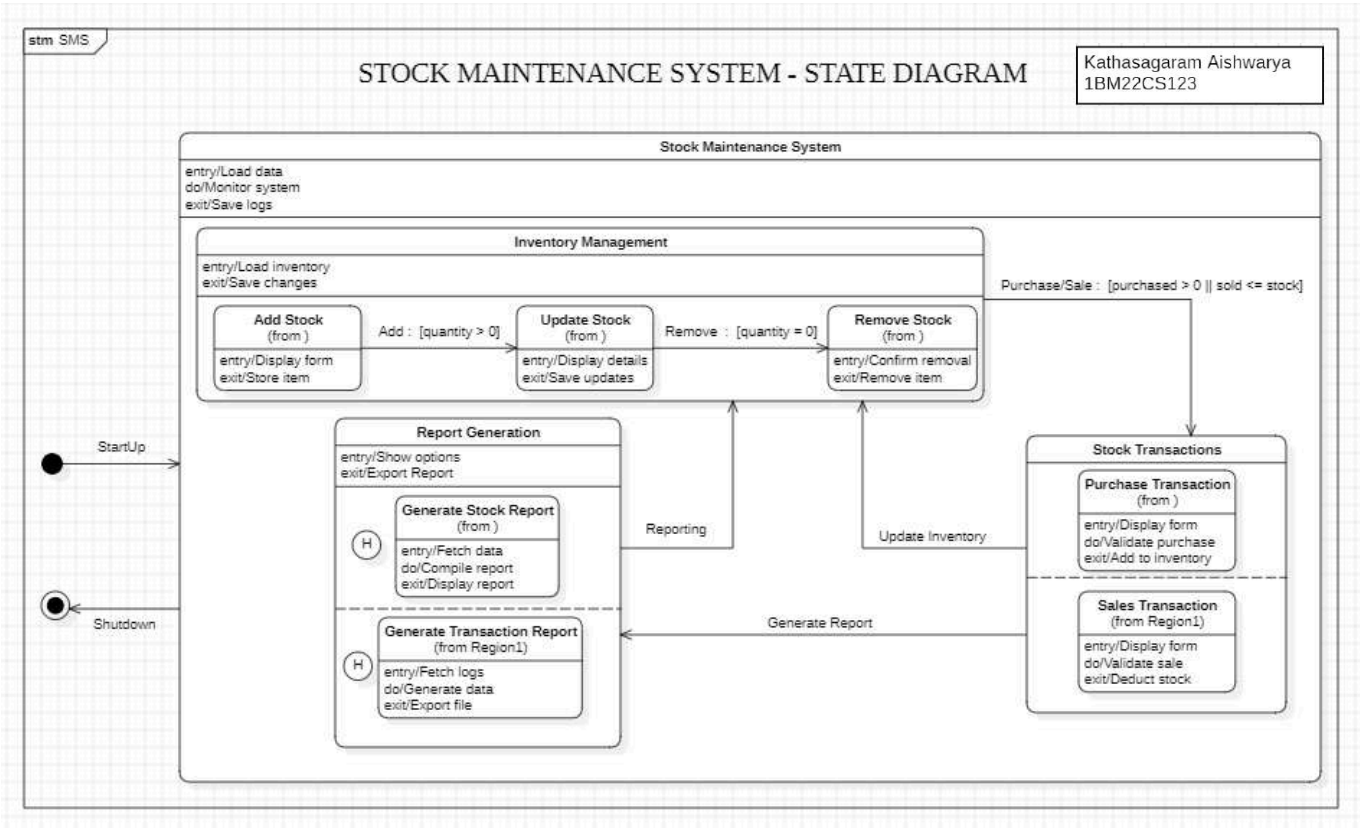
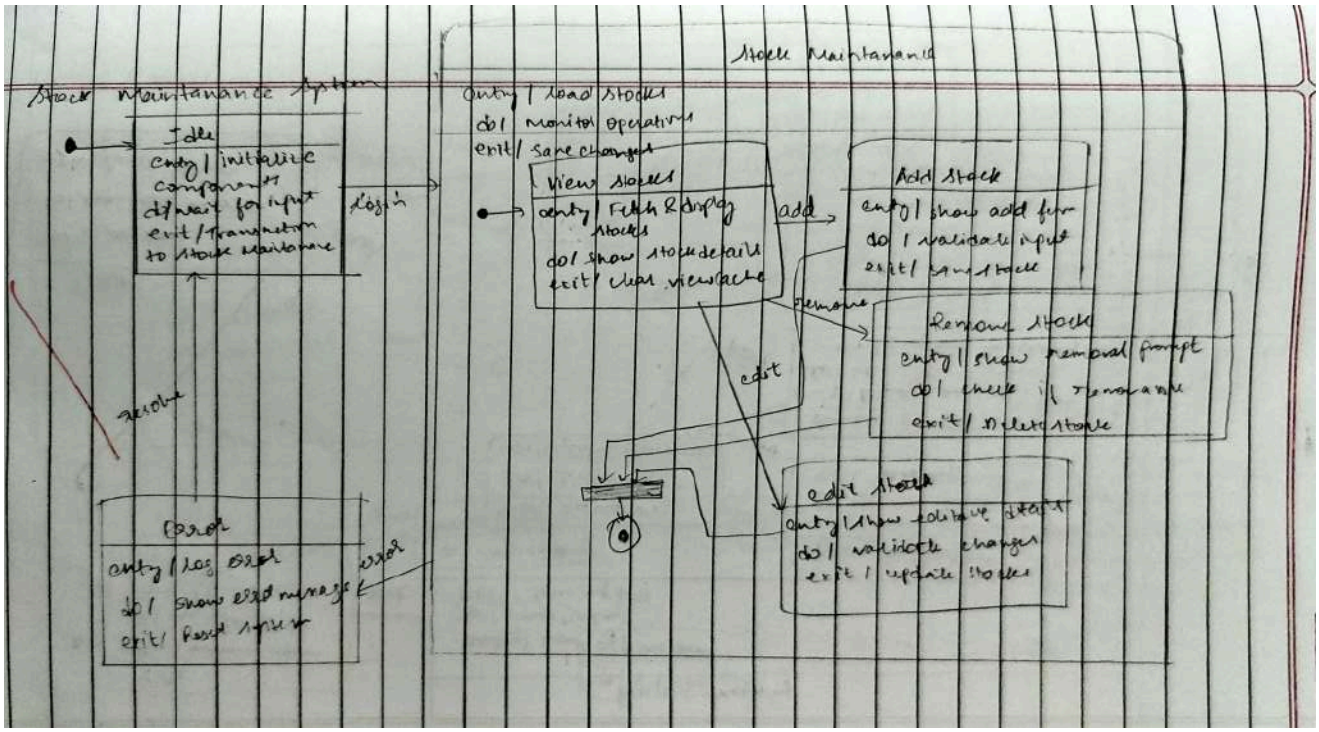



*Figure 4.2: State Diagram*

The state diagram of the Stock Maintenance System outlines the key states and transitions involved in

managing inventory and transactions. It begins with the Stock Maintenance System state, where the system is initialized and ready for user interaction. The system then transitions into the Inventory Management state, where actions such as adding, updating, and removing stock items are performed. Specific states like Add Stock, Update Stock, and Remove Stock handle individual operations related to inventory adjustments. The Report Generation state allows for the creation of various reports, including stock and transaction reports, with actions to compile, display, and export these reports.

The system also manages transactions under the Stock Transactions superstate, which includes Purchase Transaction and Sales Transaction. These states are responsible for handling the purchase and sale of products, updating inventory accordingly. The activities in each state focus on loading data, saving changes, validating transactions, and updating inventory. The diagram provides an efficient structure for managing inventory items, generating reports, and processing transactions, ensuring smooth operations within the Stock Maintenance System.



