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

Laboratory Record

OBJECT-ORIENTED MODELLING

Submitted in partial fulfilment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

JEEVAN A

1BM22CS119

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
September 2024- 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 Jeevan A (1BM22CS119) during the 5th Semester September 2024-January 2025.

Signature of the Faculty in Charge

Sandhya A Kulkarni
Assistant Professor,
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

TABLE OF CONTENTS

CHAPTER - 1

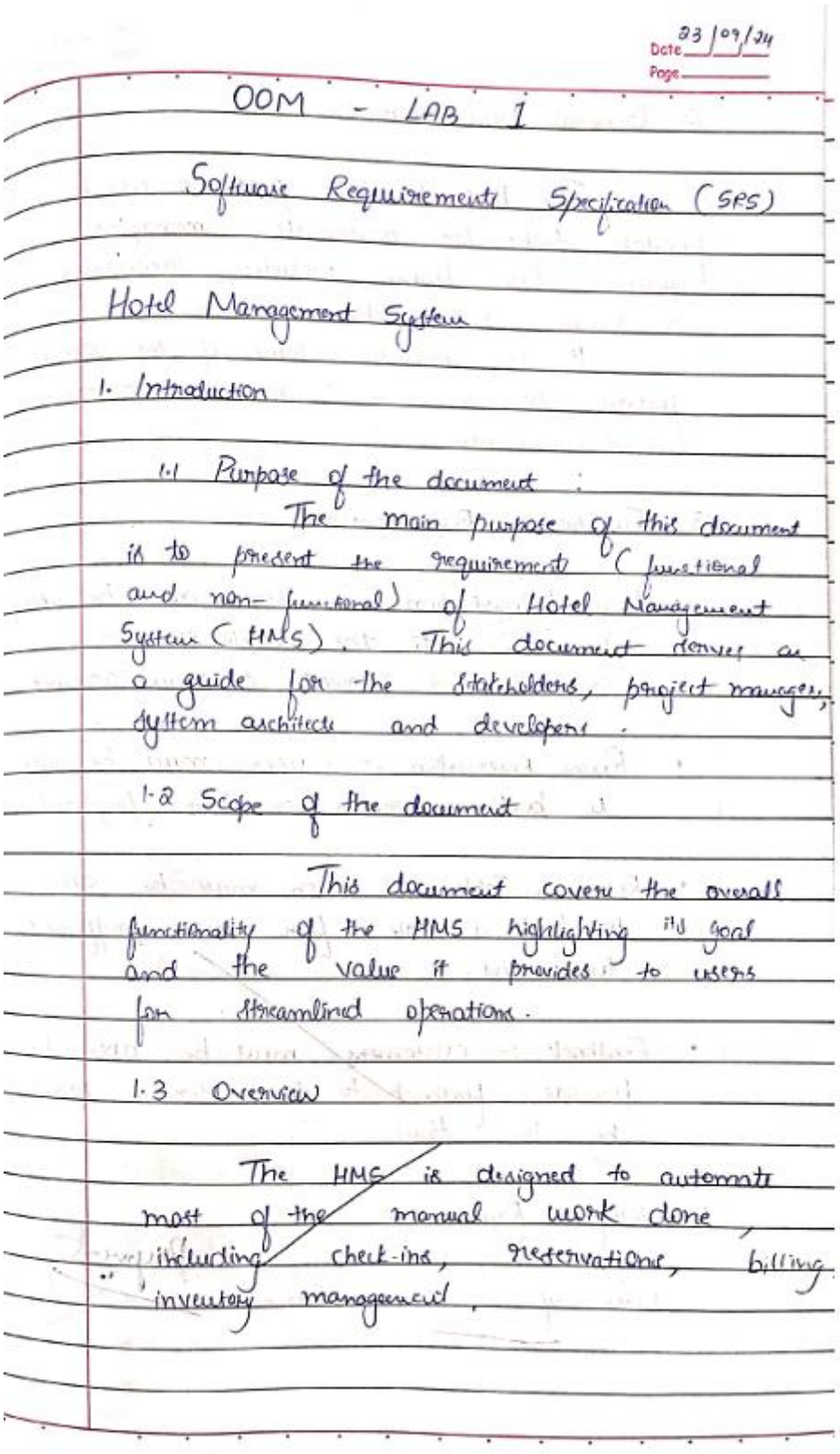
HOTEL MANAGEMENT SYSTEM

PROBLEM STATEMENT

The goal of this project is to develop a comprehensive Hotel Management System to streamline operations and enhance the guest experience. The system will address several key areas to improve overall efficiency and service quality.

- It will improve reservation management with features like online booking, real-time availability checks, secure payment processing, and automated booking confirmations. This will enhance both convenience and operational efficiency.
- system will enhance guest services by simplifying check-in and check-out procedures, efficiently handling guest requests, and providing detailed information about hotel amenities, ensuring a smooth and satisfying experience.
- will support inventory management by monitoring room availability, dynamically managing room rates, assigning rooms, and overseeing maintenance activities to optimize resource utilization.
- system will strengthen financial management through accurate invoice generation, secure payment processing, effective account management, and comprehensive financial reporting to aid financial oversight and decision-making.

SOFTWARE REQUIREMENTS SPECIFICATION



Date 03/09/24
Page

OOM - LAB 1

Software Requirements Specification (SRS)

Hotel Management System

1. Introduction

1.1 Purpose of the document :

The main purpose of this document is to present the requirements (functional and non-functional) of Hotel Management System (HMS). This document serves as a guide for the stakeholders, project managers, system architects and developers.

1.2 Scope of the document

This document covers the overall functionality of the HMS highlighting its goal and the value it provides to users for streamlined operations.

1.3 Overview

The HMS is designed to automate most of the manual work done, including check-ins, reservations, billing, inventory management.

2. General Description :

The HMS provides features which facilitates tasks for receptionists, managers & guests. Key feature includes reserving a room and Table Reservation.

It also provides some of the admin features such as billing, managing reservations and inventory.

3. Functional Requirements

- Login / Registration :- User must be able to login to the application or register and create a new account.
- Room Reservation :- User must be able to book a room on their preffered date
- Reserve a Table :- User must be able to book a table for their preffered time slot.
- Feedback :- Customers must be able to provide feedback to the service provided by the Hotel.

5. Performance Requirement

- System must be able to perform bookings without any crash / server down during rival hours.
- Should have a minimum uptime 95%.
- Able to manage upto 10,000 records concurrently.

6. Design constraints

- React JS (JS framework) for frontend
- MySQL or other RDBMS for DB Management

7. Non-Functional Requirements

- Response Time - System should have a maximum response time of 10ms
- Scalability - Should support future expansion

8. Preliminary schedule and Budget

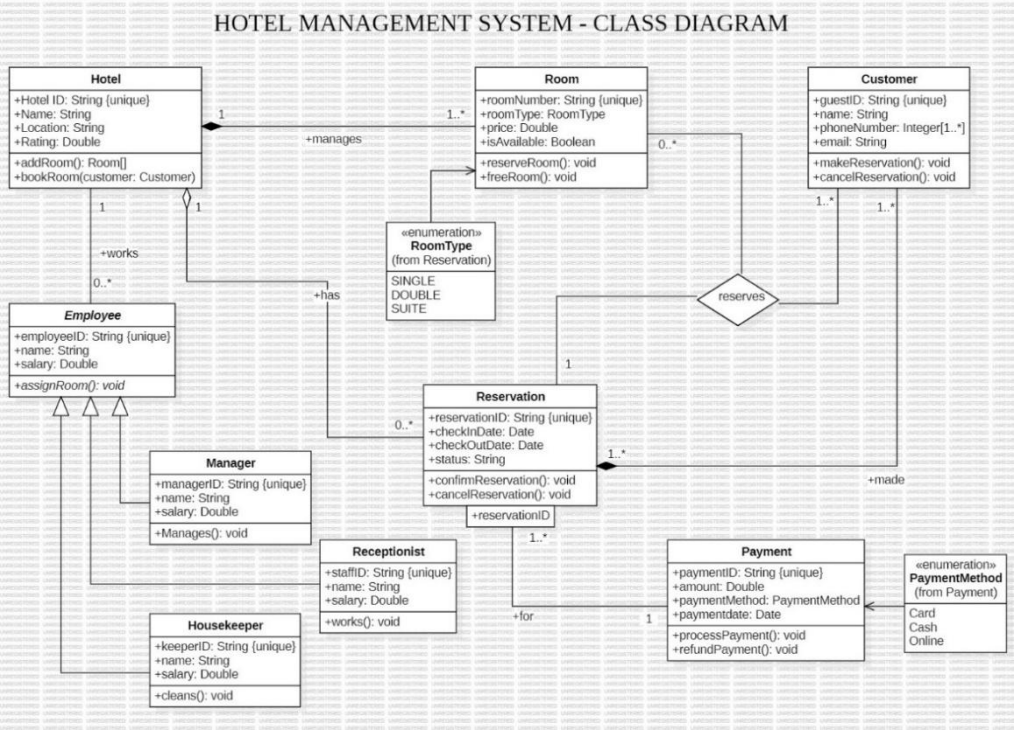
- Estimated Duration - 6 months for development and 4 months for testing
- Project cost - Around $7500 for approximately 10,000 lines of code.

Split up

- Requirement Stage - $ 500
- Development Stage - $ 2500
- Design Stage - $ 3000
- Evaluation Stage - $ 1500

UML DIAGRAMS

1. CLASS DIAGRAM



BRIEF DESCRIPTION:

Classes and Associations

1. **Hotel**:
    - **Attributes**:
        - Hotel ID
        - Name
        - Location
        - Rating
    - **Methods**:
        - addRoom()
        - bookRoom()
2. **Room**:
    - **Attributes**:
        - roomNumber
        - type
        - availability

        - price
    - **Methods**:
        - reserve()
        - free()
3. **Customer**:
    - **Attributes**:
        - customerID
        - name
        - phoneNumber
        - email
    - **Methods**:
        - makeReservation()
        - cancelReservation()
4. **Employee**:
    - **Attributes**:
        - employeeID
        - name
        - salary
    - **Methods**:
        - assignRoom()
        -
5. **Manager** (inherits from Employee):
    - **Methods**:
        - manageHotel()
6. **Reservation**:
    - **Attributes**:
        - reservationID
        - checkInDate
        - checkOutDate
        - status
    - **Methods**:
        - confirm()
        - cancel()
7. **Payment**:
    - **Attributes**:
        - paymentID
        - amount
        - paymentMethod
        - date
    - **Methods**:
        - process()
        - refund()

Associations

1. **Hotel manages Room**:
    - A hotel can manage multiple rooms.
2. **Customer reserves Room**:
    - A customer can reserve multiple rooms.
    - A room can be reserved by multiple customers.
3. **Employee works for Hotel**:
    - Multiple employees work for a single hotel.
4. **Manager manages Hotel**:
    - One manager is responsible for managing one hotel.
5. **Receptionist works for Hotel**:
    - Multiple receptionists work for a single hotel.
6. **Reservation for Room**:
    - Each reservation is specific to one room.

2. USE CASE DIAGRAM

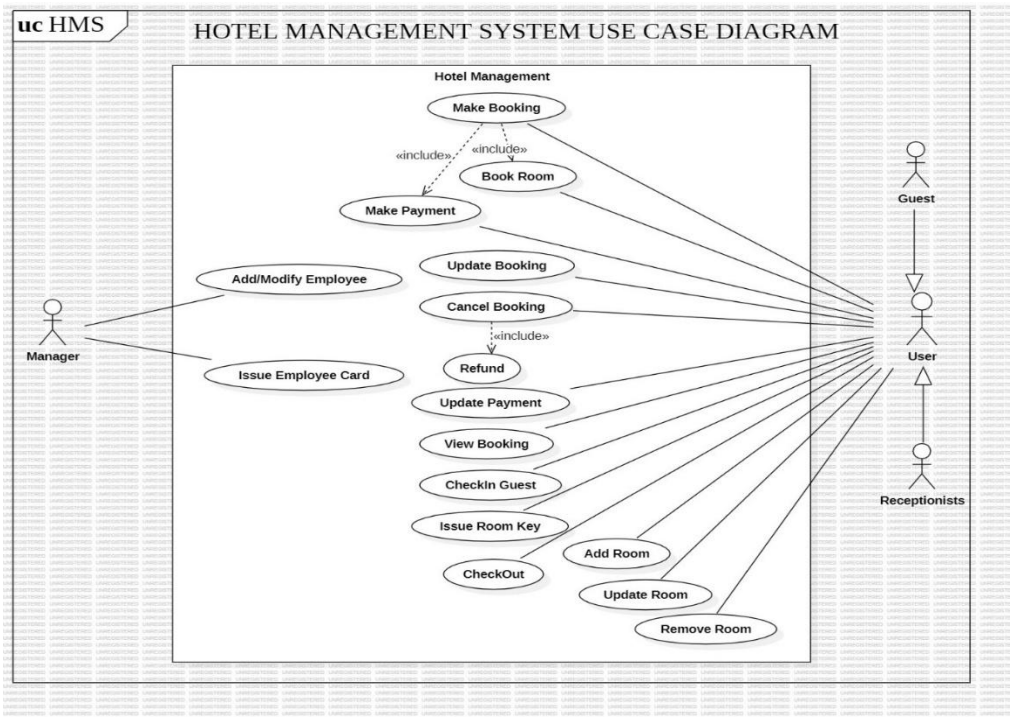


Figure :Advanced Use Case Diagram

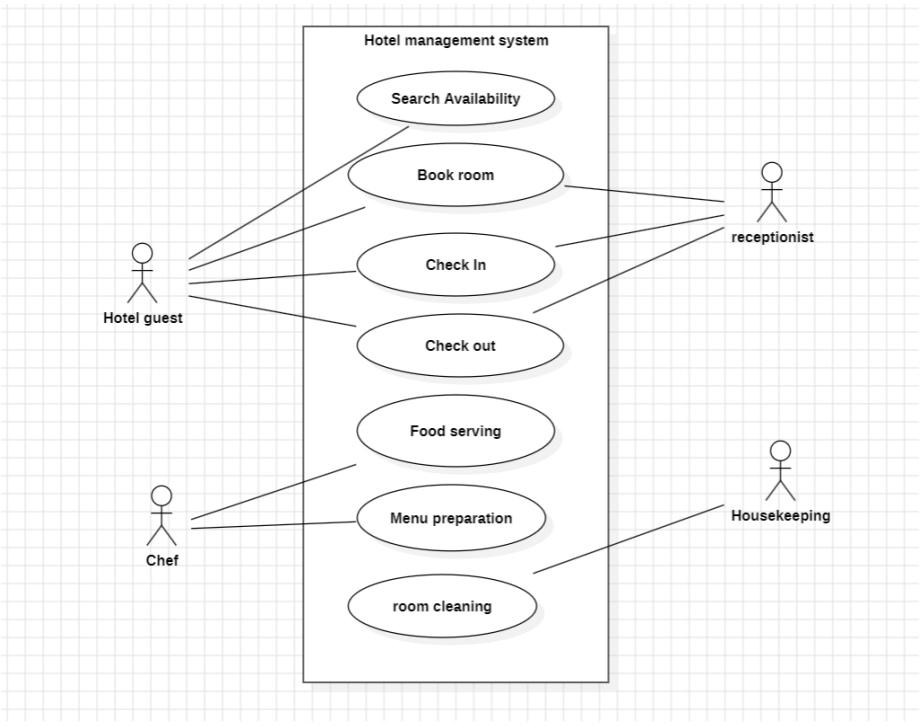


Figure: Simple Use Case Diagram

Actors

1. **Manager**:
    - **Role**: Manages employees and oversees hotel operations.
2. **Guest**:
    - **Role**: Customers who stay at the hotel.
3. **Receptionist**:
    - **Role**: Handles guest interactions, manages check-in/check-out, and assigns rooms.

Use Cases

1. **Hotel Management**:
    - **Description**: Parent use case encompassing all activities related to managing the hotel.
2. **Make Booking**:
    - **Description**: Allows guests to reserve rooms.
    - **Sub Use Case**:
        - **Book Room**: Specifically for booking an individual room.
3. **Make Payment**:
    - **Description**: Enables guests to pay for their room reservations.
4. **Update Booking**:
    - **Description**: Allows guests to modify existing reservations (e.g., changing dates, adding guests).
5. **Cancel Booking**:
    - **Description**: Enables guests to cancel their room reservations.
6. **Refund**:
    - **Description**: Handles the processing of refunds for canceled bookings.
7. **View Booking**:
    - **Description**: Allows guests to view details of their reservations.
8. **Checkin Guest**:
    - **Description**: Receptionists check guests into their rooms.
9. **Checkout**:
    - **Description**: Enables guests to check out of the hotel.
10. **Add Room**:
    - **Description**: Adds new rooms to the hotel's inventory.
11. **Update Room**:
    - **Description**: Modifies details of existing rooms (e.g., type, price).
12. **Remove Room**:
    - **Description**: Removes rooms from the hotel's inventory.
13. **Issue Employee Card**:
    - **Description**: Managers issue ID cards to employees.

Relationships

1. **Includes**:
    - **Example**: "Make Booking" includes "Book Room."
2. **Inheritance**:
    - **Example**: "Guest" inherits from "User.

3. STATE DIAGRAM

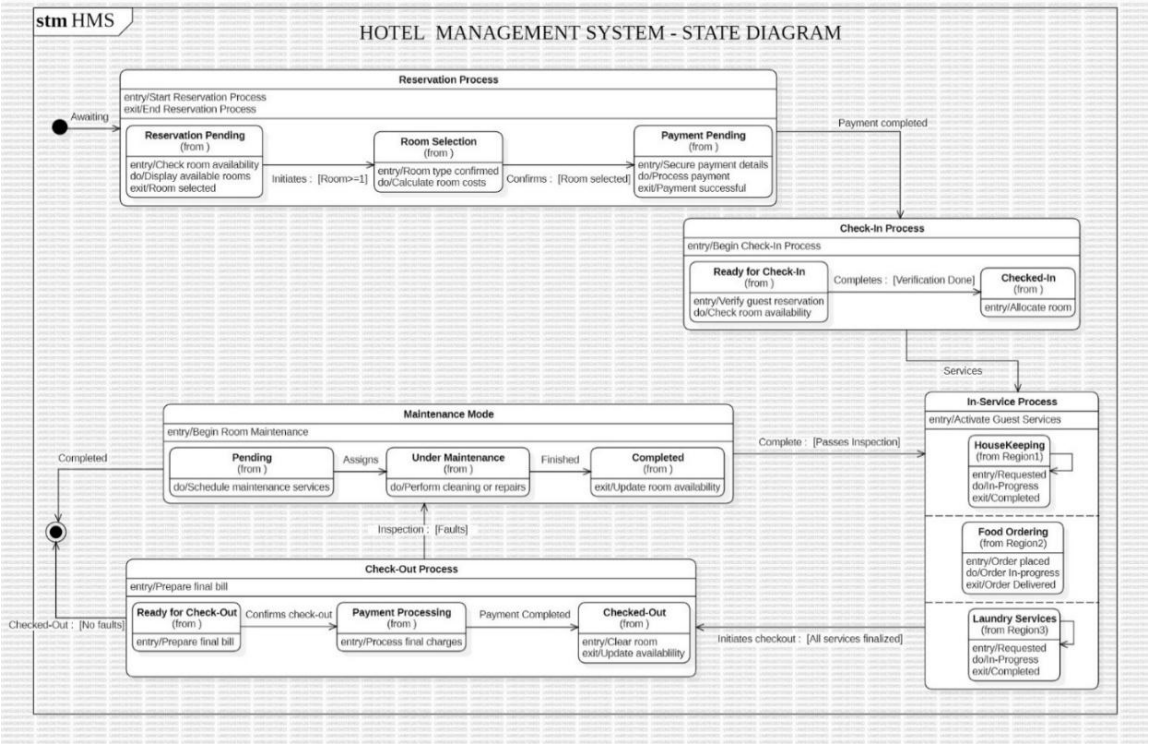


Figure: Advanced State Diagram

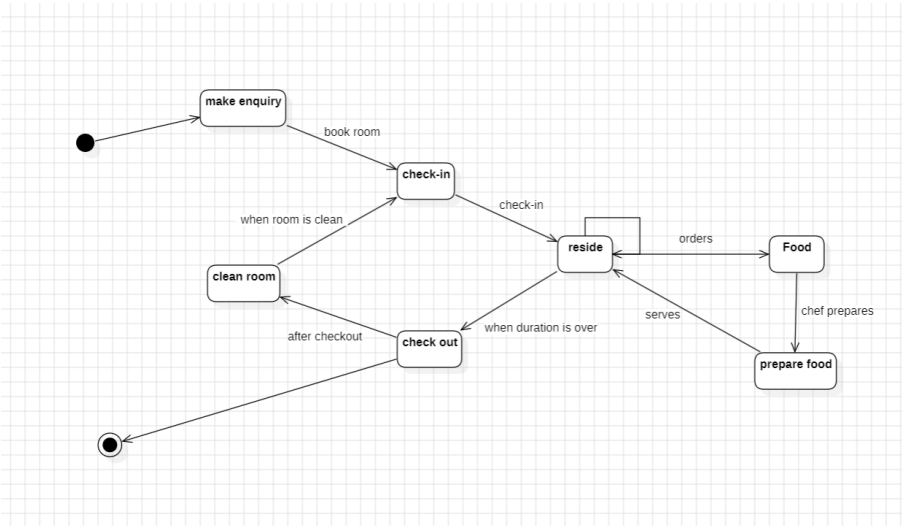


Figure: Simple State Diagram

States:

Primary Booking Process States

1. **Awaiting Reservation**:
    - **Description**: Initial state where the system is ready to accept a new reservation request.
2. **Reservation Pending**:
    - **Description**: System processes the reservation request, checking room availability.
3. **Room Selection**:
    - **Description**: Available rooms are displayed to the customer for selection.
4. **Payment Pending**:
    - **Description**: Customer is prompted to make a payment for the selected room.
5. **Payment Completed**:
    - **Description**: Payment is processed successfully, confirming the reservation.
6. **Checked-In**:
    - **Description**: Customer has checked into the hotel and the room is assigned.

Transitions for Primary Booking Process

1. **To Reservation Pending**:
    - **Trigger**: Customer initiates a reservation request.
2. **To Room Selection**:
    - **Trigger**: After checking room availability in the Reservation Pending state.
3. **To Payment Pending**:
    - **Trigger**: When the customer selects a room.
4. **To Payment Completed**:
    - **Trigger**: When the payment is successfully processed.
5. **To Checked-In**:
    - **Trigger**: When the customer arrives at the hotel and checks in.

Advanced In-Service Process (Nested State Diagram)

Nested States within In-Service Process

1. **Housekeeping**:
    - **Description**: Manages housekeeping services such as cleaning and maintenance.
    - **Sub-States**:

- **Requested**: Service is requested by the guest.
- **In-Progress**: Service is being processed.
- **Completed**: Service is completed.

2. **Food Ordering**:
    - **Description**: Handles food and beverage orders.
    - **Sub-States**:
        - **Requested**: Order is requested by the guest.
        - **In-Progress**: Order is being prepared/delivered.
        - **Completed**: Order is delivered and service is completed.
3. **Laundry Services**:
    - **Description**: Manages laundry requests.
    - **Sub-States**:
        - **Requested**: Laundry service is requested by the guest.
        - **In-Progress**: Laundry is being processed.
        - **Completed**: Laundry service is completed.

4. SEQUENCE DIAGRAM

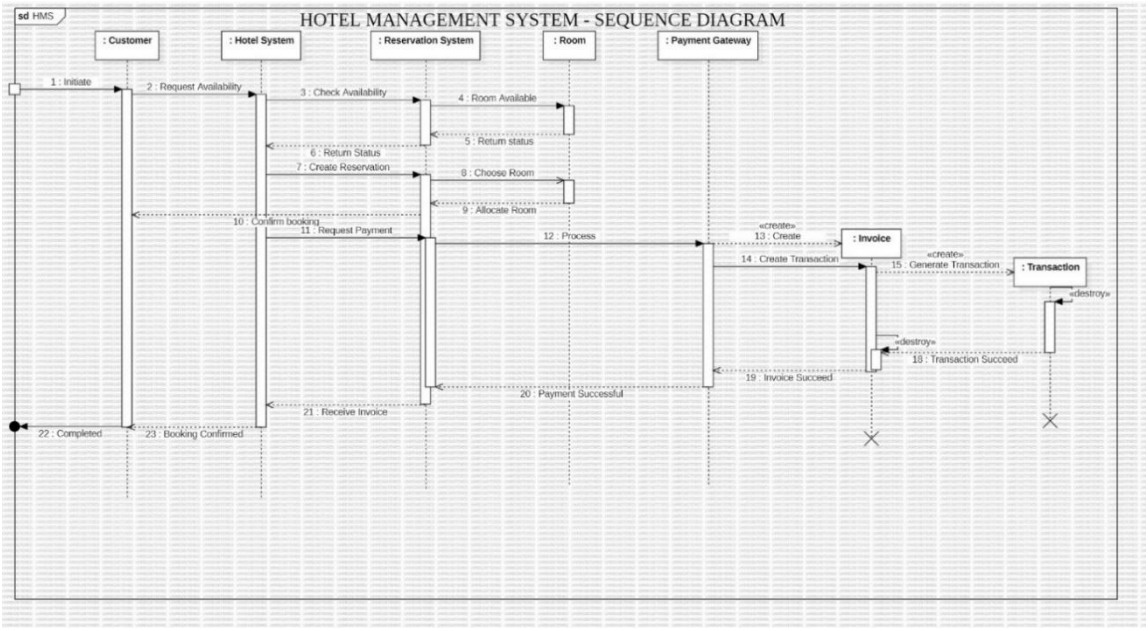


Figure: Advanced Sequence Diagram

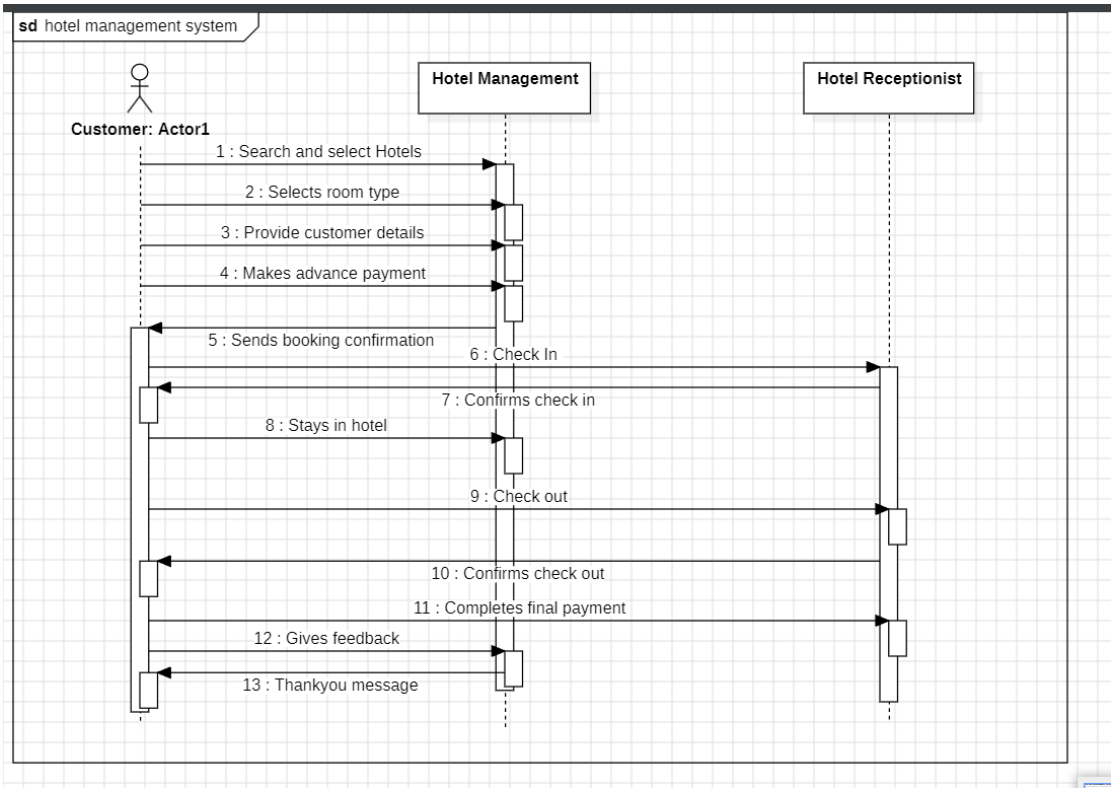


Figure: Simple Sequence Diagram

SCENARIO 1:

1. **Hotel System:** Receives the request and checks room availability in the Reservation System.
2. **Reservation System:** Checks room availability based on the requested dates and returns the status to the Hotel System.
3. **Hotel System:** Receives the availability status and sends it back to the Customer.
4. **Customer:** Reviews the available rooms and chooses a preferred room.
5. **Customer:** Sends the chosen room selection to the Hotel System.
6. **Hotel System:** Creates a reservation for the chosen room in the Reservation System.
7. **Reservation System:** Confirms the reservation and allocates the room.
8. **Hotel System:** Informs the Customer about the confirmed reservation.
9. **Hotel System:** Requests payment details from the Customer.
10. **Customer:** Provides payment details to the Hotel System.

11. **Hotel System:** Sends the payment information to the Payment Gateway.
12. **Payment Gateway:** Processes the payment and sends an invoice to the Hotel System.
13. **Hotel System:** Receives the invoice and creates a transaction record.
14. **Payment Gateway:** Generates a transaction receipt and sends it to the Customer.
15. **Customer:** Receives the invoice and payment success notification.
16. **Customer:** Completes the booking process.

Scenario 2: Customer Cancels a Reservation

1. Customer: Sends a request to the Hotel System to cancel an existing reservation.
2. Hotel System: Receives the cancellation request and checks the reservation status in the Reservation System.
3. Reservation System: Retrieves the reservation details and updates the status to "Cancelled."
4. Hotel System: Informs the Customer that the reservation has been canceled.
5. Reservation System: Releases the allocated room back to the pool of available rooms.

Hotel System: May initiate a refund process if applicable.

5. ACTIVITY DIAGRAM

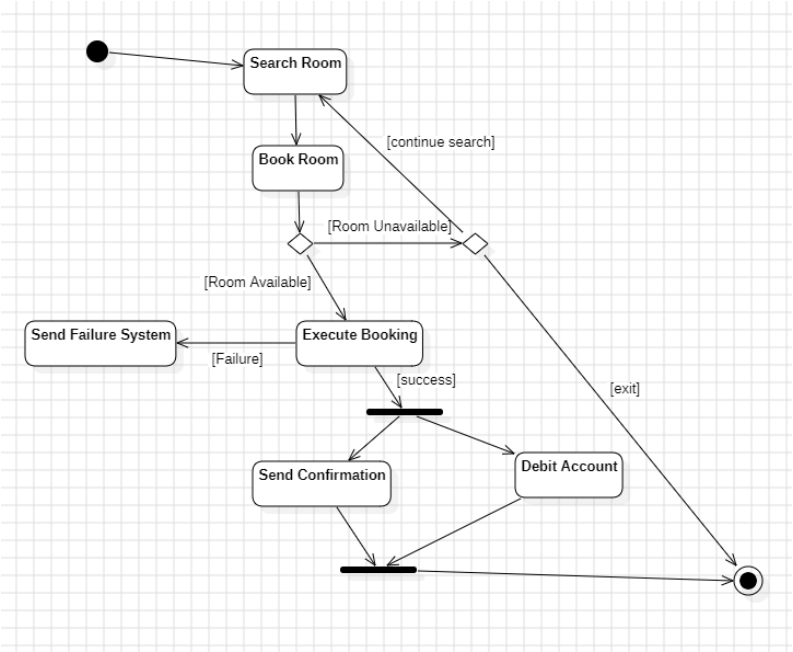


Figure: Simple State Diagram

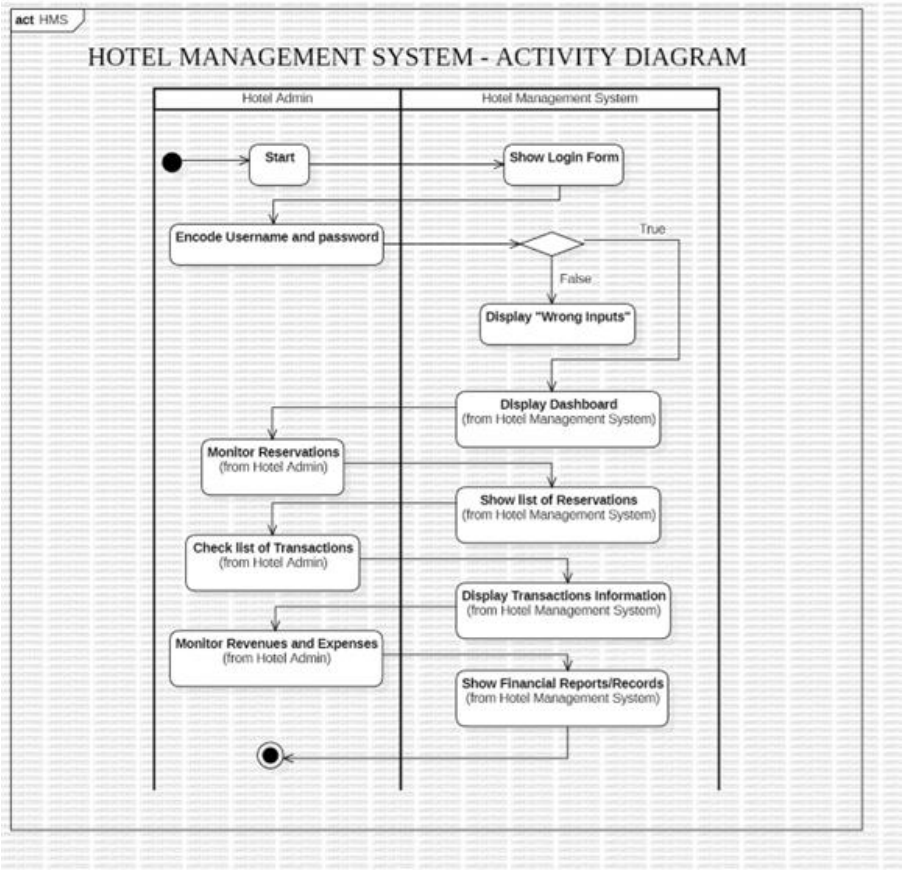


Figure: Advanced Activity Diagram

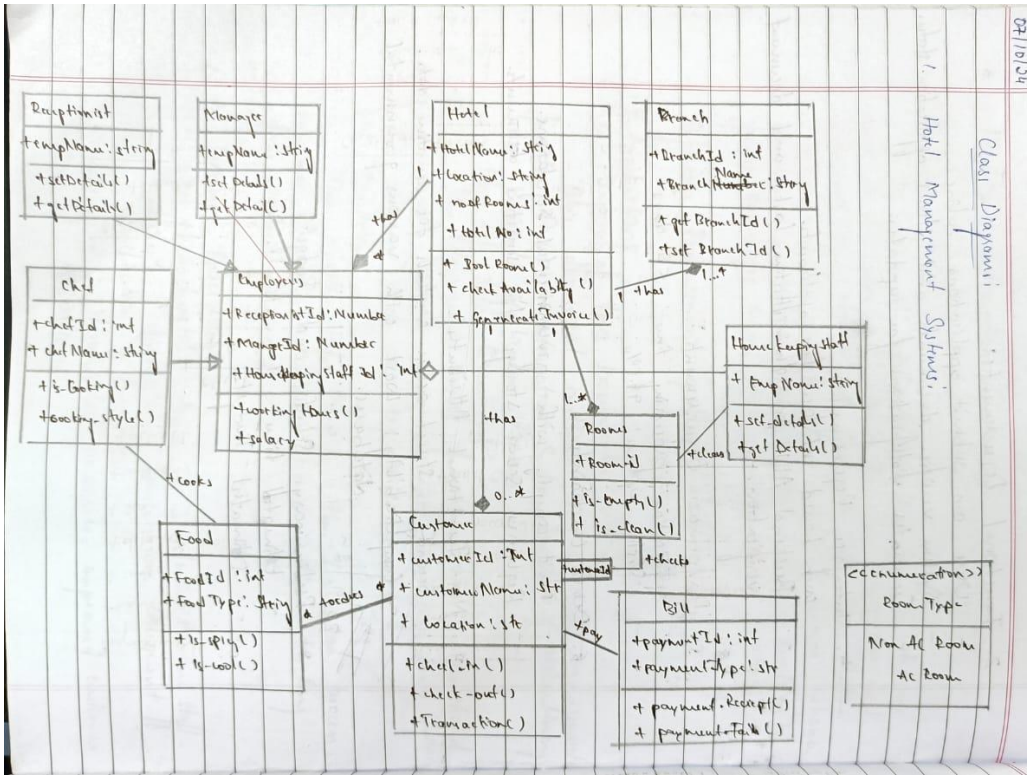
Hotel Admin

- **Start**: Initial state of the system.
- **Encode Username and Password**: Admin enters login credentials.
- **Monitor Reservations**: Admin checks reservation list.
- **Check list of Transactions**: Admin reviews transaction records.
- **Monitor Revenues and Expenses**: Admin analyzes financial performance.

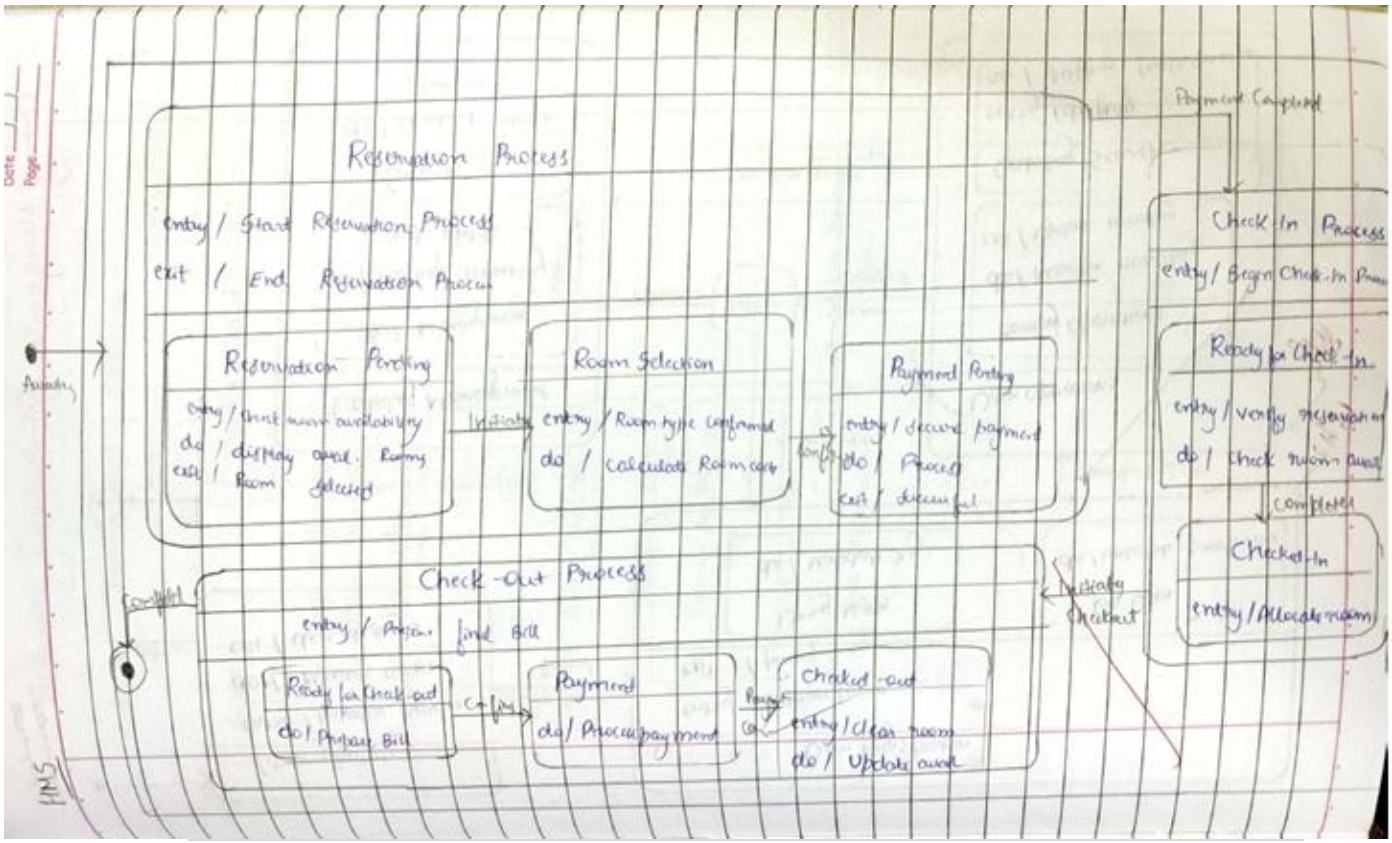
Hotel Management System

- **Show Login Form**: System displays login form to admin.
- **Display "Wrong Inputs"**: System shows error for incorrect credentials.
- **Display Dashboard**: System shows main dashboard with key info.
- **Show list of Reservations**: System displays reservation list.
- **Display Transactions Information**: System shows transaction details.
- **Show Financial Reports/Records**: System generates and displays financial reports.