## USECASE DIAGRAM

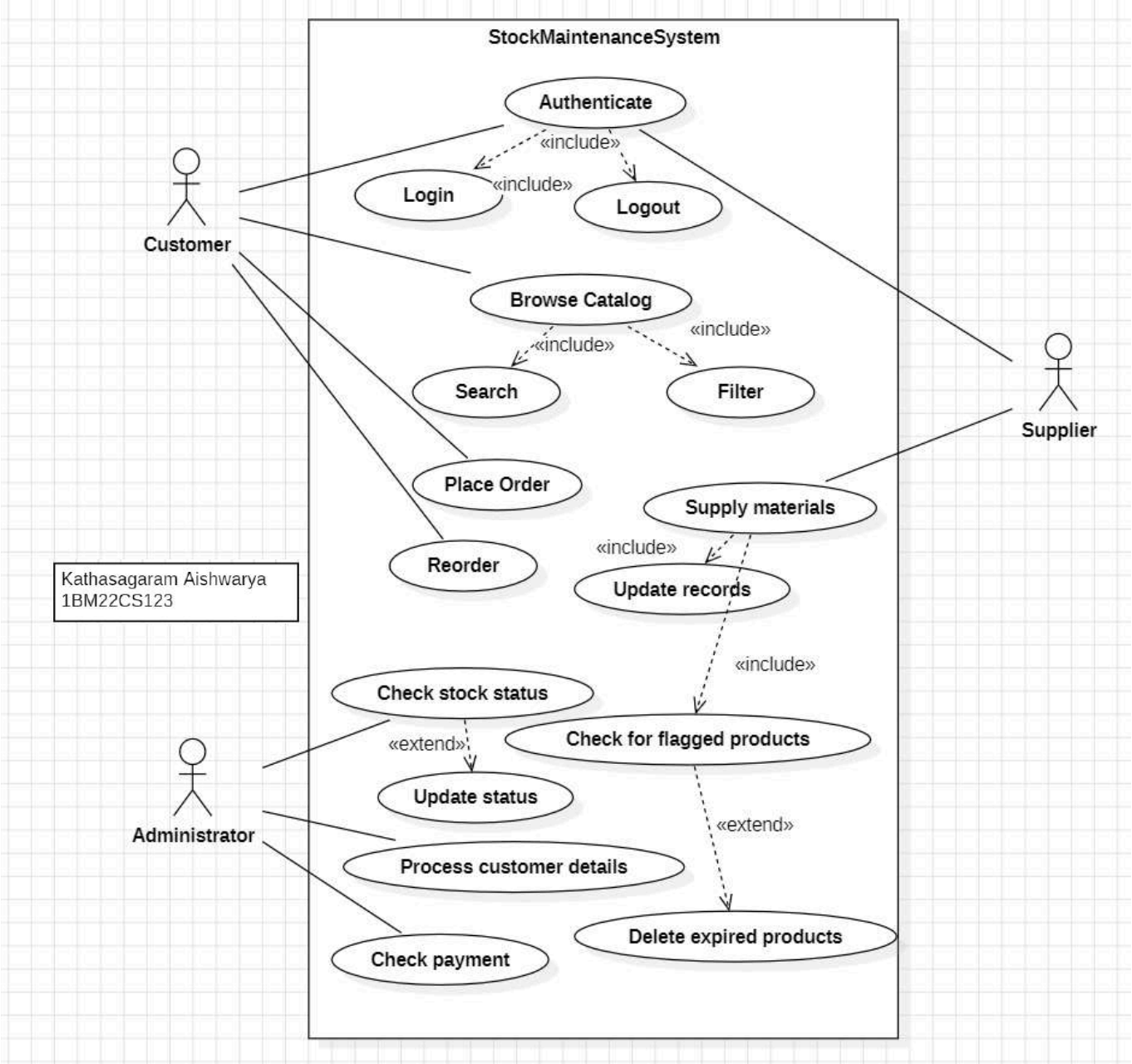
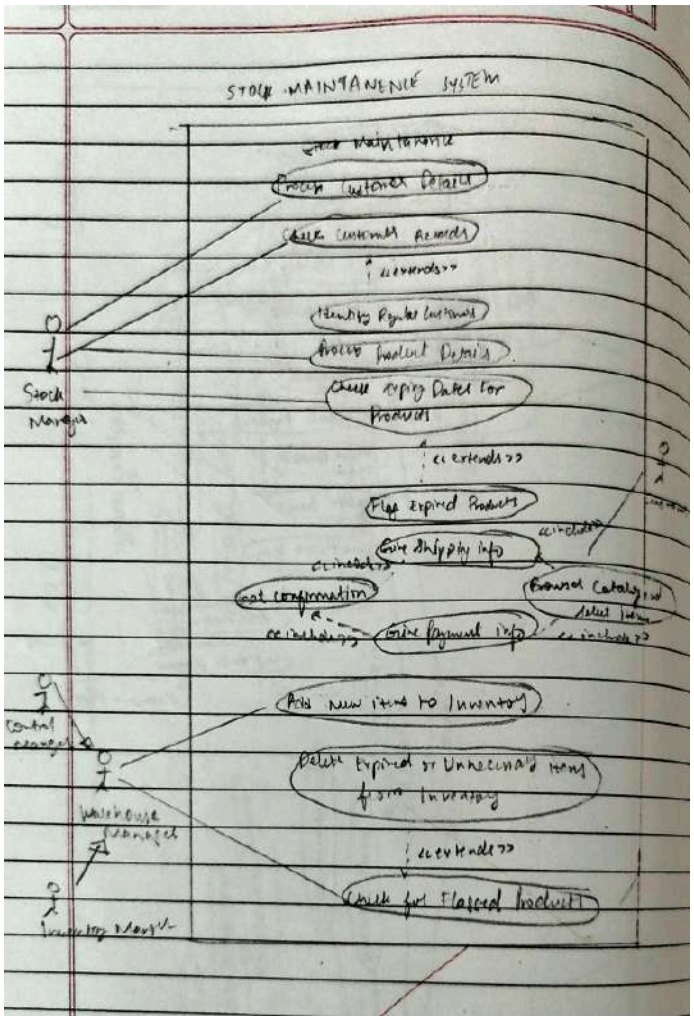



*Figure 4.3: Use Case Diagram*

The use case diagram for the Stock Maintenance System defines key actors and the interactions that facilitate the management of inventory and customer transactions. Key actors include the Stock Manager, Control Manager, Warehouse Manager, Inventory Manager, and the Customer. These actors interact with various use cases that ensure smooth stock management, product details processing, and customer transactions. The Stock Maintenance use case is central to this system, incorporating tasks such as managing customer and product details, providing shipping information, obtaining customer order

confirmations, and updating inventory with new items or removing expired stock.

Several extensions and inclusions enhance the use cases. For example, Check Customer Records and Identify Regular Customer extend the Process Customer Details use case, allowing for loyalty programs or special promotions. Similarly, the Check Expiry Dates for Products and Flag Expired Products are extensions under the Process Product Details use case to handle product expiration. The Get Confirmation use case includes Give Payment Info and Browse Catalog and Select Items to facilitate customer orders. The diagram provides a clear overview of how different roles interact to manage inventory, process orders, and ensure efficient stock flow, from adding new products to removing expired ones.



## SEQUENCE DIAGRAM

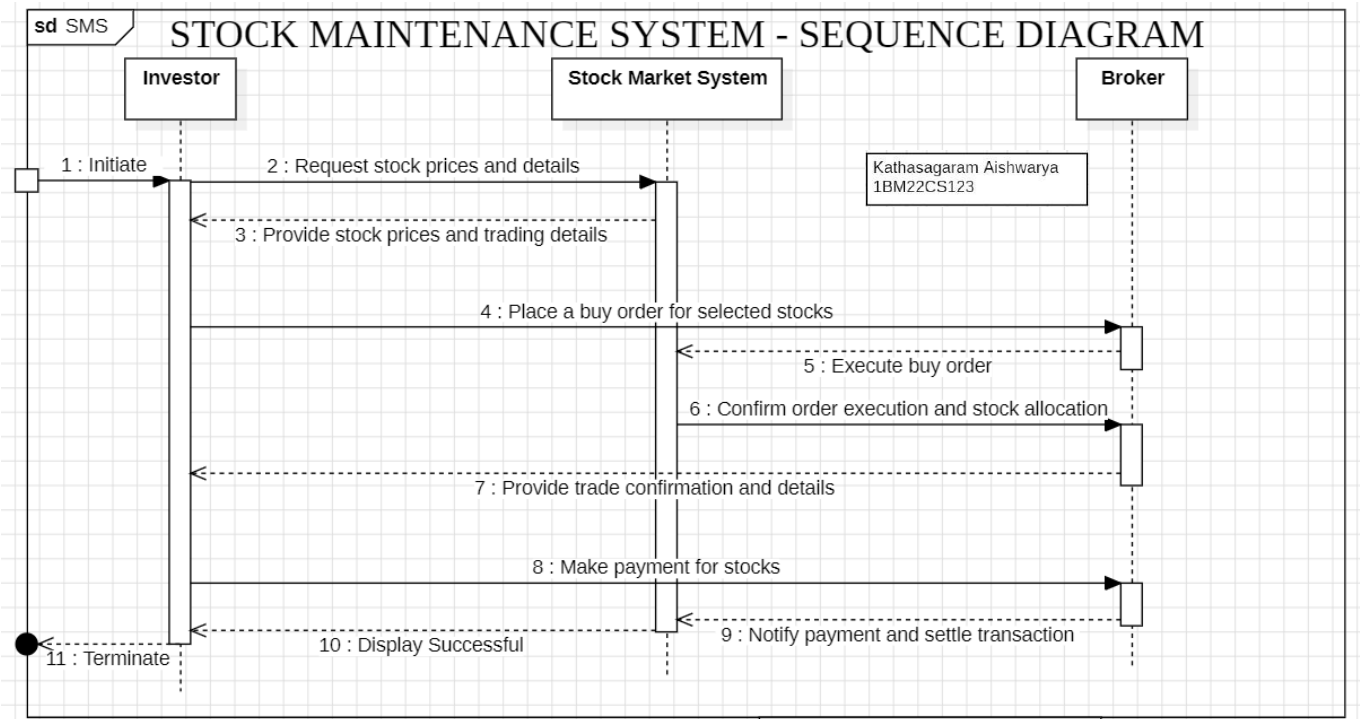
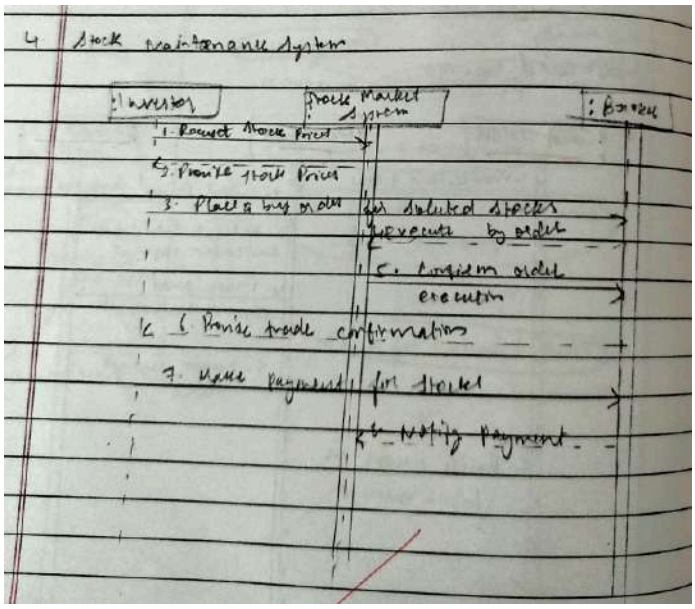



*Figure 5.4: Sequence Diagram*

The Sequence Diagram for stock trading outlines the process and interactions between the Investor, Stock Market System, and Broker during a stock purchase transaction. Initially, the Investor requests stock prices and trading details from the Stock Market System, which provides the necessary information. The Investor then places a buy order through the Broker, who executes it on behalf of the Investor using the Stock Market System. The system confirms the execution of the buy order, allocates the purchased stocks to the Investor's account, and provides the Investor with a confirmation of the trade.

Next, the Investor makes payment for the purchased stocks to the Broker, who notifies the Stock Market System of the payment and settles the transaction. The system displays a success message, and the transaction process concludes. This diagram effectively represents the flow of information and actions required for a successful stock trade, highlighting the roles and responsibilities of each party involved, from the initial order to the final confirmation and transaction settlement.



## ACTIVITY DIAGRAM

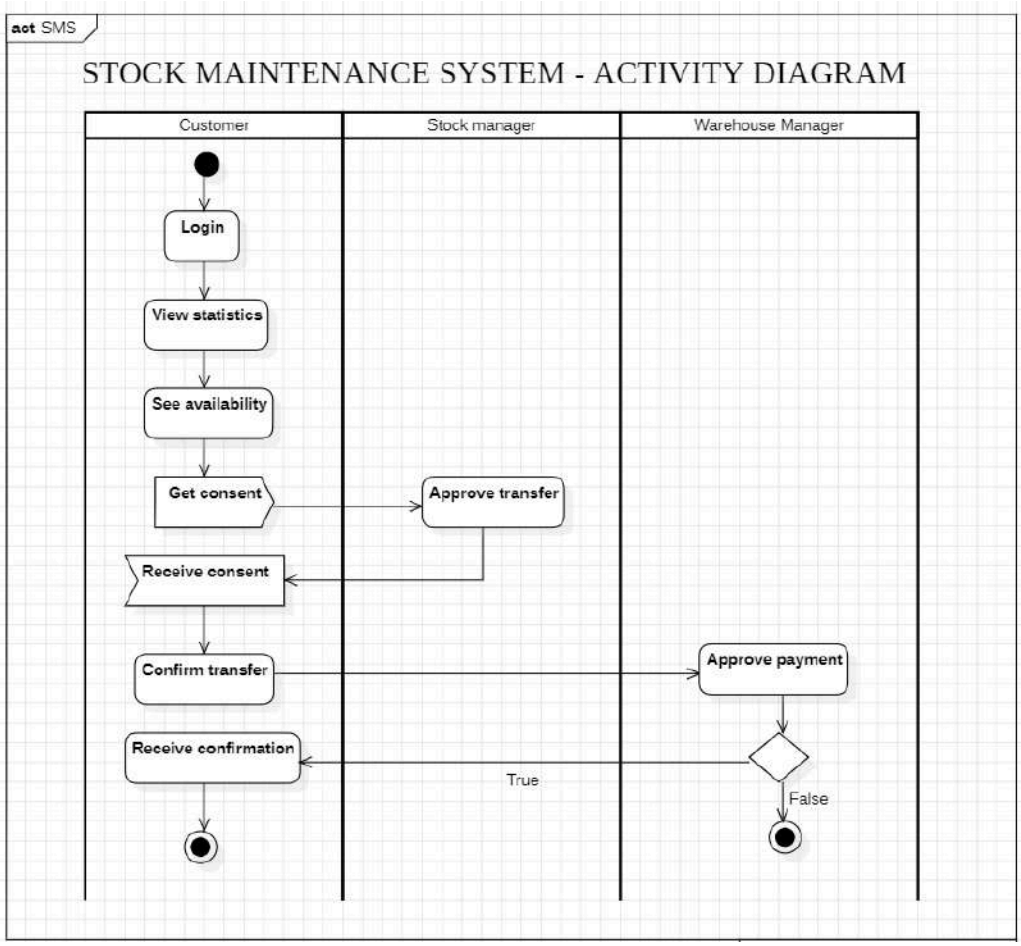
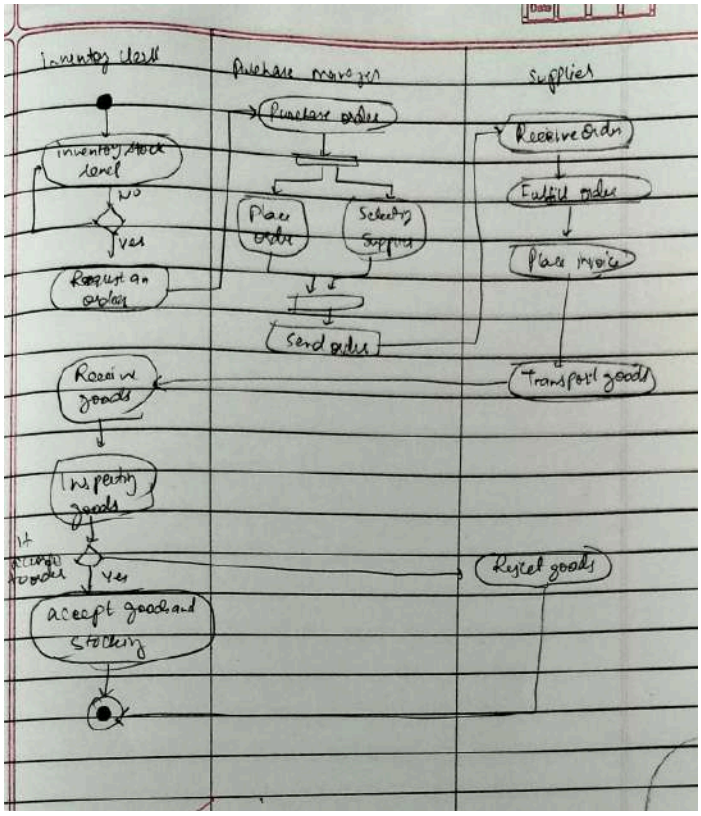



*Figure 4.5: Activity Diagram*

The Swimlane Diagram llustrates the flow of actions involved in a stock transfer process.