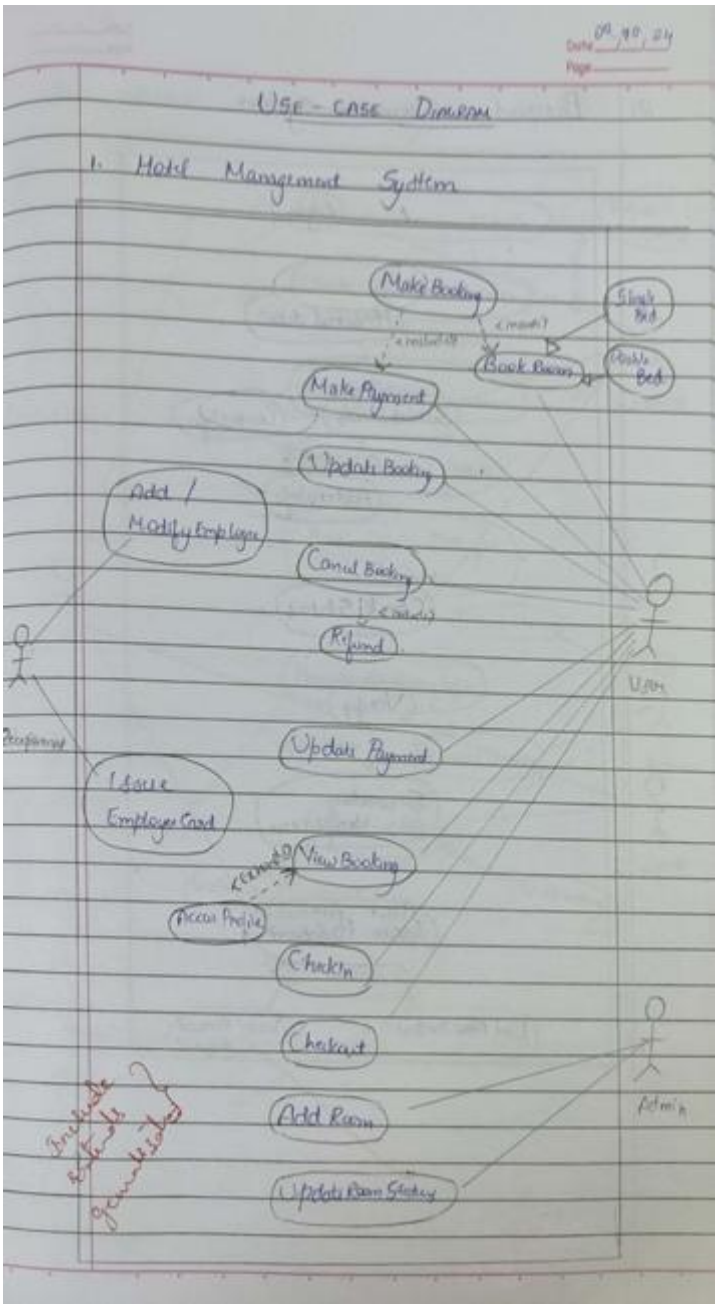
1. CLASS DIAGRAM



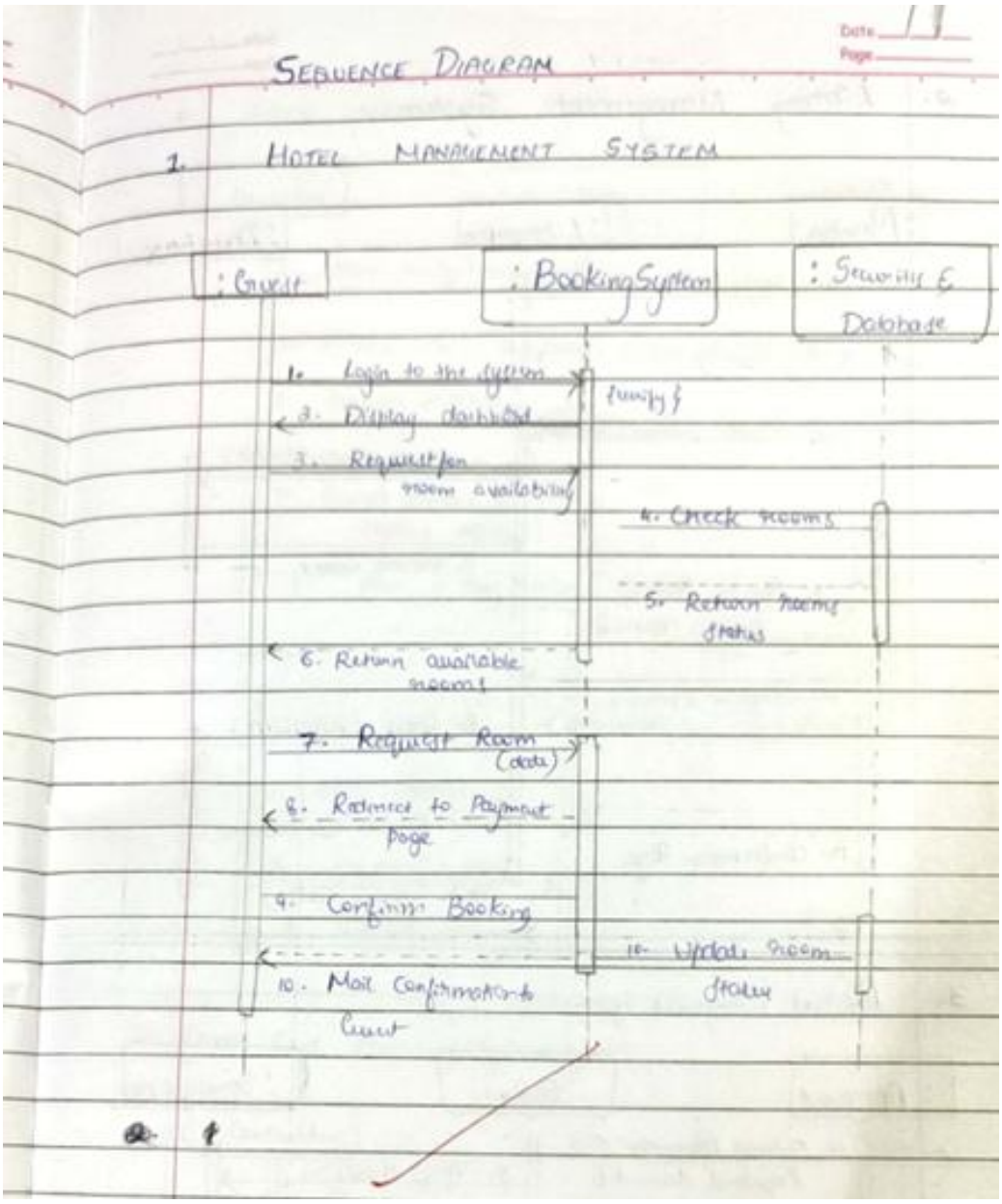
2. STATE DIAGRAM



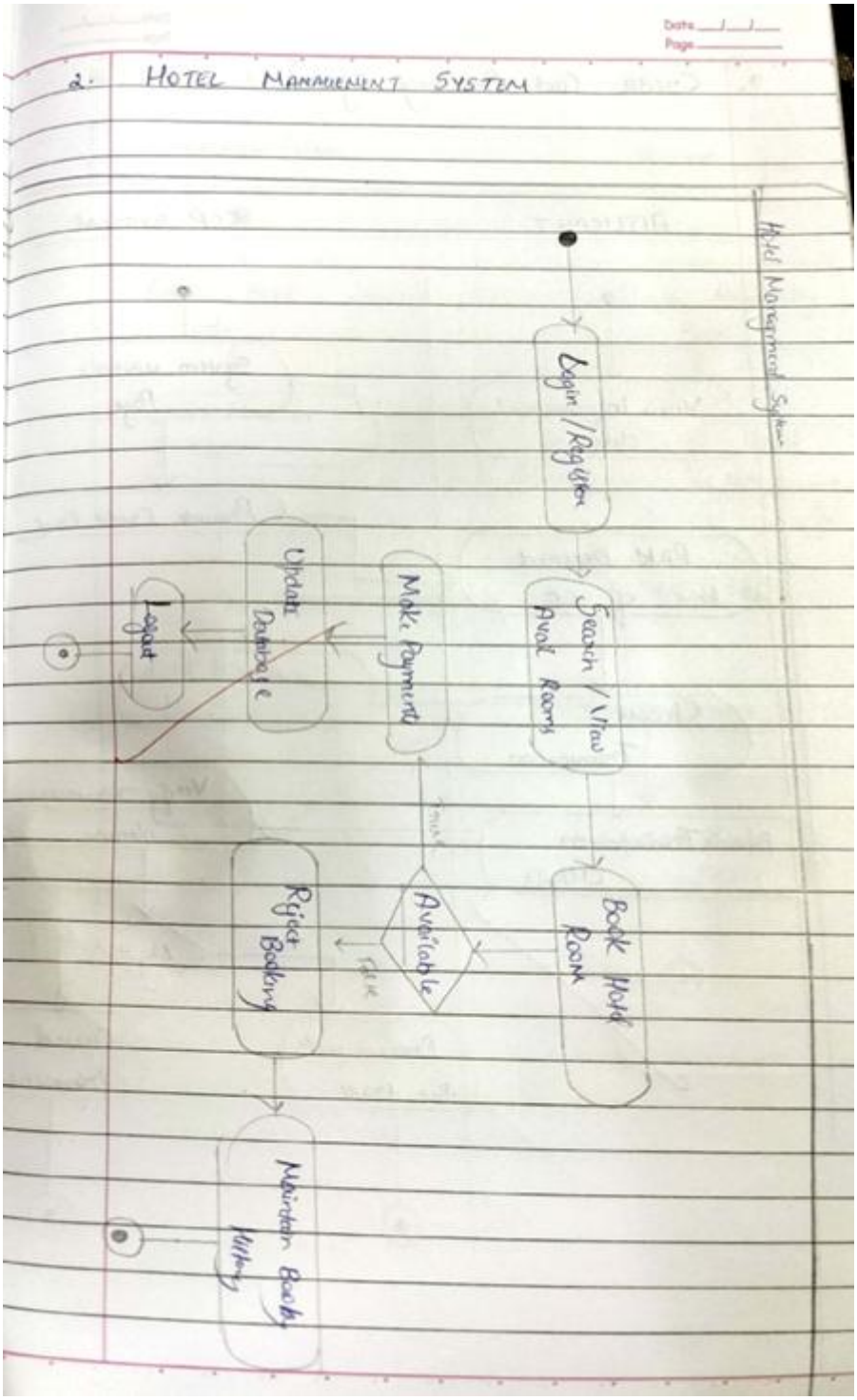
3. USE CASE DIAGRAM



4. SEQUENCE DIAGRAM



5.ACTIVITY DIAGRAM



CHAPTER – 2

CREDIT CARD PROCESSING SYSTEM

PROBLEM STATEMENT

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 system aims to address several key functionalities to enhance transaction processing and security.

- The system will facilitate real-time transaction processing, including authorization, capture, and settlement of credit card payments, ensuring fast and reliable handling of transactions.
- It will integrate advanced fraud detection algorithms to monitor transactions for suspicious activities, implement security protocols, and effectively reduce fraudulent activities.
- Account management tools will be provided, enabling users to update cardholder information, view transaction histories, and generate account statements for easier management.
- Efficient dispute resolution will be supported by enabling detailed tracking of transaction records and providing automated workflows for handling chargebacks and disputes.

SOFTWARE REQUIREMENTS SPECIFICATION

Credit Card Processing System

1. Introduction

1.1 Purpose of the document :

To outline the functional and non-functional requirements for a credit card processing system.

1.2 Scope of the document :

The system will cater to merchants, customers and financial institutions, providing secure, fast, and reliable payment solutions. It will handle transaction processing, fraud detection and data encryption.

1.3 Overview

This system is designed to facilitate secure & efficient processing of credit card transaction for various business.

2. General Description

Transaction authorization : Validate credit card information & authorize

Transaction Capture : Process and record completed details, proof of origin.

Fraud & reporting : Generate reports and transaction data, including sales & refunds.

3. Functional Requirements

- System shall allow merchants to input and process credit card information.
- Authorize transaction - verifies details.
- Should issue refunds for incomplete transactions
- Fraud detection - identifying suspicious transactions.

4. Interface Requirement :-

- API integration - Integrating system with e-commerce platforms and payment gateway.

5. Design Constraints :-

- Should handle multiple transactions
- Supports multiple currencies and international payments.

6. Performance Requirements :-

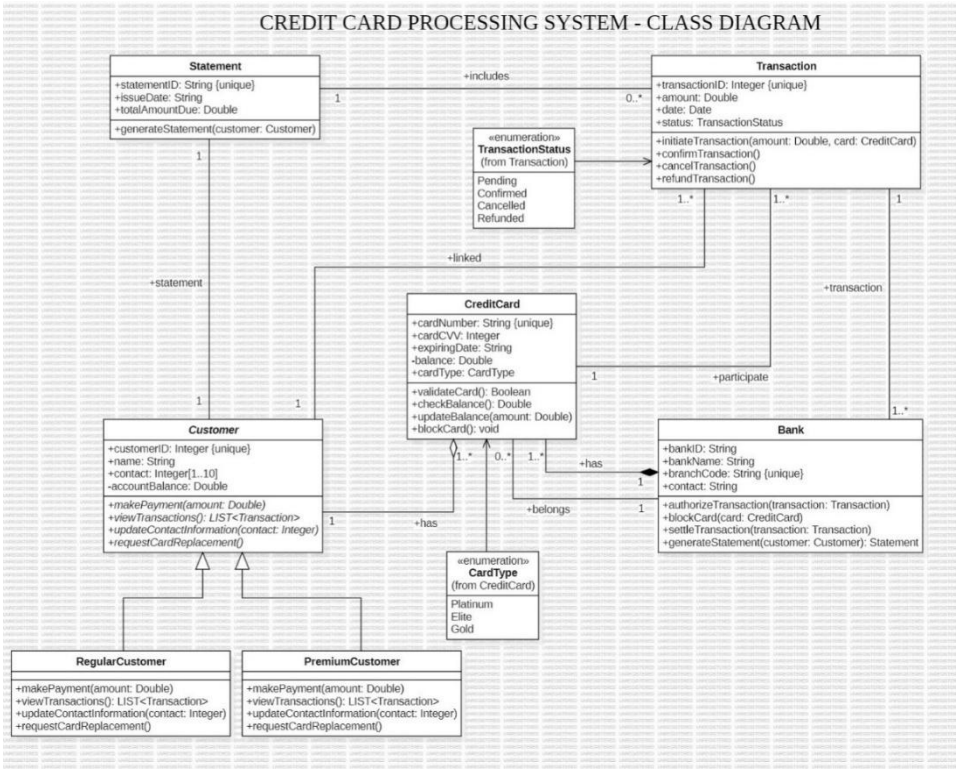
- Processes transactions in less than two seconds.

7. Non-Functional Requirements :-

- High reliability with 99.99% uptime
- Robust encryption standards for data protection.

UML DIAGRAMS

1. CLASS DIAGRAM



Key Classes and their Relationships:

1. **Customer:**
    - Attributes: customerID, name, contact, accountBalance
    - Methods: makePayment, viewTransactions, updateContactInformation, requestCardReplacement
    - Has-a relationship with CreditCard
    - Is-a relationship with RegularCustomer and PremiumCustomer
2. **RegularCustomer:**
    - Inherits from Customer
    - Methods: makePayment, viewTransactions, updateContactInformation, requestCardReplacement
3. **PremiumCustomer:**
    - Inherits from Customer

- Methods: makePayment, viewTransactions, updateContactInformation, requestCardReplacement

4. **CreditCard:**
   - Attributes: cardNumber, cardCVV, expiringDate, balance, cardType
   - Methods: validateCard, checkBalance, updateBalance, blockCard
   - Has-a relationship with CardType
   - Participates in transactions with Bank
5. **Transaction:**
   - Attributes: transactionID, amount, date, status
   - Methods: initiateTransaction, confirmTransaction, cancelTransaction, refundTransaction
   - Status is an enumeration with values: Pending, Confirmed, Cancelled, Refunded
6. **Statement:**
   - Attributes: statementID, issueDate, totalAmountDue
   - Methods: generateStatement
7. **Bank:**
   - Attributes: bankID, bankName, branchCode, contact
   - Methods: authorizeTransaction, settleTransaction, generateStatement
8. **CardType:**
   - Enumeration: Platinum, Elite, Gold, Regular

Relationships:

- **Customer has-a CreditCard:** A customer can have one or more credit cards.
- **RegularCustomer and PremiumCustomer inherit from Customer:** They are specialized types of customers.
- **CreditCard participates in Transaction:** Credit cards are used in transactions.
- **Transaction includes Statement:** A transaction can be included in a statement.
- **Bank has-a CreditCard:** A bank issues credit cards.
- **Bank authorizes and settles Transactions:** The bank processes transactions