1. Customer:
    - The Customer logs into the system and views relevant stock statistics.
    - They check the availability of the stock they wish to transfer.
    - The customer then requests consent for the stock transfer, and once received, confirms the transfer.
    - Finally, they receive confirmation that the stock transfer has been completed successfully.
2. Stock Manager:
    - The Stock Manager reviews and approves the requested stock transfer, ensuring that the process can continue smoothly.
3. Warehouse Manager:
    - The Warehouse Manager oversees the final step, which is approving the payment for the stock transfer before it is finalized.

This flow ensures that each party is responsible for specific tasks during the stock transfer process, from the initial request to the final approval and confirmation.



# 5. PASSPORT AUTOMATION SYSTEM

## 5.1 Problem Statement

The Passport Automation System simplifies the application, processing, and issuance of passports. It includes classes like Applicant, Application, Passport, and Appointment to manage data and processes. The system should support applicants in submitting required documents, scheduling appointments, and tracking application statuses. Relationships such as an applicant having one passport or multiple applications due to renewal or updates should be managed. The system should integrate with external verification services for document validation and criminal record checks.

The objective of this project is to design and implement a comprehensive Passport Automation System to streamline the process of passport issuance, renewal, and verification. The proposed system aims to address the following critical functionalities:

1. Application Management: Provide an online platform for applicants to submit passport applications, update personal details, and upload required documents.

2. Appointment Scheduling: Facilitate real-time scheduling of appointments for document verification, biometrics, and interviews at passport offices.

3. Document Verification and Validation: Automate the verification of submitted documents and cross-check applicant information with government databases to ensure accuracy and authenticity.

4. Processing and Status Tracking: Enable efficient processing of applications and provide real-time status updates to applicants at every stage of the process.

5. Payment and Fee Management: Support secure payment gateways for application fees, generate payment receipts, and maintain a record of transactions.

6. Passport Issuance and Renewal: Streamline the issuance and renewal process by automating workflows, generating unique passport numbers, and printing authorized passports.

7. Security and Fraud Detection: Implement robust security measures to prevent identity theft, detect fraudulent activities, and safeguard sensitive applicant data.

## 5.2 Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for a Passport Automation System (PAS). The system aims to streamline and digitize the process of passport application, verification, and issuance. This document outlines the functional and non-functional requirements, design constraints, and system interfaces to ensure a secure and efficient system.

1.2 Scope

The PAS is designed to manage the end-to-end process of passport services, including application submission, document verification, biometric data capture, and passport issuance. It ensures secure data handling, compliance with government regulations, and user-friendly access for applicants and administrators.

1.3 Definitions, Acronyms, and Abbreviations

- PAS: Passport Automation System
- API: Application Programming Interface
- GUI: Graphical User Interface
- OTP: One-Time Password

1.4 References

- IEEE Standard for Software Requirements Specifications (IEEE Std 830-1998)
- Data Protection and Privacy Regulations

1.5 Overview

This document is structured to provide an overview of the PAS and its requirements. Section 2 highlights the overall description and features, while Section 3 details the functional and non-functional requirements.

2. Overall Description

2.1 Product Perspective

The PAS is a web-based application integrated with government systems to manage passport-related workflows. It provides a seamless interface for applicants, officials, and administrators, ensuring transparency and efficiency throughout the process.

2.2 Product Features

Key features of the PAS include:

- Online application submission
- Document and identity verification
- Biometric data capture and storage
- Application tracking and notifications
- Passport issuance and renewal services

2.3 User Characteristics

The system is designed for the following user groups:

- Applicants: Submit applications, track status, and receive notifications.
- Government Officials: Verify documents, process applications, and manage approvals.
- System Administrators: Oversee operations, manage user roles, and ensure system integrity.

2.4 Constraints

- The system must comply with data protection and privacy regulations.
- High availability is essential to handle nationwide operations.
- Integration with existing government databases and biometric systems is required.

2.5 Assumptions and Dependencies

- Reliable internet connectivity is assumed for online operations.
- The system depends on third-party APIs for identity verification and OTP services.

3. Specific Requirements

3.1 Functional Requirements

- Application Management:
  - Enable users to submit new passport applications and renew existing ones.
  - Provide a user-friendly interface for uploading required documents.
- Verification and Approval:
  - Automate document verification using integrated databases.

    - Allow manual review and approval by officials.
- Biometric Data Processing:
    - Capture and store biometric information securely.
    - Integrate with national identity databases for verification.
- Notifications and Updates:
    - Send real-time notifications to users about application status via email and SMS.
    - Provide tracking options for users to monitor application progress.
- Passport Issuance:
    - Automate the generation of passports post-verification.
    - Ensure secure data transfer to printing facilities.

3.2 Performance Requirements

- The system should handle up to 10,000 concurrent users.
- Response time for application submission must not exceed 5 seconds.
- Daily automated backups must be performed to prevent data loss.

3.3 Interface Requirements

- GUI:
    - Simple and intuitive design for applicants and officials.
    - Role-based access for different user categories.
- API Integration:
    - RESTful APIs for integration with identity verification services and national databases.

3.4 Design Constraints

- The system must use scalable and secure technologies to accommodate increasing user volumes.
- Compatibility with biometric scanning devices is mandatory.

3.5 Non-Functional Attributes

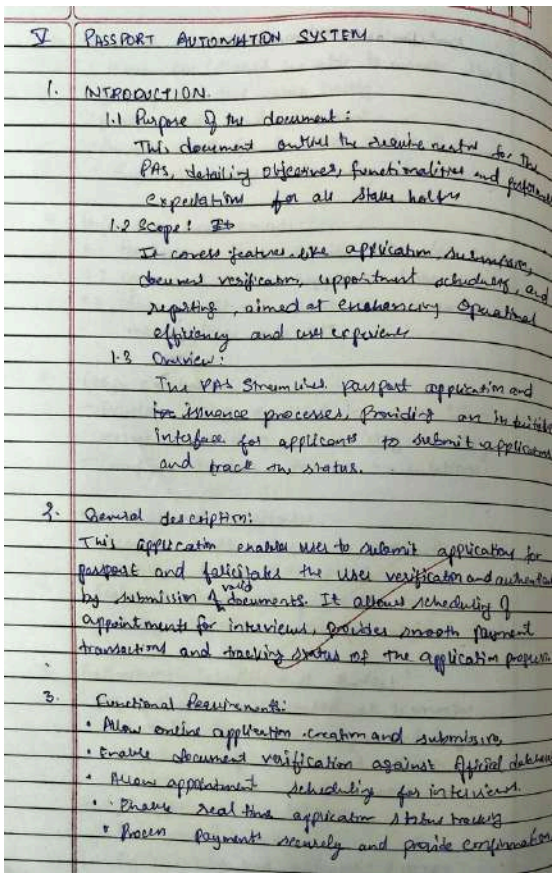
- Reliability: Ensure 99.99% uptime for uninterrupted service.
- Security: Implement end-to-end encryption and secure authentication protocols.
- Usability: Provide an accessible and multilingual interface for diverse user

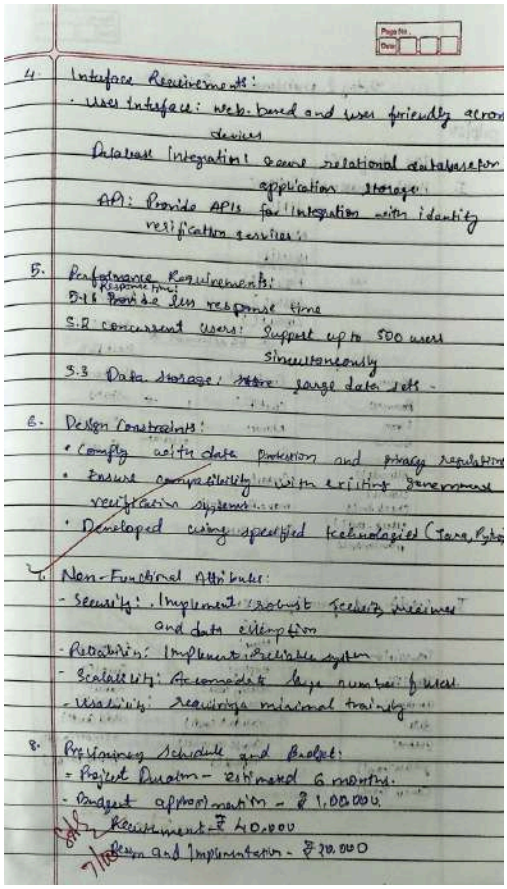
groups.