2. STATE DIAGRAM

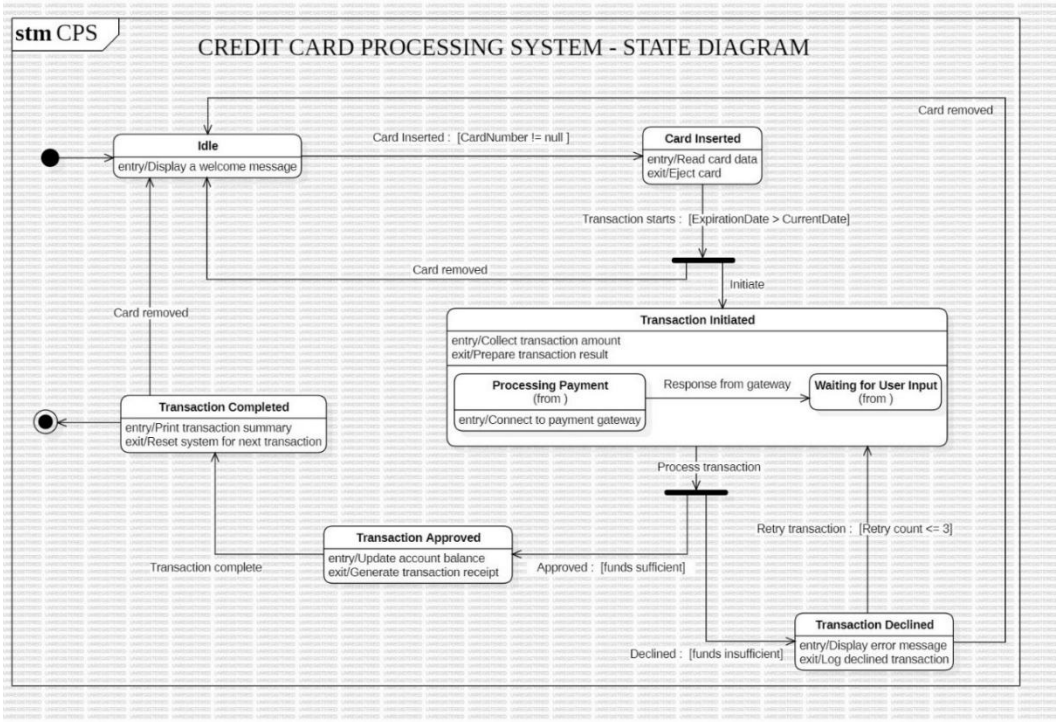


Figure 1 Advanced

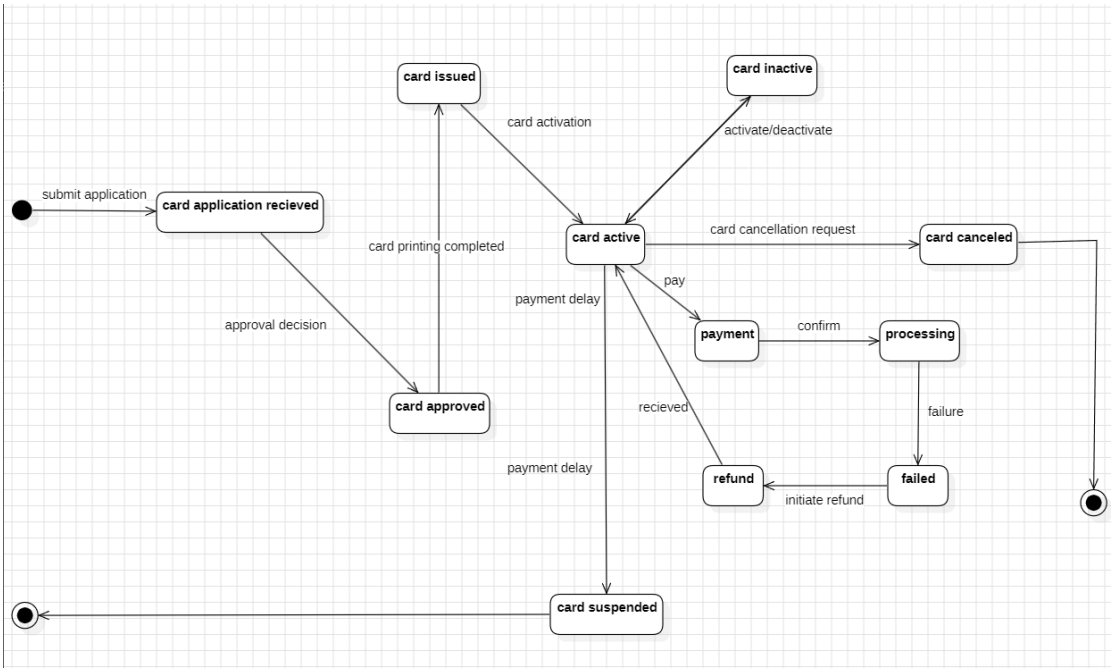


Figure 2 Simple

Key States and Transitions:

1. **Idle:**
    - Entry action: Displays a welcome message.
    - Transition: Card Inserted (CardNumber != null): Proceeds to "Card Inserted" state.
2. **Card Inserted:**
    - Entry action: Reads card data.
    - Exit action: Ejects the card.
    - Transition: Transaction starts (ExpirationDate > CurrentDate): Proceeds to "Transaction Initiated" state.
    - Transition: Card removed: Returns to "Idle" state.
3. **Transaction Initiated:**
    - Entry action: Collects transaction amount.
    - Exit action: Prepares transaction result.
    - Transitions:
        - Processing Payment: Proceeds to "Processing Payment" state.
        - Waiting for User Input: Waits for user input.
        - Retry transaction (retry count <= 3): Proceeds to "Processing Payment" state.
4. **Processing Payment:**
    - Entry action: Connects to the payment gateway.
    - Transitions:
        - Response from gateway: Proceeds to "Transaction Approved" or "Transaction Declined" state based on the response.
5. **Transaction Approved:**
    - Entry action: Updates account balance.
    - Exit action: Generates transaction receipt.
    - Transition: Transaction complete: Proceeds to "Transaction Completed" state.
6. **Transaction Declined:**
    - Entry action: Displays an error message.
    - Exit action: Logs the declined transaction.
    - Transition: Card removed: Returns to "Idle" state.

7. **Transaction Completed:**
    - Entry action: Prints transaction summary.
    - Exit action: Resets the system for the next transaction.
    - Transition: Card removed: Returns to "Idle" state.

3. USE CASE DIAGRAM

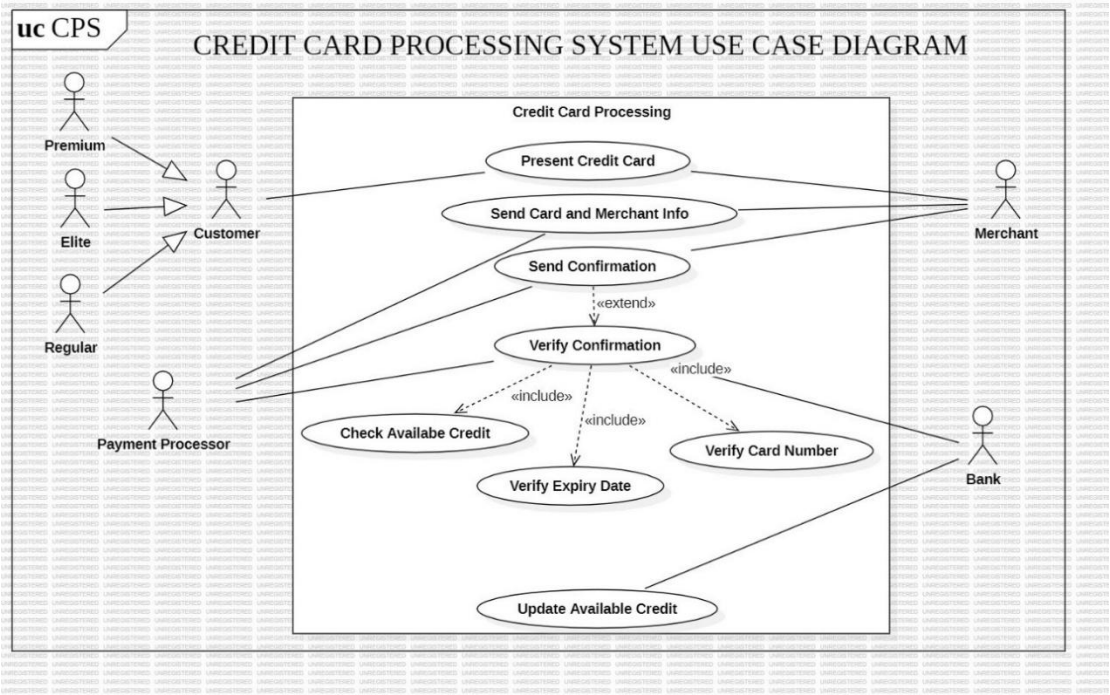


Figure 3 Advanced use case

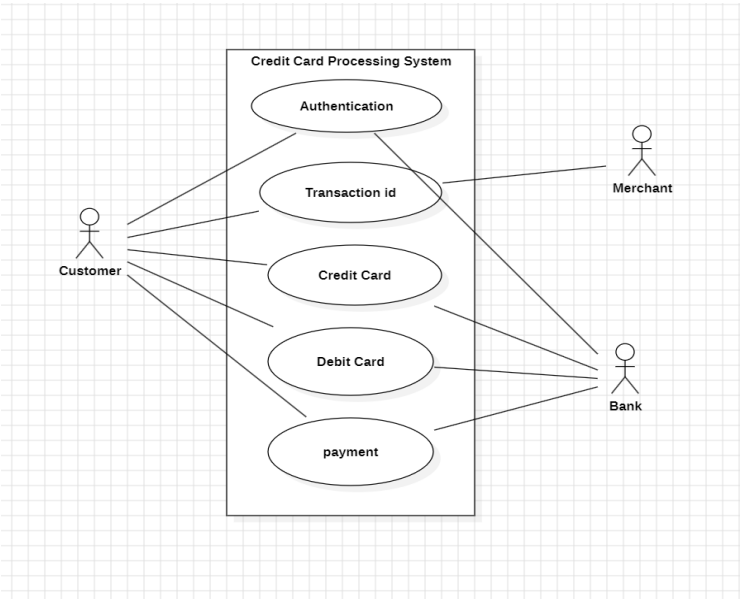


Figure 4 Simple use case

Key Actors and Use Cases:

1. **Customer:**
    - **Present Credit Card:** The customer presents their credit card for a transaction.
    - **Send Card and Merchant Info:** The customer sends their card details and merchant information to the system.
    - **Send Confirmation:** The customer confirms the transaction.
    - **Verify Confirmation:** The system verifies the customer's confirmation.
    - **Check Available Credit:** The system checks if the customer has sufficient credit available for the transaction.
    - **Verify Card Number:** The system verifies the validity of the card number.
    - **Verify Expiry Date:** The system verifies the card's expiry date.
2. **Merchant:**
    - **Present Credit Card:** The merchant receives the customer's credit card.
    - **Send Card and Merchant Info:** The merchant sends the card details and merchant information to the system.
3. **Payment Processor:**
    - **Check Available Credit:** The payment processor verifies the customer's credit limit.
    - **Verify Card Number:** The payment processor validates the card number.
    - **Verify Expiry Date:** The payment processor checks the card's expiry date.
4. **Bank:**
    - **Update Available Credit:** The bank updates the customer's available credit after a successful transaction.

Relationships:

- **Include Relationship:** The use case "Verify Confirmation" includes the sub-use cases "Check Available Credit," "Verify Card Number," and "Verify Expiry Date."
- **Extend Relationship:** The use case "Send Confirmation" can be extended by additional scenarios, such as requesting a receipt or adding a tip.

4. SEQUENCE DIAGRAM

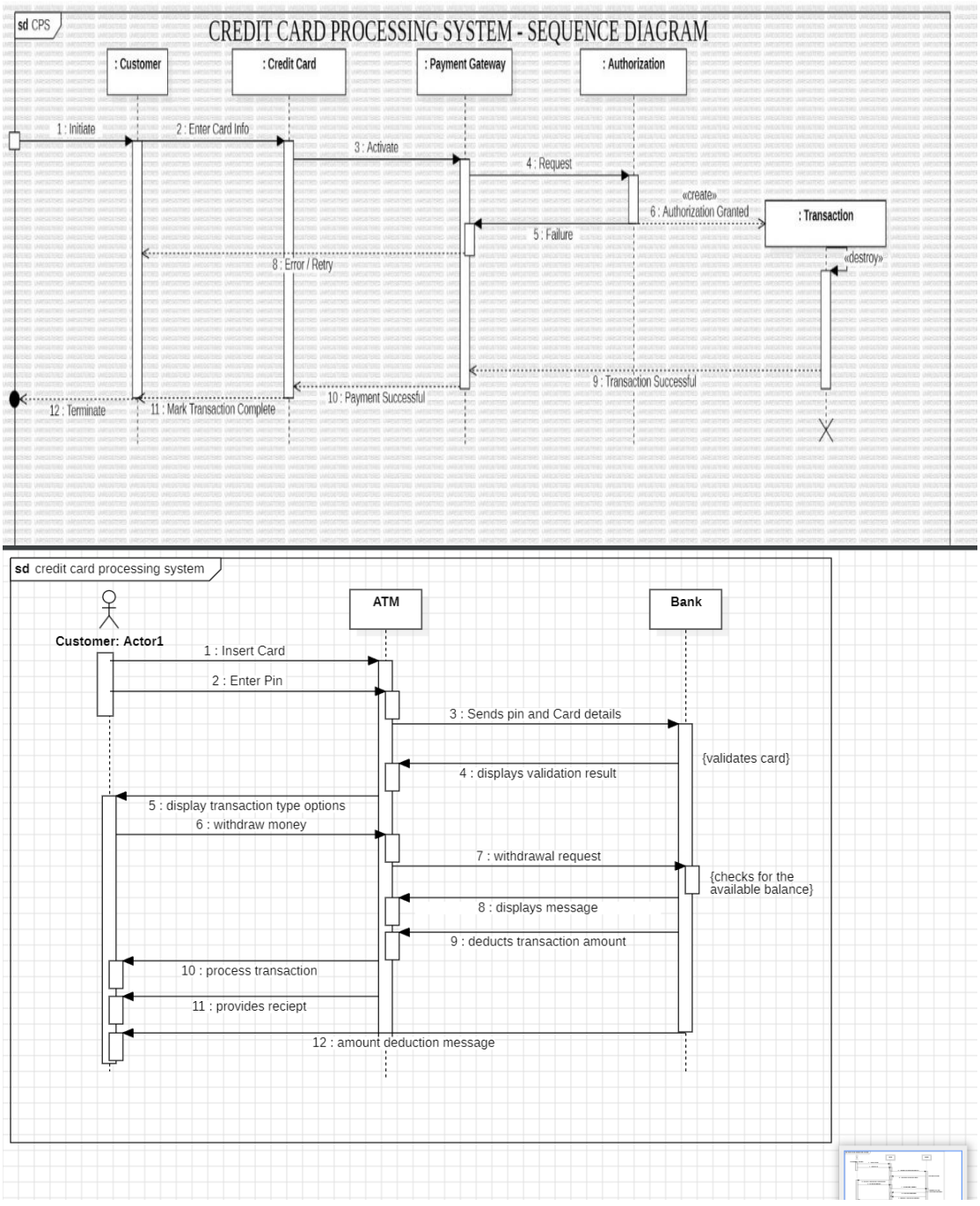


Figure 5 Simple

Key Entities and Interactions:

1. **Customer:**
    - Initiates the process (message 1).
    - Enters card information (message 2).
    - Receives authorization status (message 5).
    - Receives notification of transaction success or failure (message 9).
    - Terminates the interaction (message 12).
2. **Credit Card:**
    - Receives card information (message 2).
    - Activates the card (message 3).
3. **Payment Gateway:**
    - Receives the activation request (message 3).
    - Sends an authorization request to the Authorization entity (message 4).
    - Receives authorization response (message 6).
    - Notifies the Customer of transaction success or failure (message 9).
4. **Authorization:**
    - Receives the authorization request (message 4).
    - Grants or denies authorization (message 6).
5. **Transaction:**
    - Is created upon authorization (message 6).
    - Is destroyed in case of failure (message 8).

5. ACTIVITY DIAGRAM

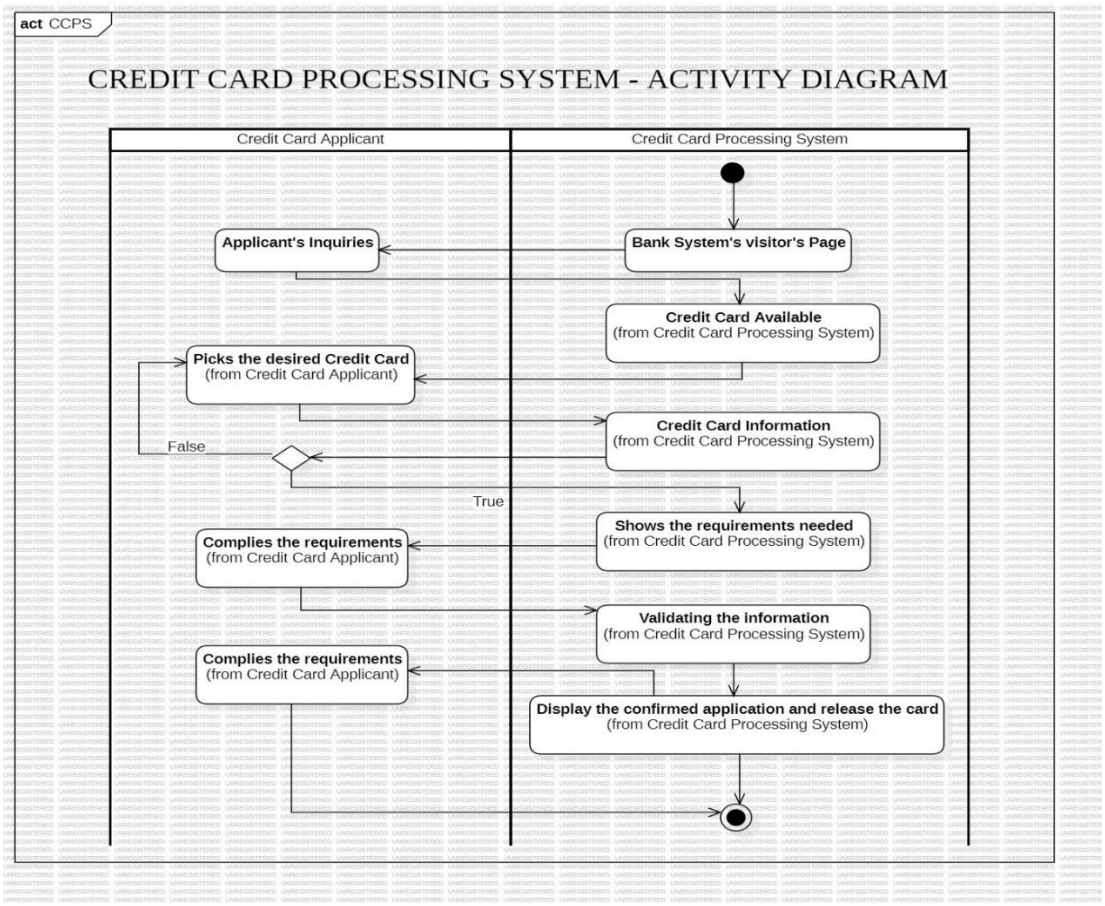


Figure 7 Advanced

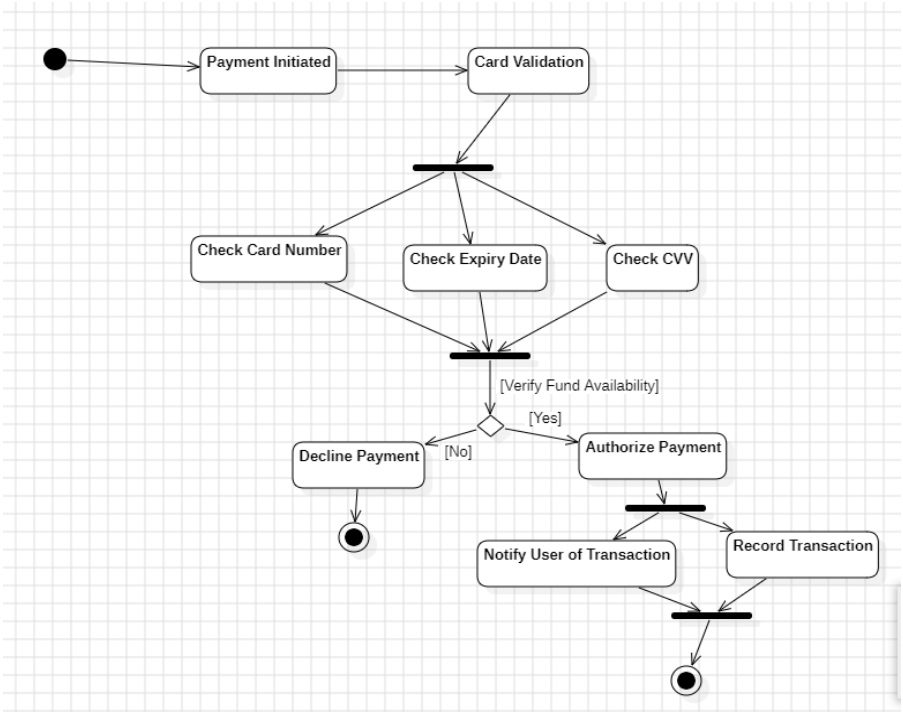
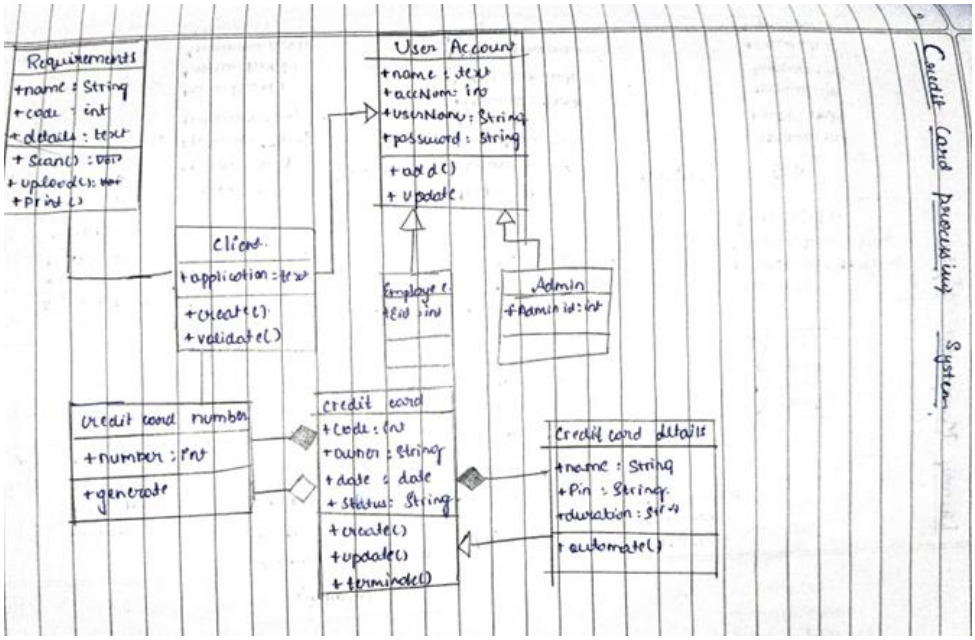


Figure 8 Simple

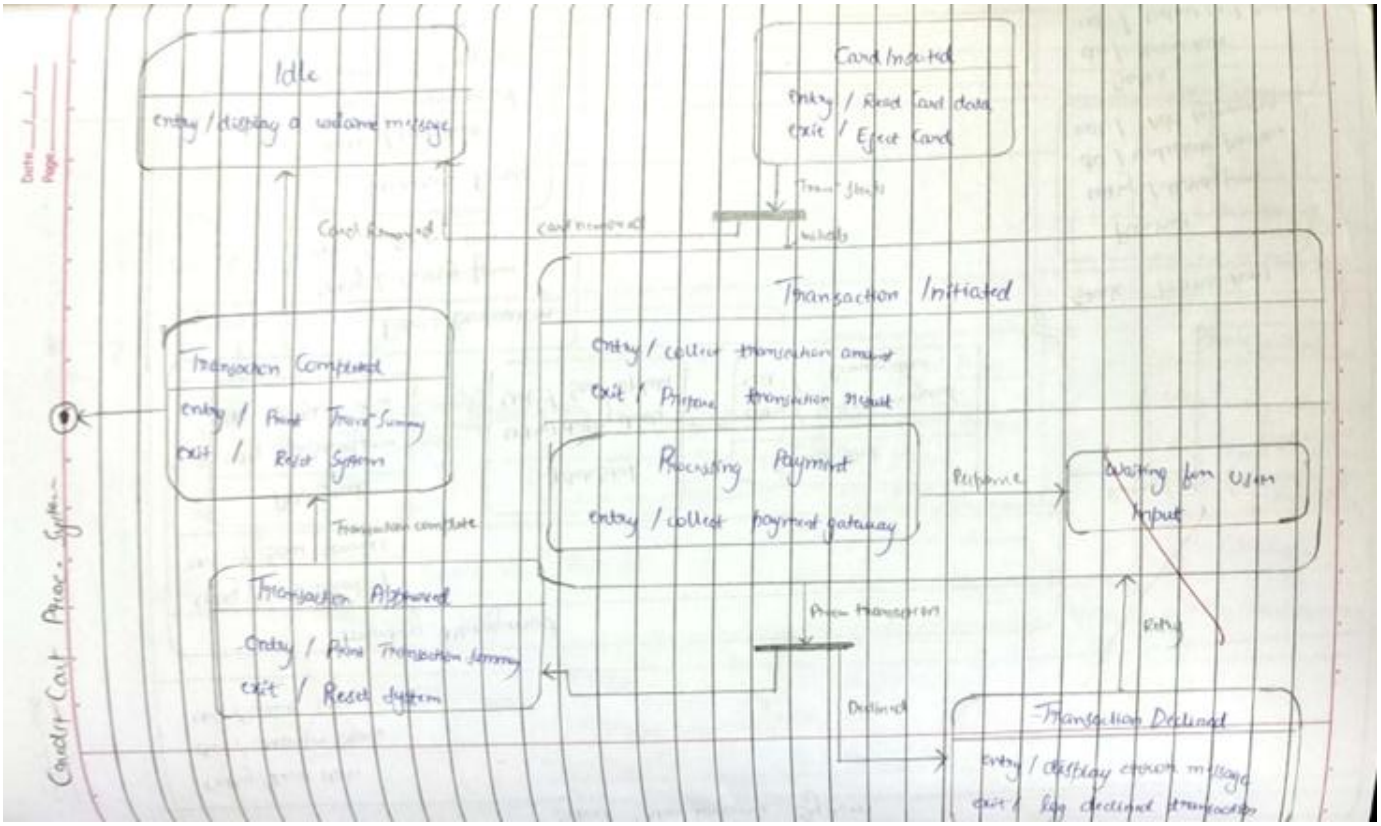
Key Activities and Flows:

1. **Applicant's Inquiries:** The applicant initiates the process by inquiring about available credit cards.
2. **Bank System's Visitor's Page:** The Credit Card Processing System displays the available credit card options to the applicant.
3. **Picks the desired Credit Card:** The applicant chooses the desired credit card.
4. **Credit Card Information:** The Credit Card Processing System displays the information related to the chosen credit card, including requirements and terms.
5. **Complies the requirements:** The applicant checks if they meet the requirements for the selected credit card.
6. **Shows the requirements needed:** If the applicant does not meet the requirements, the Credit Card Processing System displays the missing requirements.
7. **Validating the Information:** The Credit Card Processing System validates the information provided by the applicant.
8. **Display the confirmed application and release the card:** If the applicant meets all the requirements and the information is validated, the Credit Card Processing System displays the confirmed application and releases the credit card.

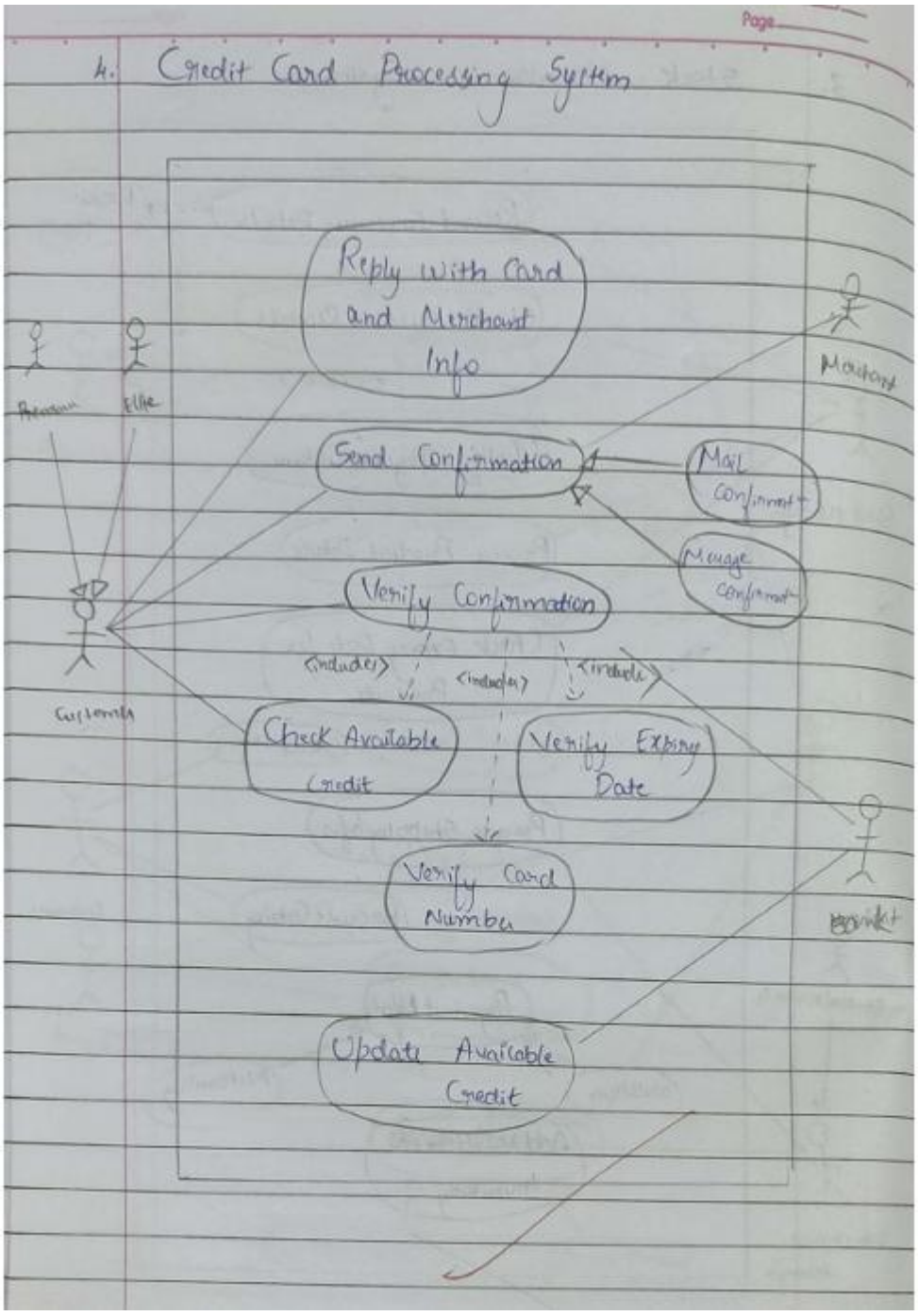
1. CLASS DIAGRAM



2. STATE DIAGRAM

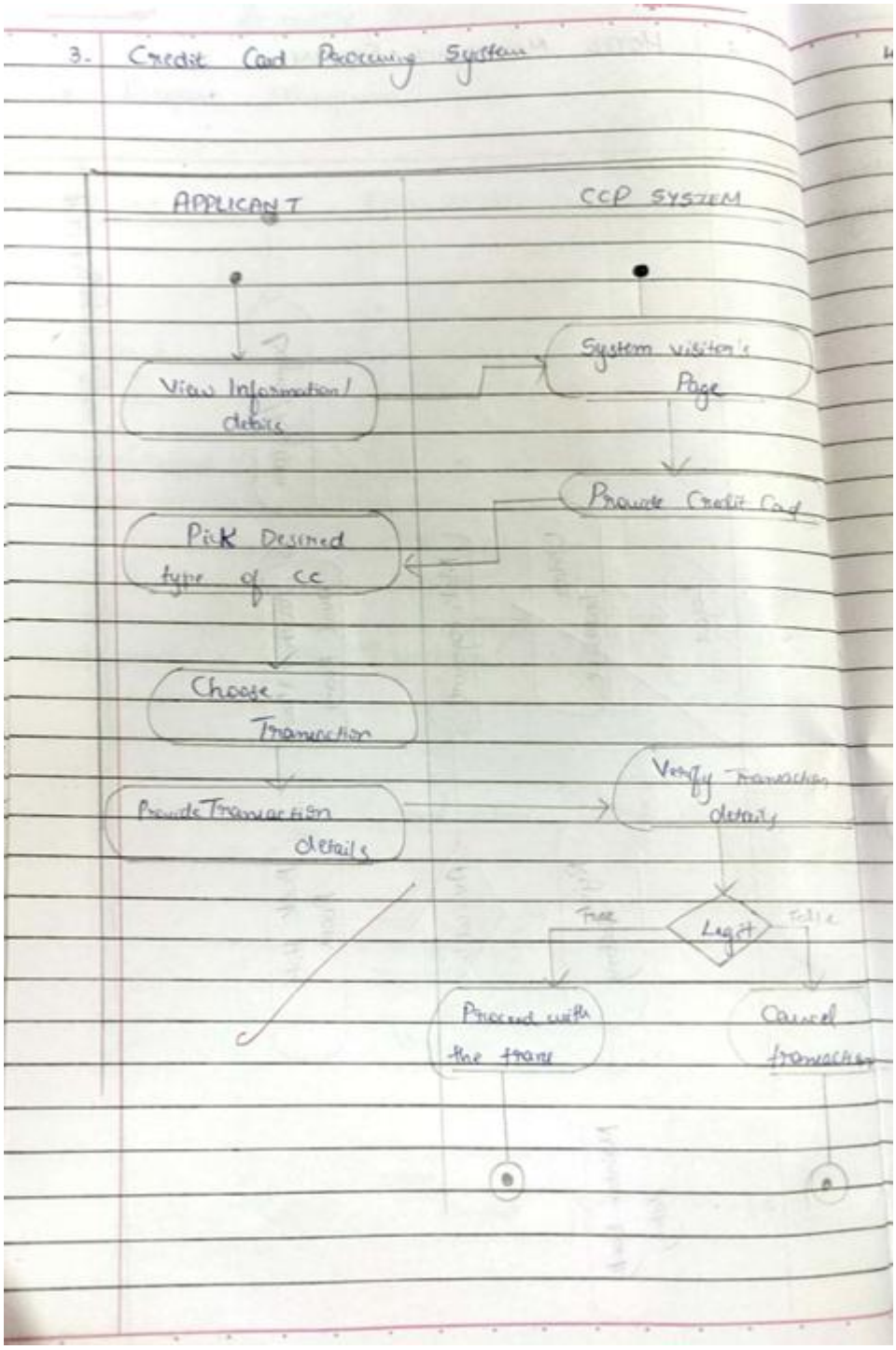


3. USE CASE DIAGRAM



4. SEQUENCE DIAGRAM

5. ACTIVITY DIAGRAM



CHAPTER – 3

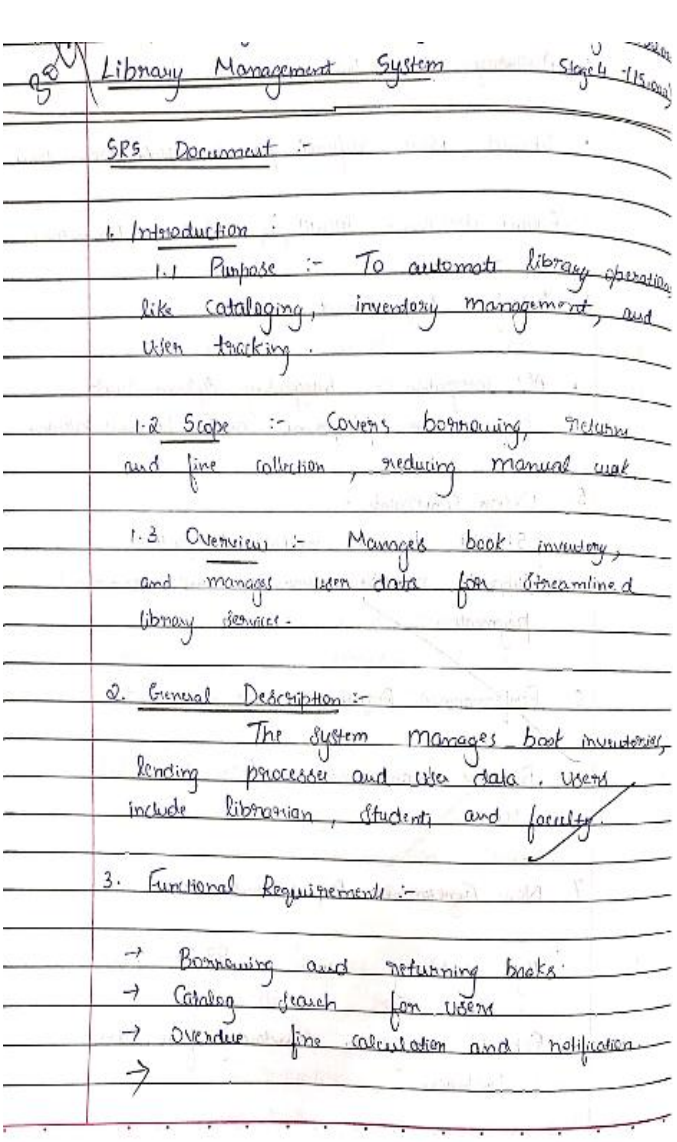
LIBRARY MANAGEMENT SYSTEM

PROBLEM STATEMENT

The objective of this project is to develop an efficient and user-friendly Library Management System to streamline library operations and enhance user experience.

- The system will address several critical functionalities to improve library management and user satisfaction.
- The system will facilitate book management by allowing the addition, removal, and categorization of books, along with real-time tracking of book availability to ensure efficient inventory management.
- It will enable user management, supporting registration and management of librarians, members, and administrators, with appropriate access permissions and roles for each user type.
- Borrowing and return management will be streamlined, providing seamless processes for issuing and returning books, automated due date tracking, fine calculation, and reminders for overdue books.
- Advanced search functionalities will be implemented, allowing users to search by keywords, genre, or author, enabling them to locate books quickly.
- A reservation system will allow users to reserve books that are currently unavailable, with automated notifications when the reserved books become available.
- The system will generate detailed reports and analytics on library usage, borrowing trends, and book inventory to support decision-making and resource allocation.

SOFTWARE REQUIREMENTS SPECIFICATION



Library Management System

SRS Document :-

1. Introduction :

1.1 Purpose :- To automate library operations like cataloging, inventory management, and user tracking.

1.2 Scope :- Covers borrowing, returns and fine collection, reducing manual work

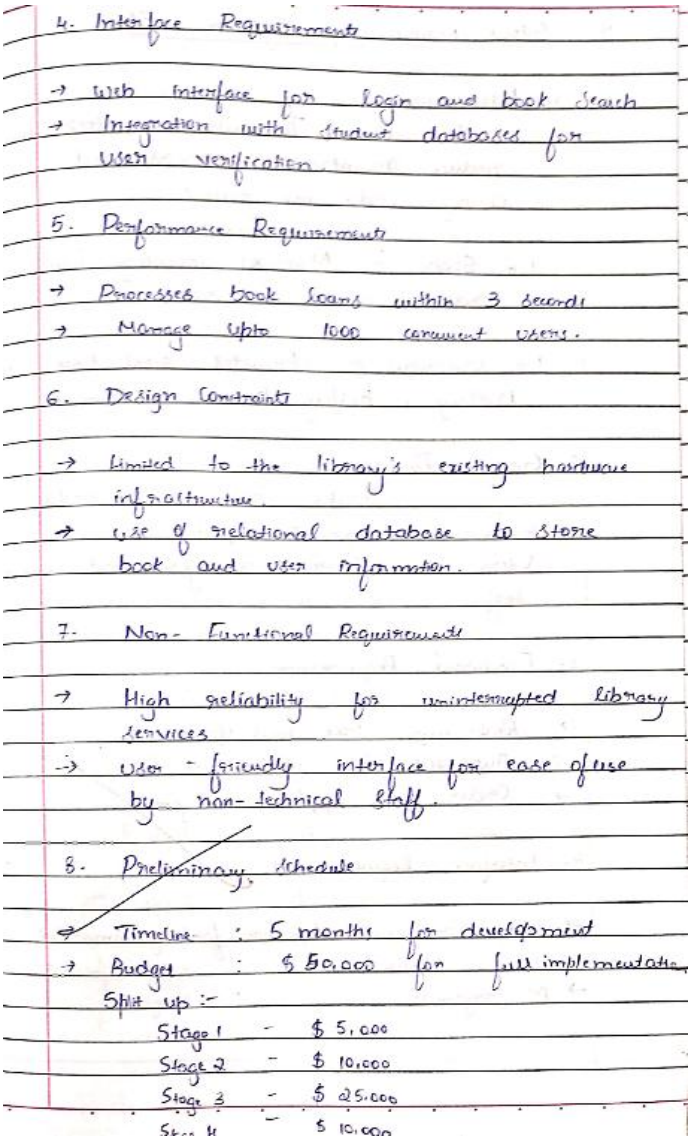
1.3 Overview :- Manages book inventory, and manages user data for streamlined library services.