- Scalability: Support expanding user bases and additional features.
- Maintainability: Modular architecture for ease of updates and maintenance.

3.6 Performance, Schedule, and Budget

- Performance: Support peak loads during high-demand periods without degradation.
- Schedule: Development and deployment to be completed within 18 months.
- Budget: Estimated at $500,000, including infrastructure, development, and compliance costs.



## CLASS DIAGRAM

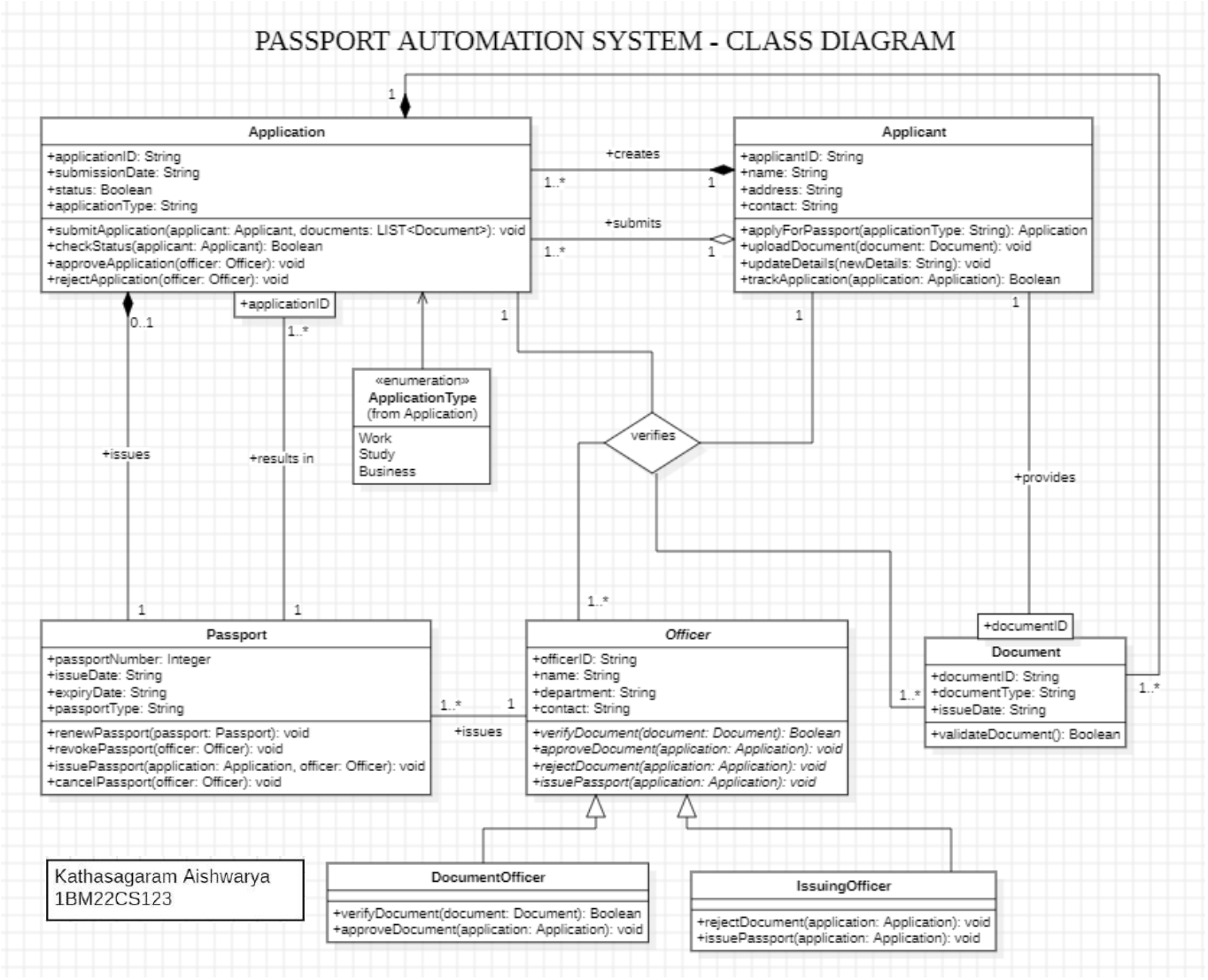
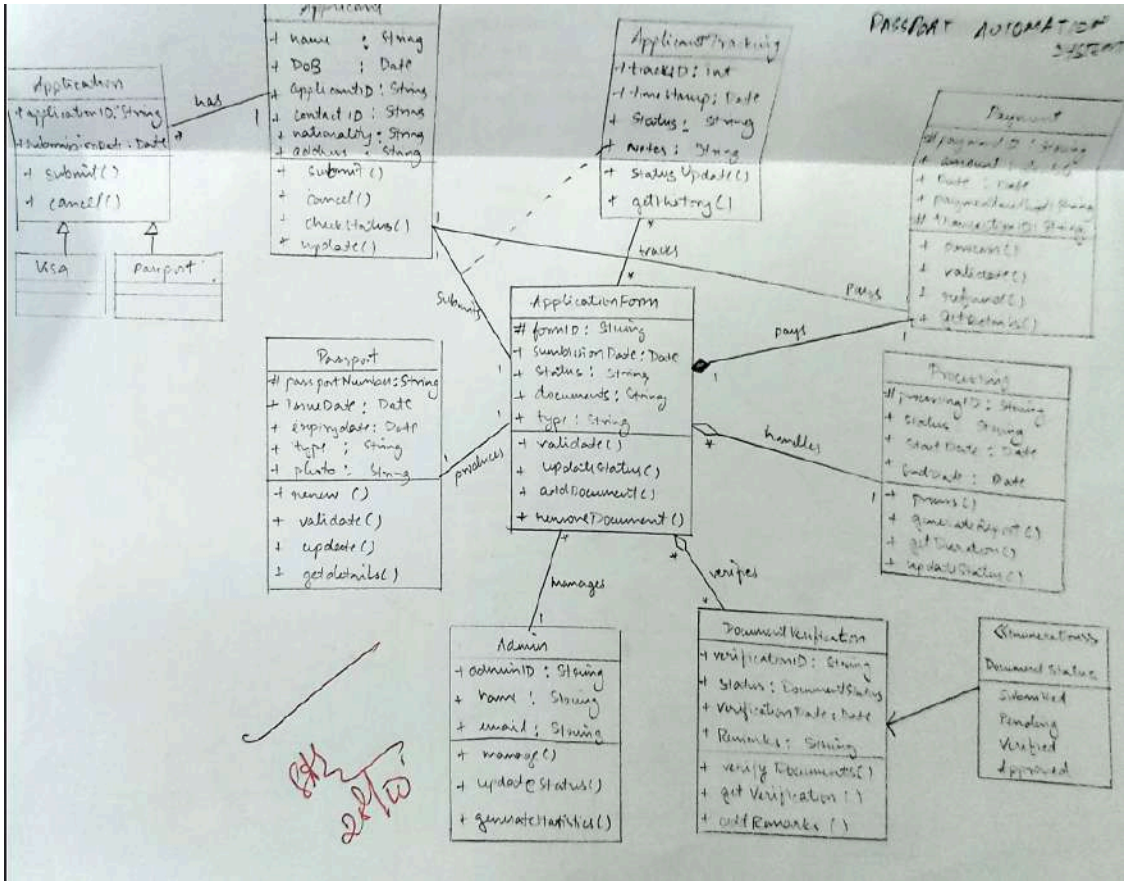



*Figure 5.1: Class Diagram*

This class diagram outlines the components of a passport application system, detailing the relationships and attributes of various entities involved in the process. The Application class represents passport applications and contains information such as application ID, submission date, status, and application type. It also manages the processes of submission, status checking, approval, and rejection. The Applicant class captures the personal details of individuals applying for passports and enables them to submit applications, track statuses, and upload required documents.

The Officer class is divided into two specialized roles: DocumentOfficer, responsible for document verification and approval, and IssuingOfficer, responsible for issuing and revoking passports. The

Document class holds details about each document required for the application process, while the Passport class manages the passport's lifecycle, including its issuance, renewal, and revocation. The associations between these classes show that one applicant can submit multiple applications, each containing several documents. Officers verify and approve documents, reject applications, and issue or revoke passports as needed. The ApplicationType enumeration categorizes applications into Work, Study, or Business types. This system ensures a streamlined flow for applicants, officers, and the management of passport-related tasks.