2. General Description :-

The system manages book inventories, lending processes and user data. Users include librarian, students and faculty.

3. Functional Requirements :-

→ Borrowing and returning books
→ Catalog search for users
→ Overdue fine calculation and notification
→



4. Interface Requirements

→ Web interface for login and book search
→ Integration with student databases for user verification.

5. Performance Requirements

→ Processes book loans within 3 seconds
→ Manage upto 1000 concurrent users.

6. Design Constraints

→ Limited to the library's existing hardware infrastructure.
→ Use of relational database to store book and user information.

7. Non-Functional Requirements

→ High reliability for uninterrupted library services
→ User-friendly interface for ease of use by non-technical staff.

8. Preliminary Schedule

→ Timeline : 5 months for development
→ Budget : $ 50,000 for full implementation
Split up :-
Stage 1 - $ 5,000
Stage 2 - $ 10,000
Stage 3 - $ 25,000
Stage 4 - $ 10,000

UML DIAGRAMS

1. CLASS DIAGRAM

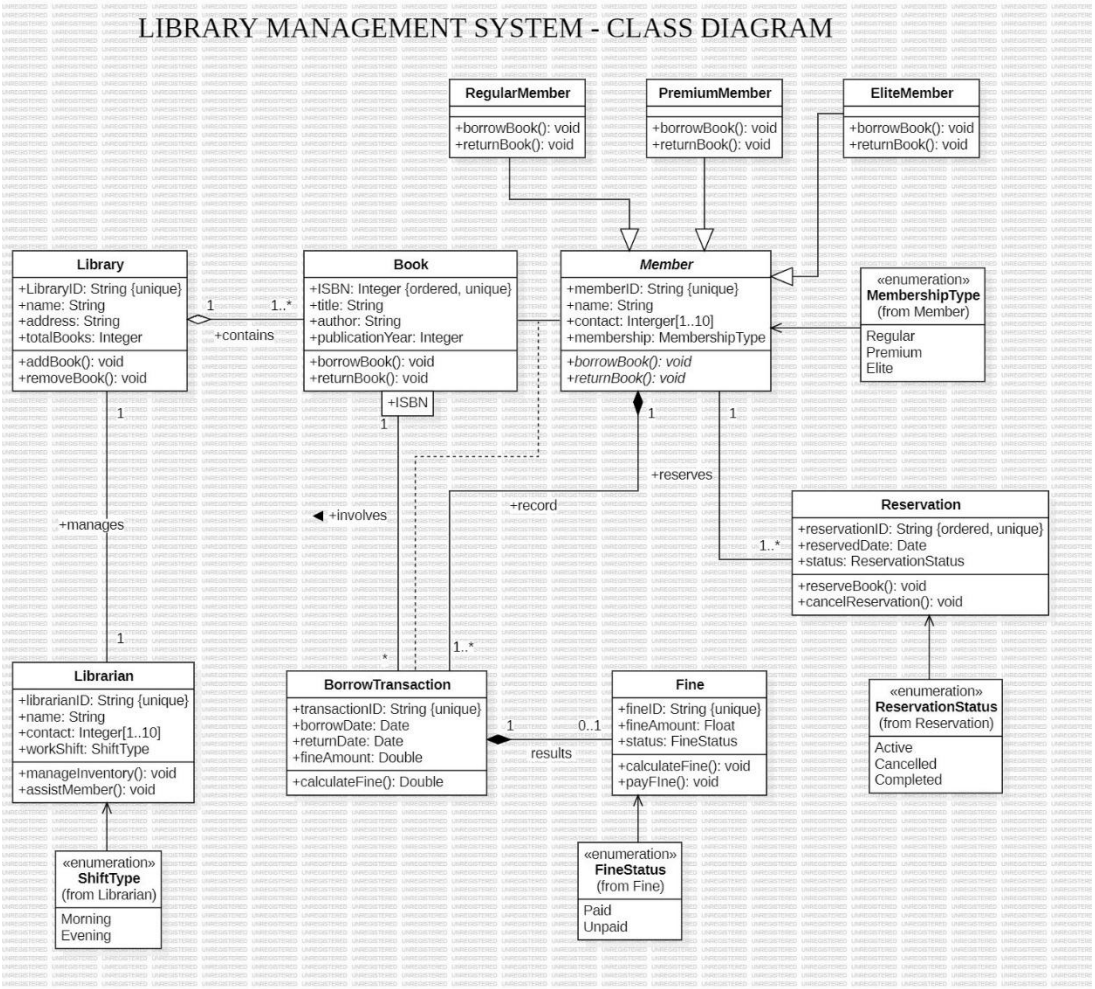


Figure 3.1: Class Diagram

Classes:

- **Library:** This class manages the library's inventory, including books. It has attributes like LibraryID, name, address, and totalBooks.
- **Book:** This class represents a book with attributes like ISBN, title, author, and publication year.
- **Member:** This is an abstract class with subclasses RegularMember, PremiumMember, and EliteMember. Each member type has different borrowing privileges.
- **Librarian:** This class manages the library's operations, including inventory and member assistance.

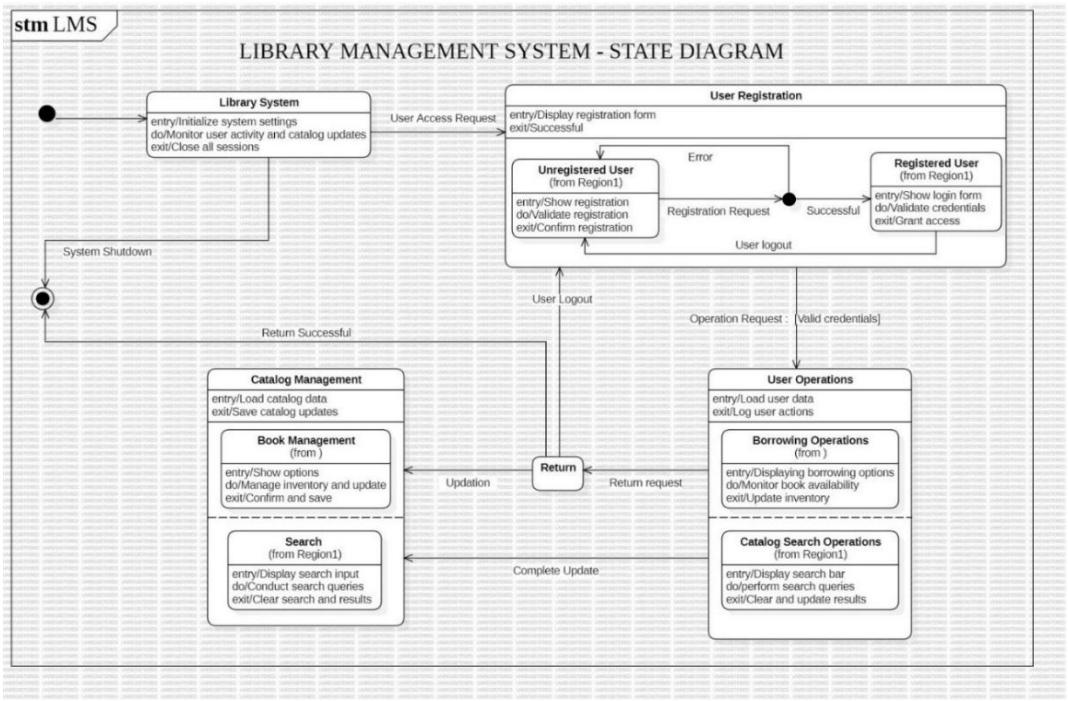
- **Borrow Transaction:** This class records borrowing and returning of books, including dates and fines.
- **Reservation:** This class handles book reservations, including reservation ID, reserved date, and status.
- **Fine:** This class tracks fines imposed on members for overdue books.

Relationships:

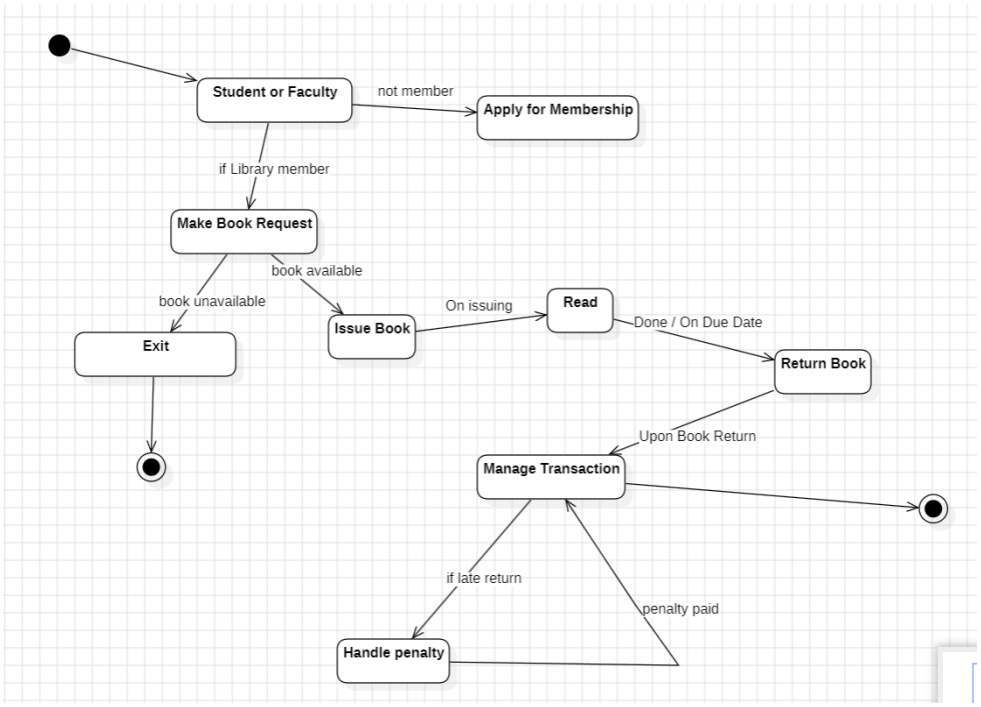
- **Library contains Books:** A library has many books.
- **Library manages Members:** A library manages various types of members.
- **Member borrows Books:** Members can borrow books.
- **Librarian manages Library:** A librarian manages the library's operations.
- **Borrow Transaction involves Books:** A borrowing transaction involves a specific book.
- **Reservation involves Books:** A reservation is made for a specific book.
- **Fine results from Borrow Transaction:** Fines are calculated based on overdue borrow transactions.

2.STATE DIAGRAM

Advanced



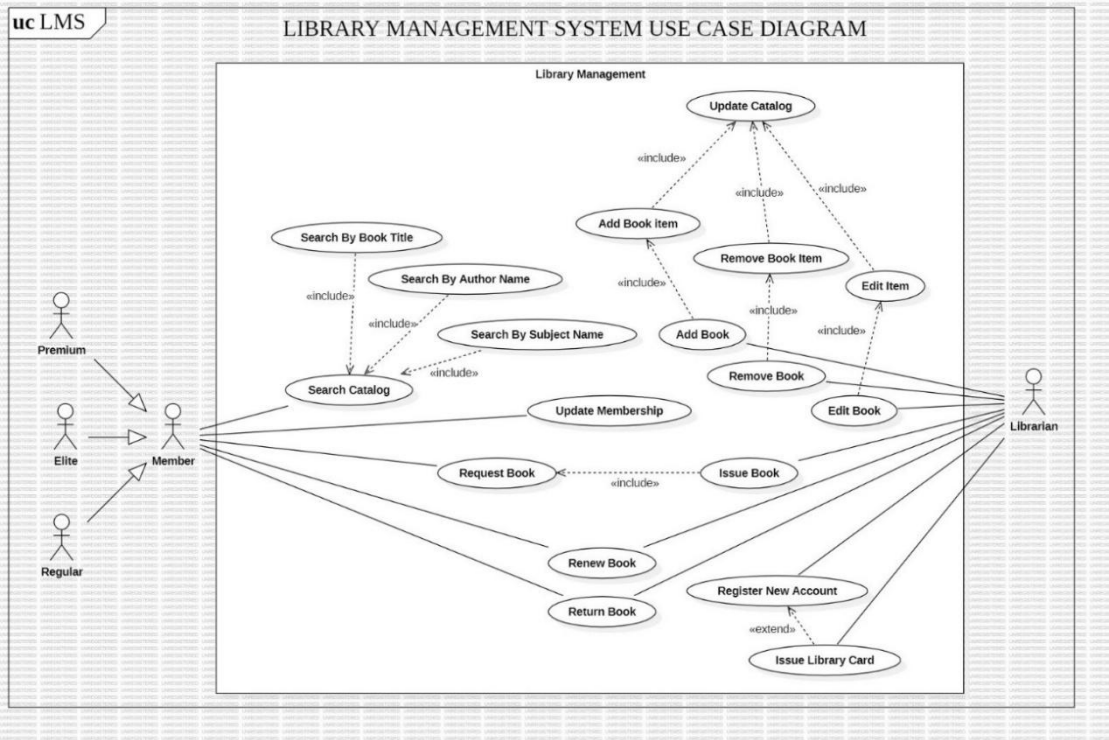
Simple:



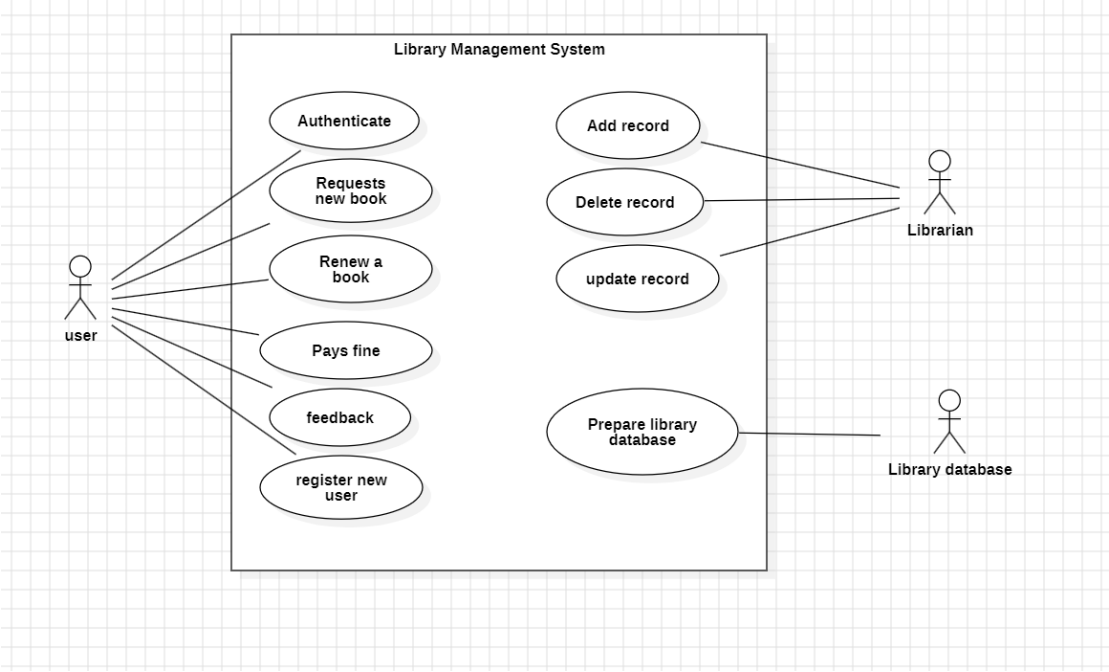
Key States and Transitions:

1. **Library System**
    - Entry action: Initializes system settings.
    - Do action: Monitors user activity and catalog updates.
    - Exit action: Closes all sessions.
    - Transitions:
        - User Access Request: Leads to user registration or login.
        - System Shutdown: Terminates the system.
2. **User Registration**
    - Entry action: Displays registration form.
    - Transitions:
        - Successful registration: Leads to the "Registered User" state.
        - Error: Remains in the "User Registration" state.
3. **Registered User**
    - Entry action: Shows login form.
    - Do action: Validates credentials.
    - Exit action: Grants access to user operations.
    - Transition: User logout: Returns to the "Library System" state.
4. **User Operations**
    - Entry action: Loads user data.
    - Exit action: Logs user actions.
    - Transitions:
        - Borrowing Operations: Allows users to borrow books.
        - Catalog Search Operations: Enables users to search the library catalog.
5. **Borrowing Operations**
    - Entry action: Displays borrowing options.
    - Do action: Monitors book availability.
    - Exit action: Updates inventory.
    - Transition: Return request: Initiates the return process.
6. **Catalog Search Operations**
    - Entry action: Displays search bar.
    - Do action: Performs search queries.
    - Exit action: Clears search and updates results.

3.USE CASE DIAGRAM (adv)



USE CASE (Simple)



Actors:

- **Member:** Represents library users, which can be of different types:
- **Librarian:** Represents the library staff responsible for managing the library operations.

Use Cases:

- **Library Management:** This is the main use case, encompassing all the functions related to managing the library.
    - **Add Book:** Use case for adding a new book title to the catalog.
    - **Remove Book:** Use case for removing a book title from the catalog.
    - **Update Membership:** Use case for updating member information or membership types.
- **Search Catalog:** Use case for searching the library catalog.
- **Request Book:** Use case for members to request a book that is currently unavailable.
- **Renew Book:** Use case for members to renew a borrowed book.
- **Return Book:** Use case for members to return a borrowed book.

4. SEQUENCE DIAGRAM

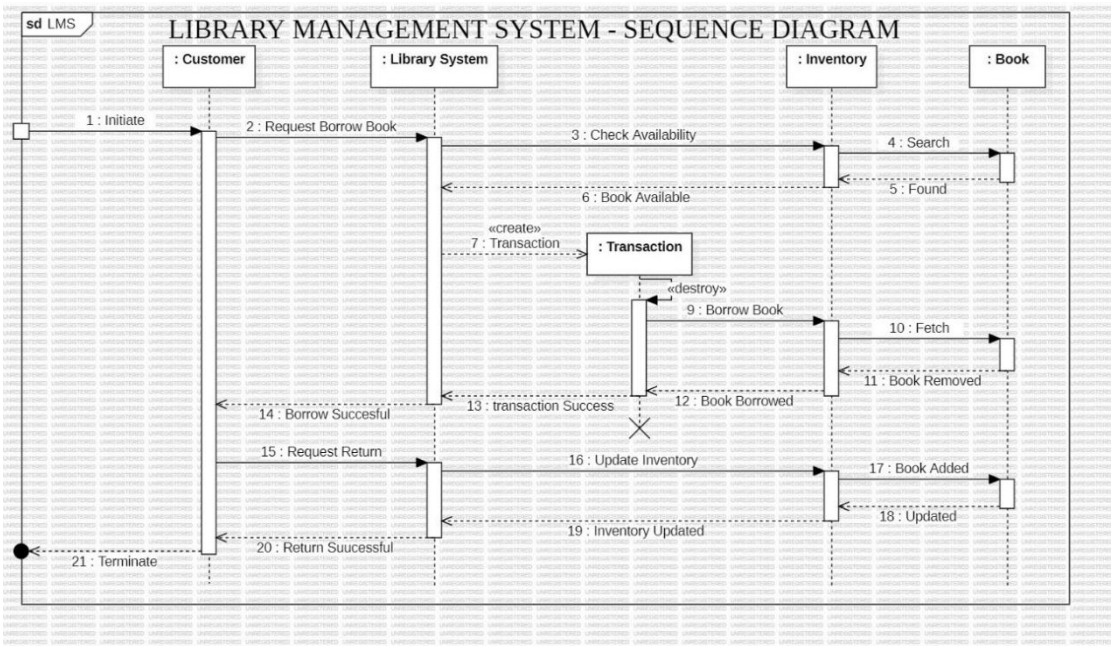


Figure 9 Advanced

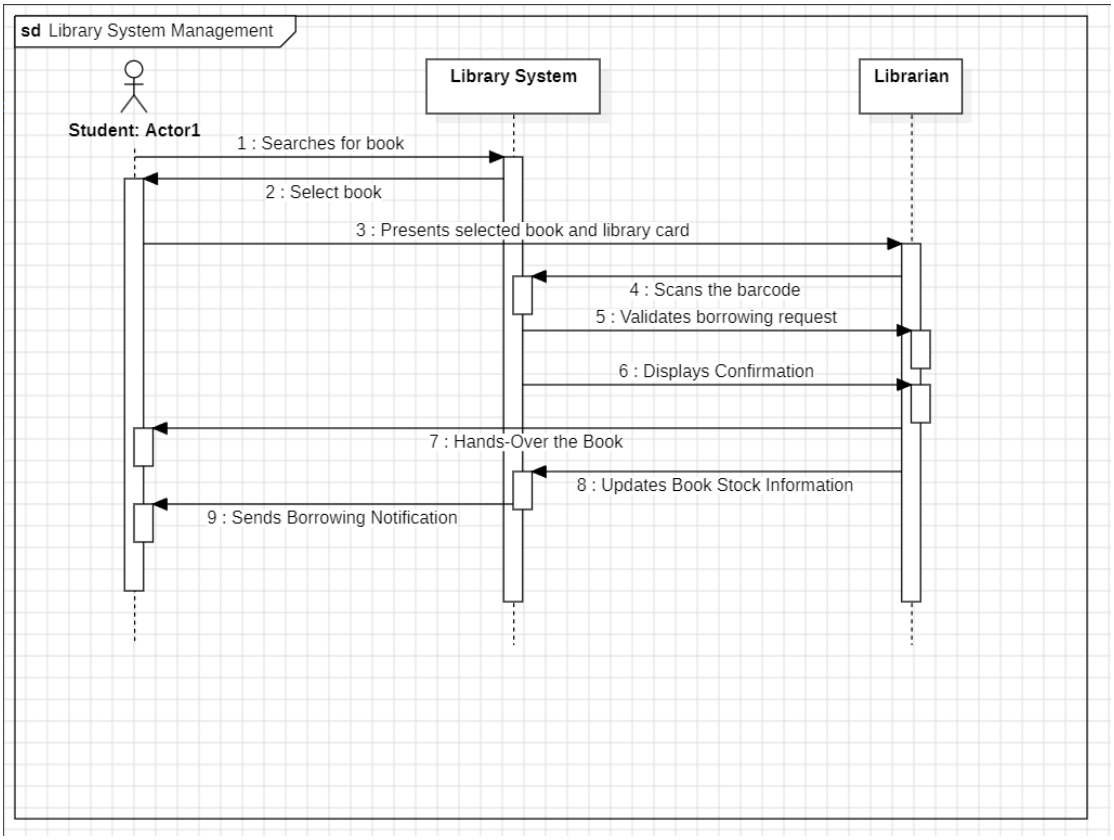


Figure 10 Simple

Key Entities and Interactions:

1. **Customer:**
    - Initiates the process by requesting to borrow a book (message 2).
    - Receives confirmation of successful borrowing (message 14).
    - Requests to return the book (message 15).
    - Receives confirmation of successful return (message 20).
    - Terminates the interaction (message 21).
2. **Library System:**
    - Receives the borrow book request (message 2).
    - Checks the availability of the book in the inventory (message 3).
    - Searches for the book (message 4).
    - Notifies the customer of book availability (message 6).
    - Creates a transaction object (message 7).
    - Updates the inventory to reflect the book being borrowed (messages 9, 10, 11, 12).
    - Receives the return book request (message 15).
    - Updates the inventory to reflect the book being returned (messages 16, 17, 18, 19).
    - Notifies the customer of successful return (message 20).
3. **Inventory:**
    - Receives the availability check request (message 3).
    - Searches for the book (message 5).
    - Notifies the Library System of book availability (message 6).
    - Receives the update request when the book is borrowed (message 9).

- Updates its records to reflect the book being removed (messages 10, 11).
- Receives the update request when the book is returned (message 16).
- Updates its records to reflect the book being added (messages 17, 18).

4. **Book:**
   - Receives the fetch request when the book is borrowed (message 10).
   - Receives the add request when the book is returned (message 17).

5. ACTIVITY DIAGRAM

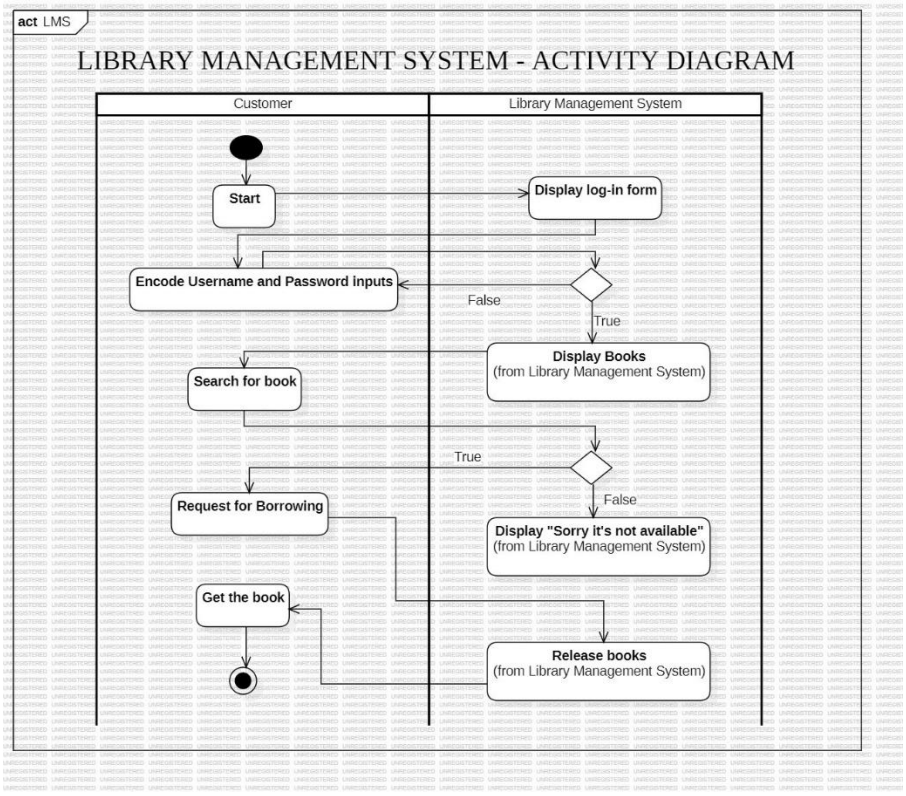


Figure 11 Advanced

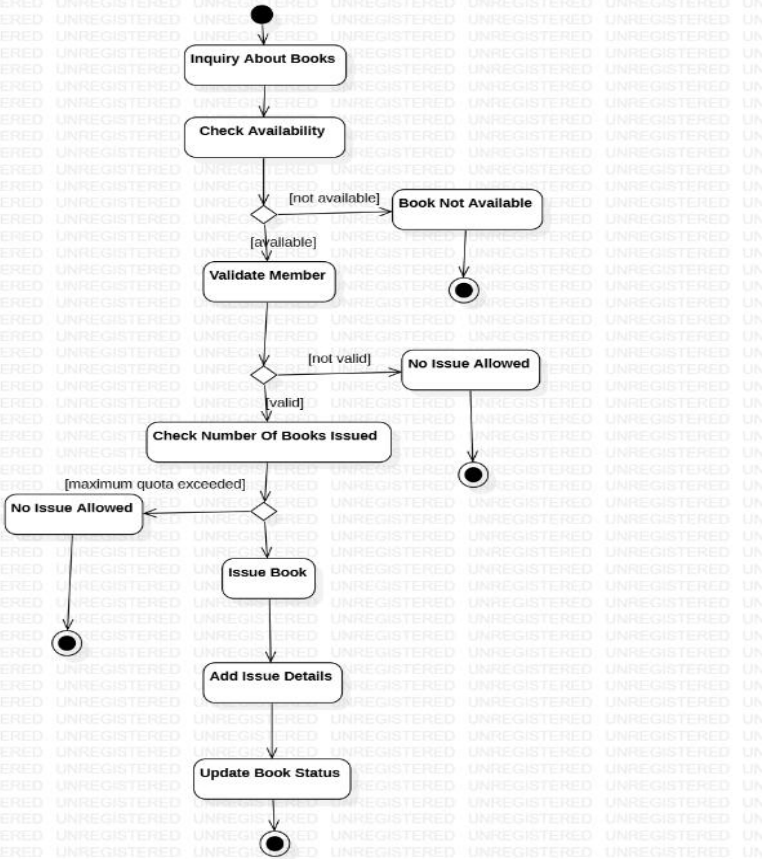
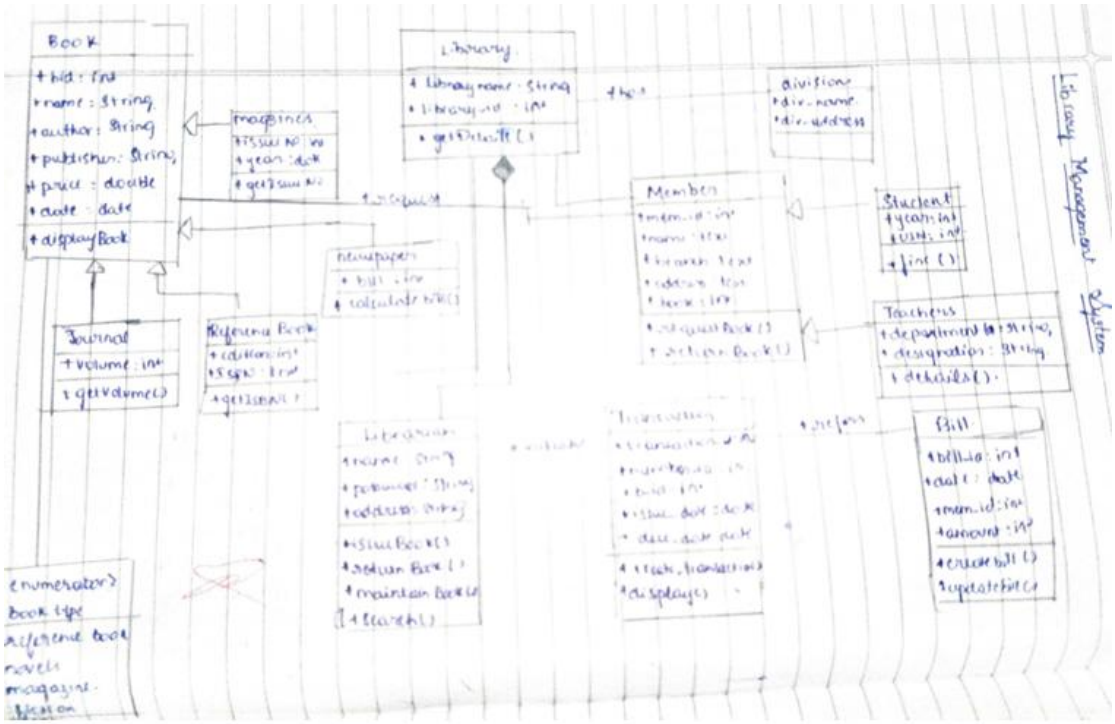


Figure 12 Simple

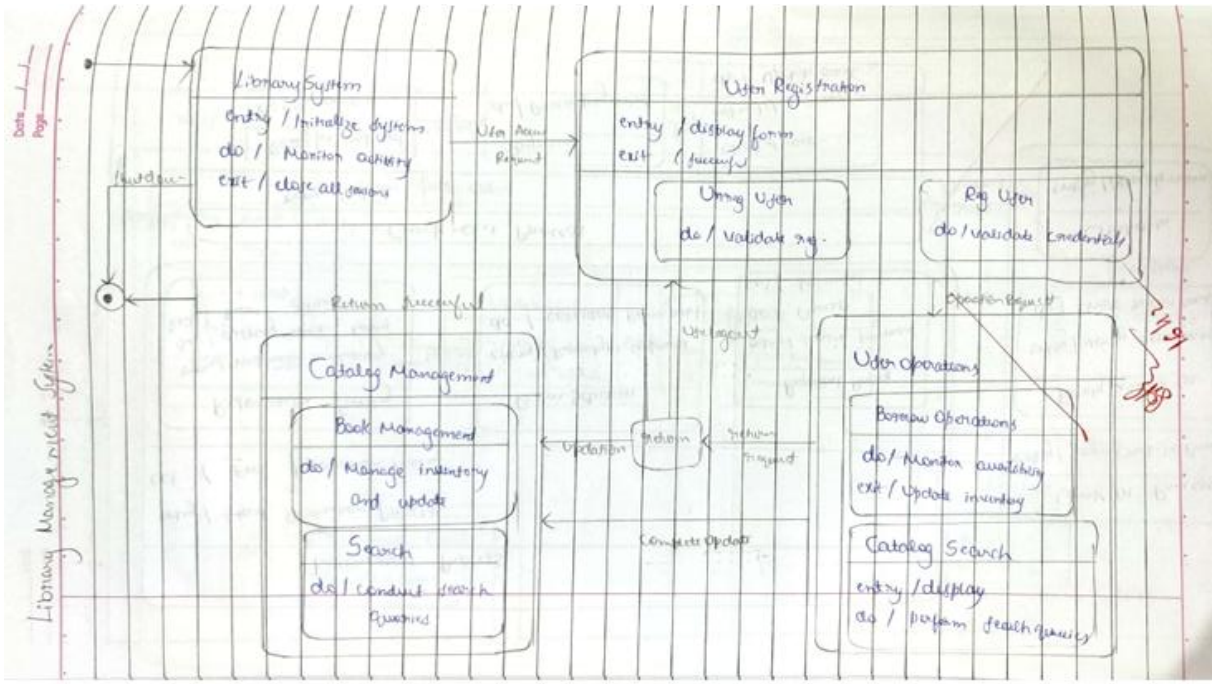
Swimlanes:

1. **Customer:** Represents actions by the library user, including:
    - Start: Initial user interaction.
    - Enter Credentials: User inputs login details.
    - Search for Book: User searches the catalog.
    - Request Borrowing: User requests a book.
    - Receive Book: User borrows the book.
2. **Library Management System:** Represents actions by the system, including:
    - Display Login: The system shows the login form.
    - Show Books: Displays available books after login.
    - Show "Unavailable": Displays an error if the book is unavailable.
    - Release Book: Handles book returns.