## STATE DIAGRAM

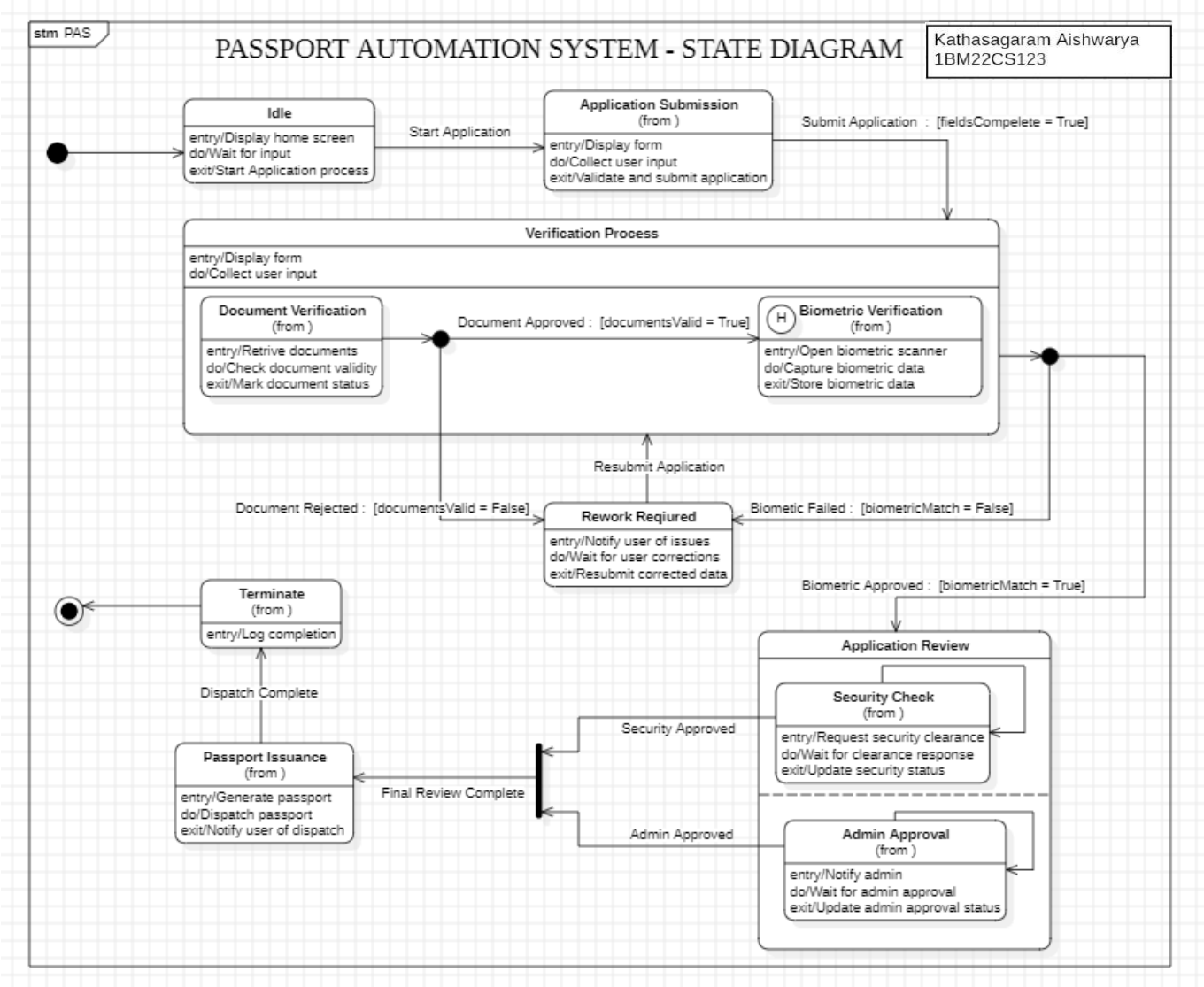
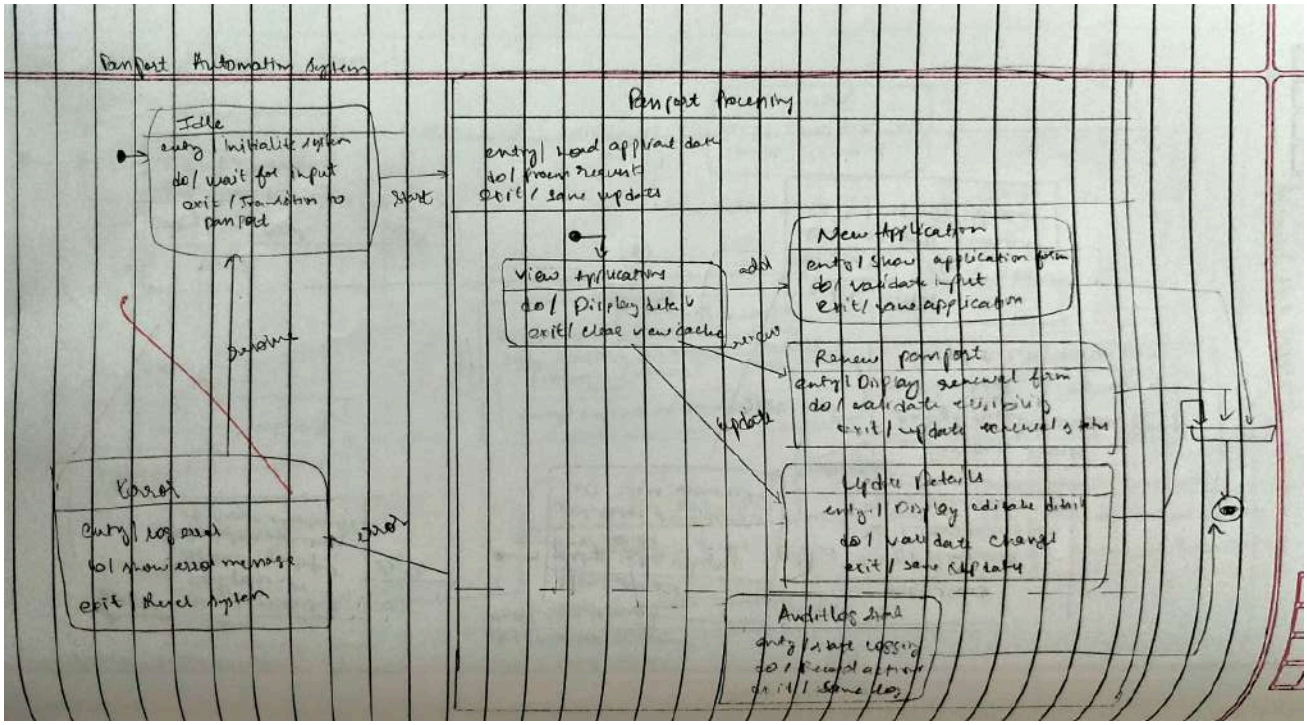



*Figure 5.2: State Diagram*

This state diagram illustrates the flow of the passport application process, from the initial submission to the final passport issuance. It begins with the Idle state, where the system is waiting for user interaction. When the applicant initiates a passport application, the system transitions to the Application Submission state, where the applicant provides necessary information and submits the form. Once the application is submitted, it enters the Verification Process, which involves multiple stages like Document Verification, Biometric Verification, and Security Check.

If documents or biometric data are valid, the system progresses through each verification stage. If any discrepancies or issues are found, such as invalid documents or failed biometric verification, the system

transitions to the Rework Required state, where the applicant is asked to resubmit or correct the information. After passing document and biometric verifications, the application moves to the Security Check, followed by Admin Approval. Once all checks are completed successfully, the system enters the Final Review Complete state, signaling that the application is ready for passport issuance. Finally, the Passport Issuance state generates and dispatches the passport to the applicant, marking the completion of the process in the Dispatch Complete state. The transitions between states are triggered by various actions, including document approval, biometric verification, security clearance, and administrative approval. Activities in each state guide the flow of the process and ensure that all steps are completed before the passport is issued.



## USECASE DIAGRAM

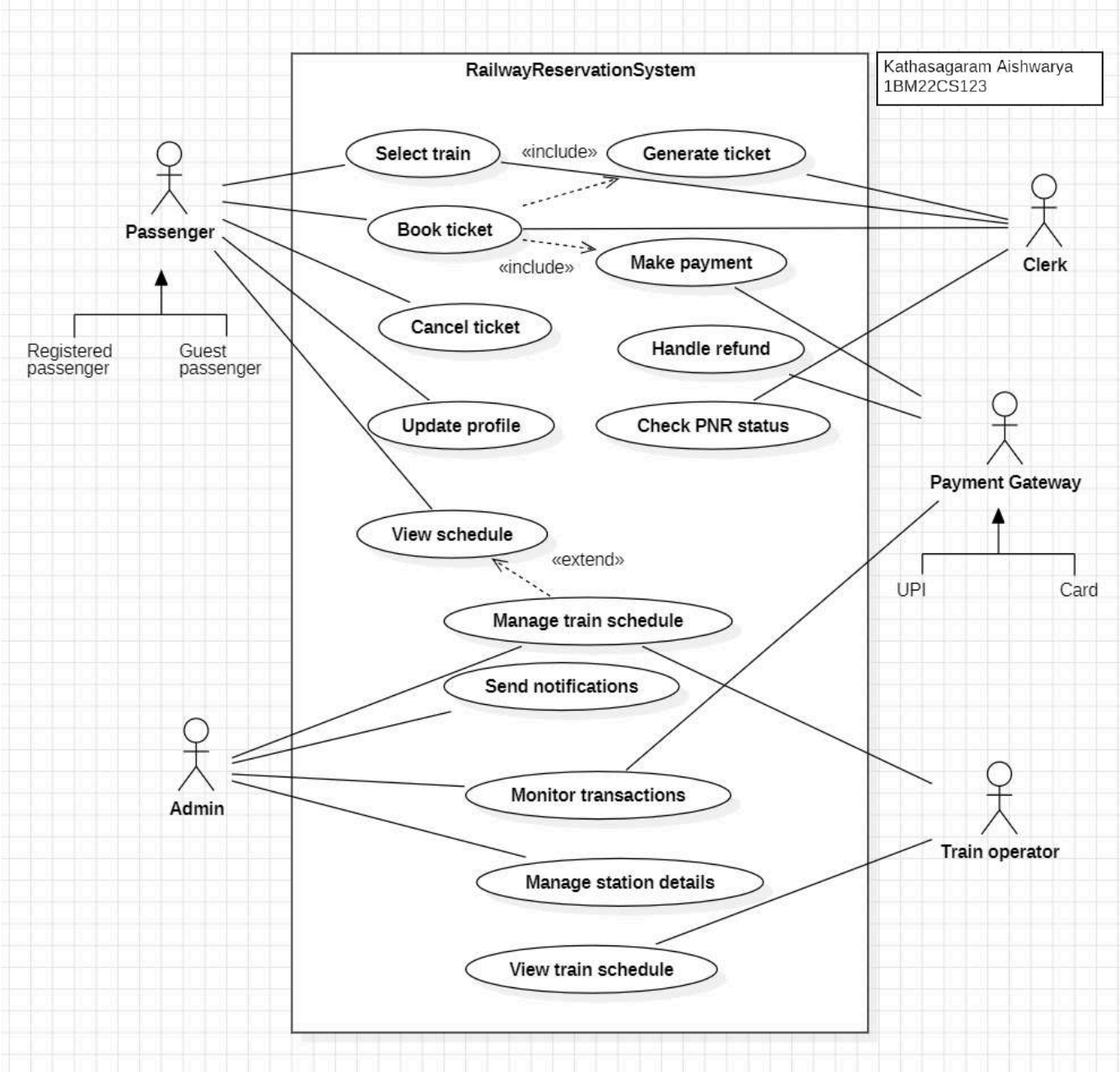


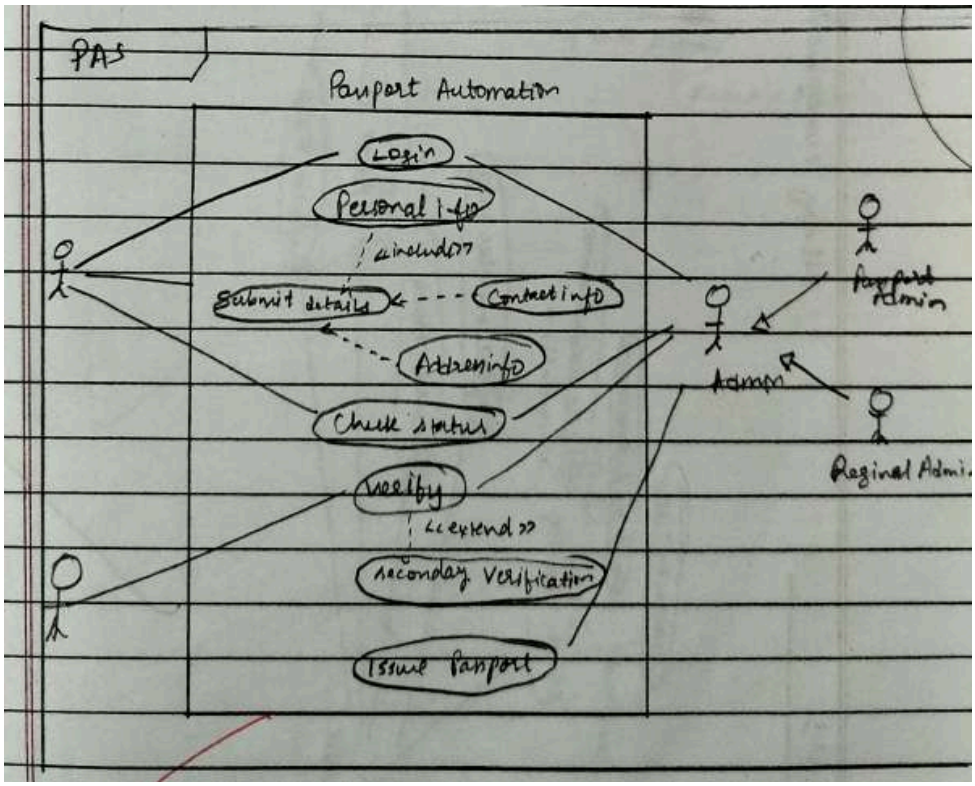

*Figure 5.3: Use Case Diagram*

The Passport Automation System involves multiple actors that interact within the system. The Applicant is the primary user who applies for the passport and provides necessary details for processing. The Passport Admin manages the application process, ensuring the collection of required information and forwarding the application for verification. The Admin has higher-level control over the system, ensuring its overall functionality and security. The Regional Adminhandles regional tasks, including overseeing the verification process in specific geographical areas. The Police may be involved for additional verification,

particularly for background checks and address confirmation.

The system's primary use case is Passport Automation, which includes several steps. The first step, Login, allows users to access the system. Submit Details is a critical use case, where the Applicant provides personal information (such as name, date of birth, and nationality), contact details (like phone number and email), and address information. The Check Status use case enables the Applicant to track the progress of their application. The Verify use case encompasses the verification of details provided by the applicant. This can be extended to include Secondary Verification, a process that may involve background checks or document validation through the police. The final step is Issue Passport, where the passport is issued to the Applicant once all the required checks have been successfully completed.

The relationships between use cases are depicted as Includes and Extends. For example, Submit Details includes Personal Info, Contact Info, and Address Info, meaning these steps are part of the larger submission process. Additionally, Verify may be extended to include Secondary Verification as needed, depending on the application and verification requirements. This system structure helps in understanding the roles of various actors and the flow of information, ensuring that the passport application process is efficient, secure, and well-managed.



## SEQUENCE DIAGRAM

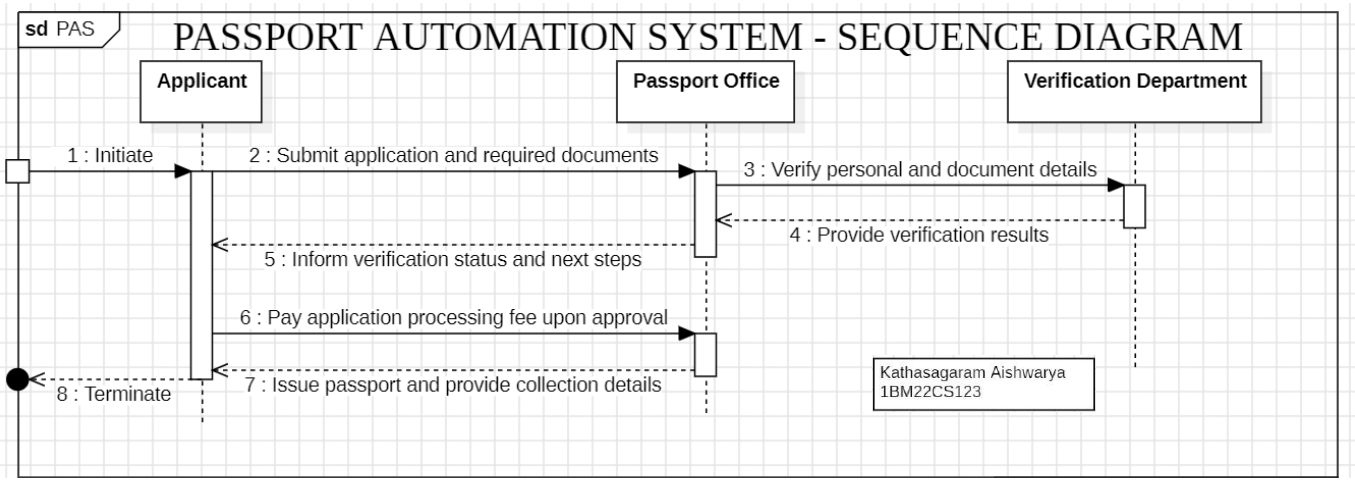
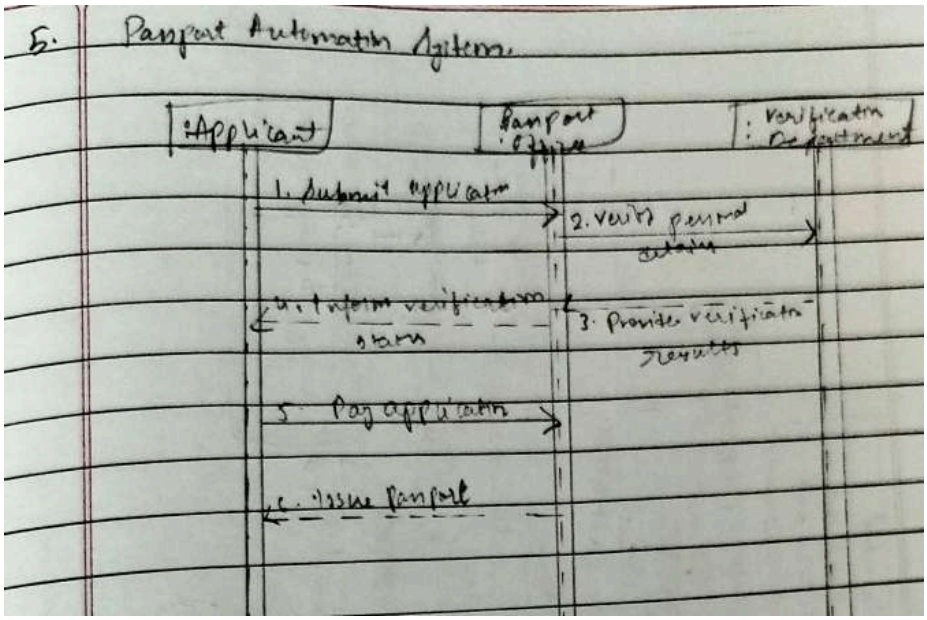



*Figure 5.4: Sequence Diagram*

The passport application process begins when the Applicant initiates their application. After initiating the process, the Applicant submits the required forms and documents to the Passport Office. The Passport Office then sends the application and documents to the Verification Department, which verifies the personal details and documents submitted by the Applicant. Once the verification is complete, the Verification Department sends the results back to the Passport Office, which notifies the Applicant about the verification status and outlines the next steps, such as making the payment or scheduling an appointment.

Once the Applicant receives the notification, they pay the required processing fee if their application is approved. After processing the payment, the Passport Office issues the passport and provides the Applicant with the necessary details to collect it. The application process concludes when the passport is issued and collected by the Applicant, finalizing the procedure. The system ensures that the Applicant follows the steps necessary for a successful application and receives their passport upon completing all requirements.



## ACTIVITY DIAGRAM

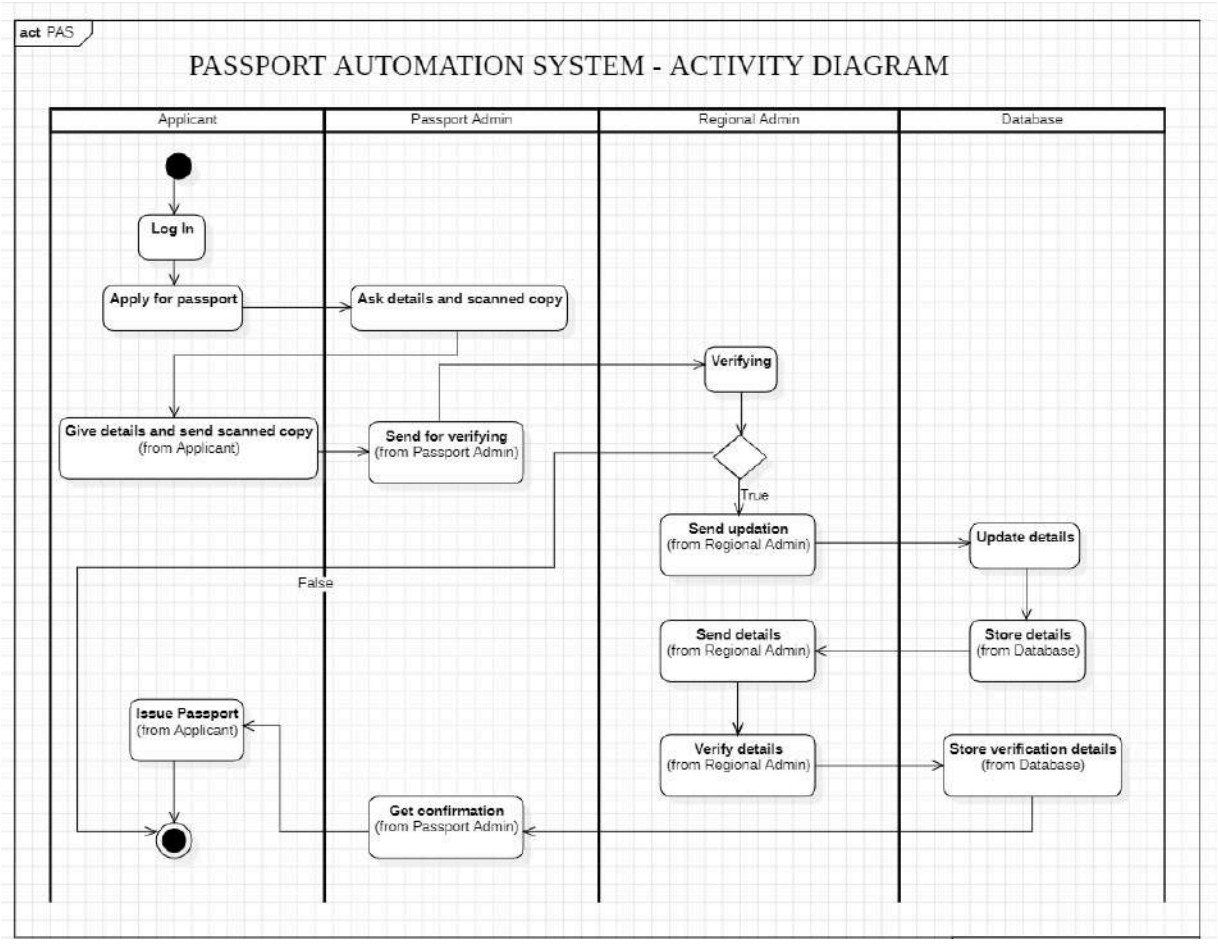



*Figure 5.5: Activity Diagram*

In the passport application process, the Applicant begins by logging into the system and applying for a passport. They then provide personal details and upload scanned copies of necessary documents. Once the application is submitted, the Passport Admin requests the required information and documents from the Applicant. After receiving these, the Passport Admin sends the application and documents to the Regional Admin for verification. The Regional Adminverifies the details and sends an update on the verification status back to the Passport Admin, who then informs the Applicant accordingly.

Simultaneously, the Regional Admin also stores the verified details in the Database, which keeps a record of both the applicant's information and the verification results. The Database ensures that all relevant details, including the verification outcomes, are securely stored for future reference and processing. After the verification process, the Passport Admin proceeds with issuing the passport and notifying the Applicant of the successful completion of their application. This organized flow ensures that the system runs efficiently, keeping the Applicant, Passport Admin, Regional Admin, and Database in sync throughout the process.