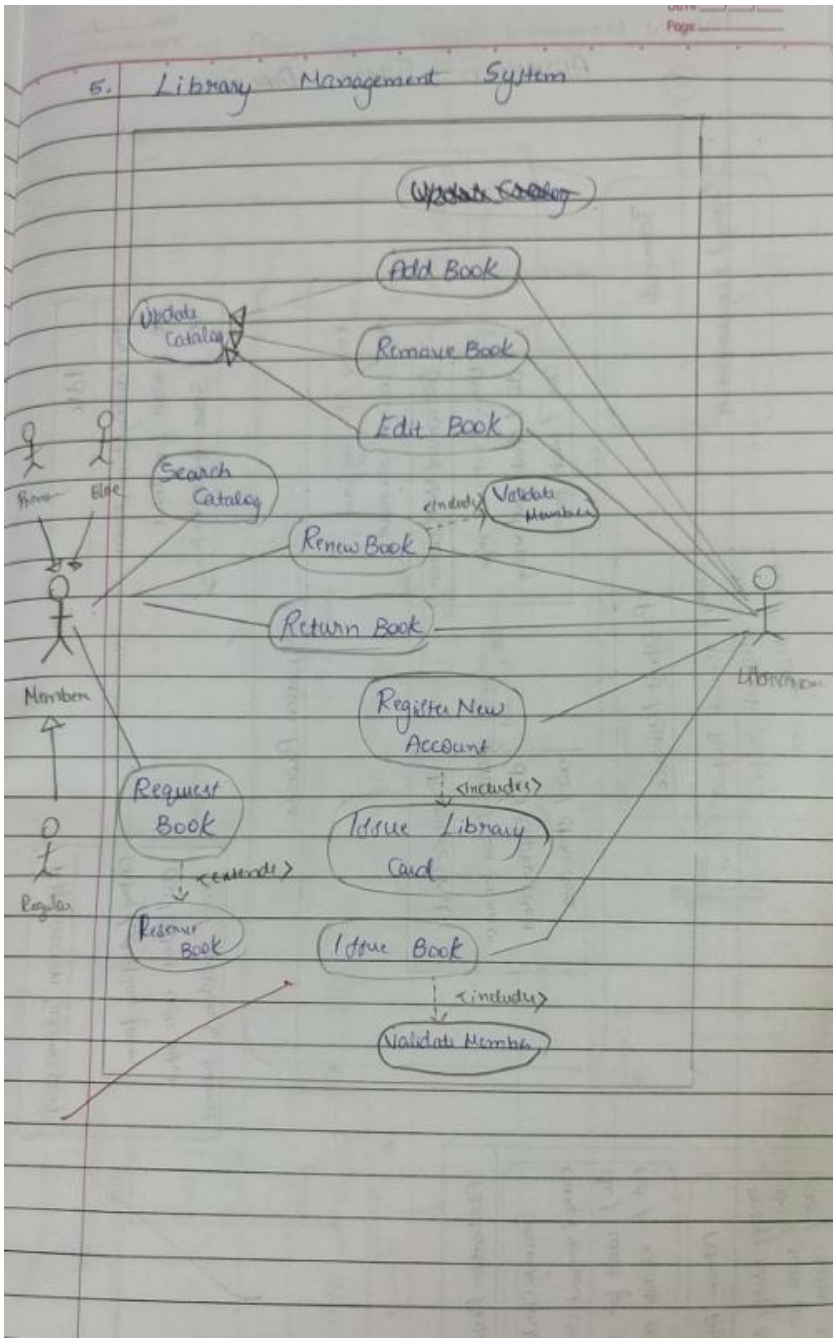
1. CLASS DIAGRAM



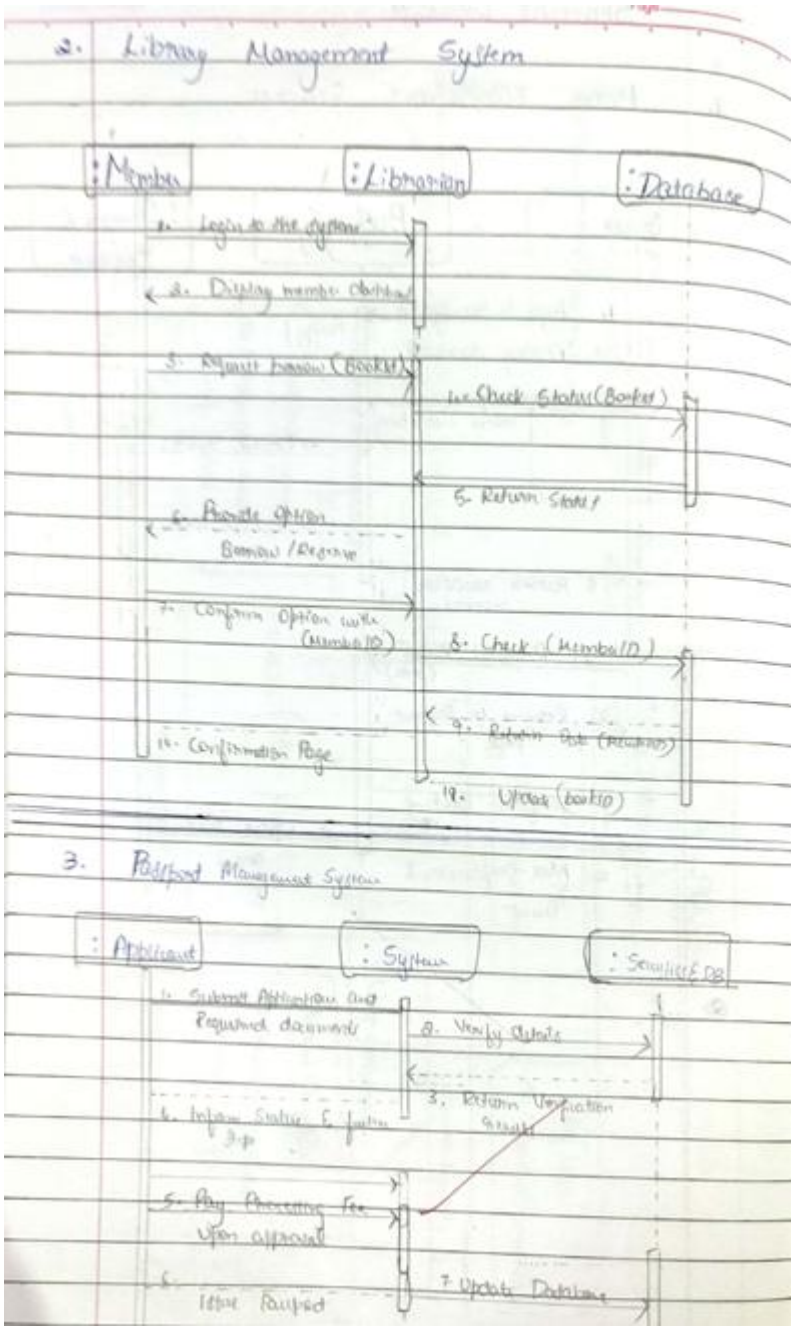
2. STATE DIAGRAM



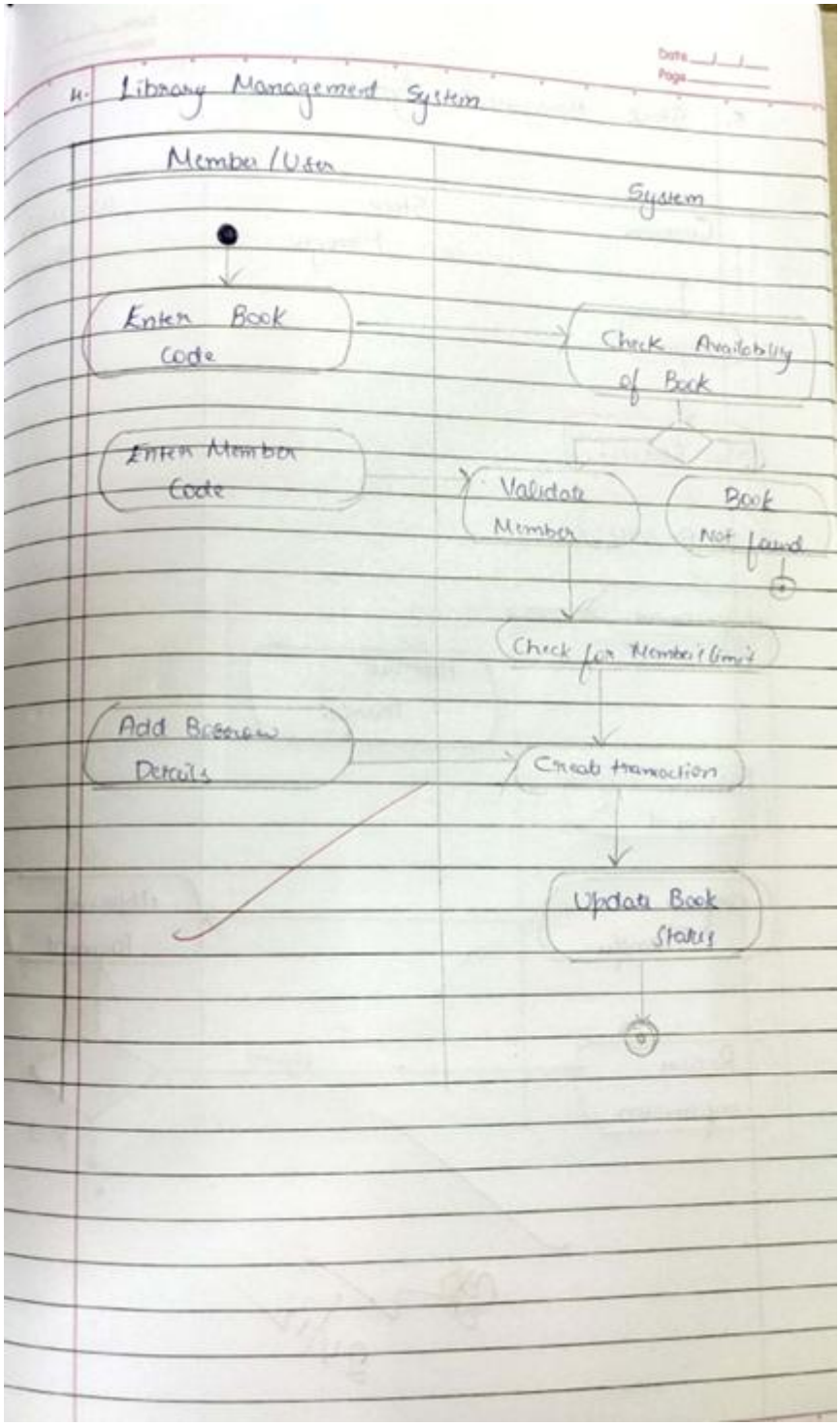
3. USE CASE DIAGRAM



4. SEQUENCE DIAGRAM



5. ACTIVITY DIAGRAM



CHAPTER – 4

STOCK MAINTENANCE SYSTEM

PROBLEM STATEMENT

The objective of this project is to develop a reliable and efficient Stock Maintenance System to streamline inventory management and enhance operational efficiency. The system will address several key functionalities to improve stock management and business operations.

- The system will enable real-time inventory tracking, categorizing items, and managing stock additions, deletions, and adjustments to ensure accurate inventory records.
- It will provide tools for supplier management, allowing the maintenance of supplier information, management of purchase orders, and tracking of delivery schedules to ensure seamless stock replenishment.
- Stock monitoring mechanisms will be implemented to track stock usage, set reorder thresholds, and generate alerts for low stock levels to prevent shortages.
- Sales and distribution tracking will be facilitated, maintaining accurate records of stock movements related to sales and ensuring proper sales performance monitoring.
- The system will generate detailed reports on inventory levels, stock turnover, and supplier performance, along with analytics to support informed decision-making.

SOFTWARE REQUIREMENTS SPECIFICATION

IV Stock Maintainance System SRS

1. Introduction

1.1 Purpose :- To automate stock tracking, reduce maintainance errors and optimize order management.

1.2 Scope :- Manages inventory levels, orders, alerts and scope.

1.3 Overview :- Provides real-time stock tracking, helping busines manage

2. General Description :-

Tracks Stock levels, order status, and generate reports. Users include store managers and warehouse staff.

3. Functional Requirements

→ Real-time stock level updates
→ Automated low stock alerts
→ Order and supplier management.

4. Interface Requirement

→ Integration with suppliers for automated orders requests
→ API Integration

5. Performance Requirements

→ Handle stock updates in real time.
→ Concurrently manage upto 20,000 records
→ Support minimum of 500 users.

6. Design constraints

→ Limited to use on company-provided hardware
→ Uses SQL based database for stock records.

7. Non-functional Requirements

→ High reliability and low error rate in inventory tracking
→ Fast and scalable to accomodate large inventory volumes.

8. Schedule and Budget

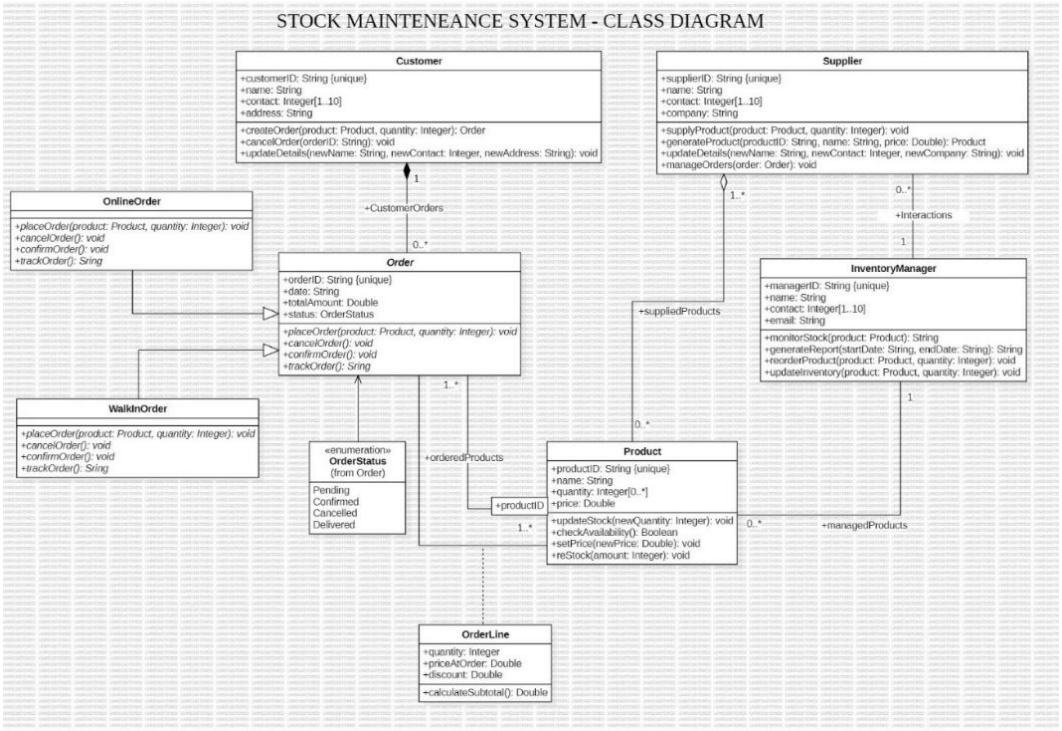
→ Timeline : 3 months for Dev. & 2 months for testing
→ Budget : $ 60,000 for dev.

Split-up's

Stage 1 - $ 10,000
Stage 2 - $ 20,000
Stage 3 - $ 20,000
Stage 4 - $ 10,000

UML DIAGRAMS

1. CLASS DIAGRAM



Key Classes and their Relationships:

1. **Customer:**
    - Attributes: name, address, contact
    - Methods: createOrder, cancelOrder, updateDetails
    - Has-a relationship with CustomerOrder
2. **CustomerOrder:**
    - Attributes: orderID, status
    - Methods: placeOrder, cancelOrder, confirmOrder, trackOrder
    - Has-a relationship with OnlineOrder and WalkInOrder
    - Has-a relationship with OrderLine
3. **OnlineOrder:**
    - Attributes: orderID, orderDate, orderAmount, paymentStatus
    - Methods: placeOrder, cancelOrder, confirmOrder, trackOrder
4. **WalkInOrder:**
    - Attributes: orderID, orderDate, orderAmount, paymentStatus
    - Methods: placeOrder, cancelOrder, confirmOrder, trackOrder
5. **Order:**
    - Attributes: orderID, status
    - Methods: placeOrder, cancelOrder, confirmOrder, trackOrder
    - Has-a relationship with OrderLine
6. **OrderLine:**
    - Attributes: product, quantity, priceAtOrder, discount, subTotal
7. **Product:**
    - Attributes: productID, name, price, quantityInStock
    - Methods: updatePrice, updateStockAmount
8. **InventoryManager:**
    - Attributes: name, contact, address
    - Methods: manageProducts, restockProducts, generateReport, reorderProducts
9. **Supplier:**
    - Attributes: supplierID, name, contact, address
    - Methods: supplyProducts, processOrder

Relationships:

- **Customer has-a CustomerOrder:** A customer can place multiple orders.

- **CustomerOrder has-a OnlineOrder or WalkInOrder:** An order can be either online or walk-in.
- **CustomerOrder has-a OrderLine:** An order can consist of multiple order lines.
- **Order has-a OrderLine:** An order can consist of multiple order lines.
- **InventoryManager manages Products:** The Inventory Manager is responsible for managing the inventory of products.
- **Supplier supplies Products:** Suppliers provide products to the system.

Enumerations:

- **OrderStatus:** Defines the possible statuses of an order (e.g., Pending, Confirmed, Cancelled, Delivered)

2. STATE DIAGRAM

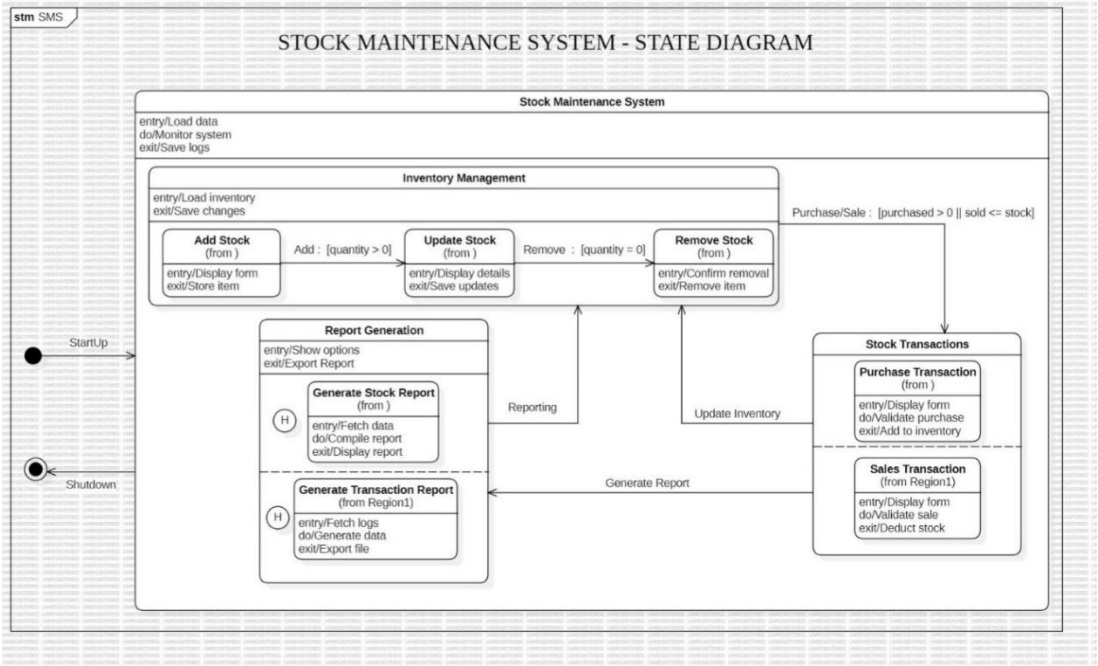


Figure 13 advanced

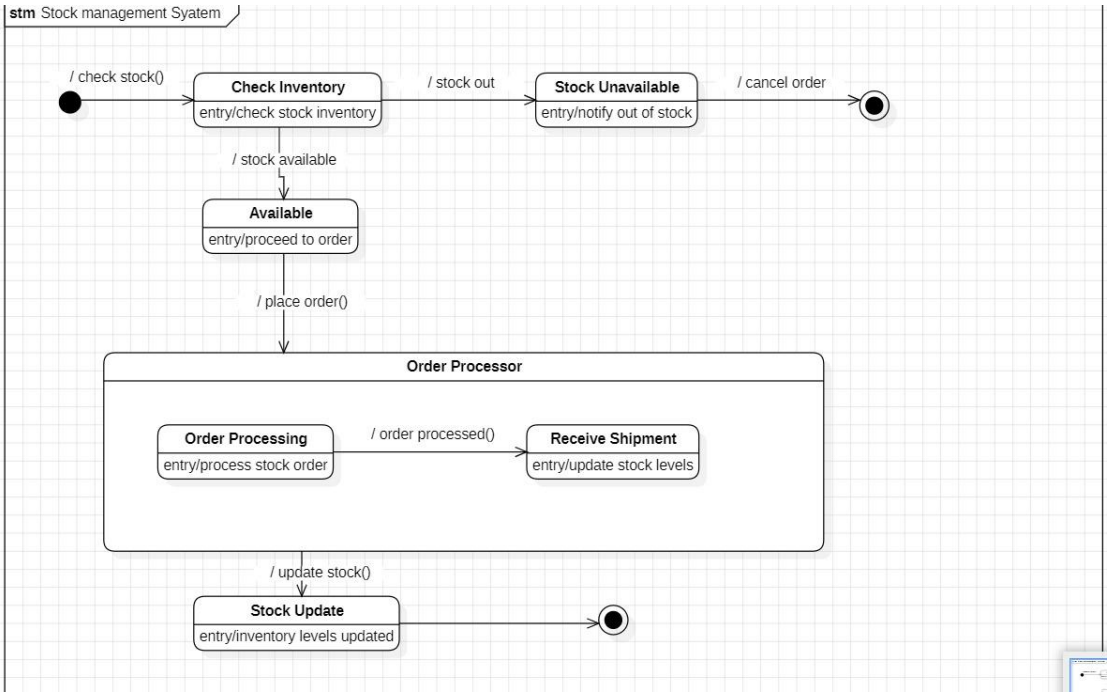


Figure 14 simple

Key States and Transitions

1. **Stock Maintenance System**
    - Entry action: Loads data and monitors the system.
    - Exit action: Saves logs.
    - Transitions:
        - Inventory Management: Handles inventory operations.
        - Reporting: Generates various reports.
        - Shutdown: Terminates the system.
2. **Inventory Management**
    - Entry action: Loads inventory data.
    - Exit action: Saves changes.
    - Transitions:
        - Add Stock: Adds stock items to the inventory.
        - Update Stock: Modifies existing stock details.
        - Remove Stock: Removes stock items from the inventory.
3. **Add Stock**
    - Entry action: Displays a form for entering stock details.
    - Exit action: Stores the new item in the inventory.
4. **Update Stock**
    - Entry action: Displays details of the selected stock item.
    - Exit action: Saves updates to the inventory.
5. **Remove Stock**
    - Entry action: Confirms the removal of the selected stock item.
    - Exit action: Removes the item from the inventory.
6. **Reporting**
    - Entry action: Shows report generation options.
    - Transitions:
        - Generate Stock Report: Creates a report of the current stock inventory.
        - Generate Transaction Report: Generates a report of past stock transactions.

7. **Generate Stock Report**
    - Entry action: Fetches stock data.
    - Do action: Compiles the report data.
    - Exit action: Displays the report.
8. **Generate Transaction Report**
    - Entry action: Fetches transaction logs.
    - Do action: Generates the report data.
    - Exit action: Exports the report file.
9. **Stock Transactions**
    - Transitions:
        - Purchase Transaction: Handles the addition of stock due to purchases.
        - Sales Transaction: Handles the deduction of stock due to sales.
10. **Purchase Transaction**

- Entry action: Displays a form for entering purchase details.
- Do action: Validates the purchase information.
- Exit action: Adds the purchased stock to the inventory.

11. **Sales Transaction**

- Entry action: Displays a form for entering sales details.
- Do action: Validates the sale information.
- Exit action: Deducts the sold stock from the inventory.

3. USE CASE DIAGRAM

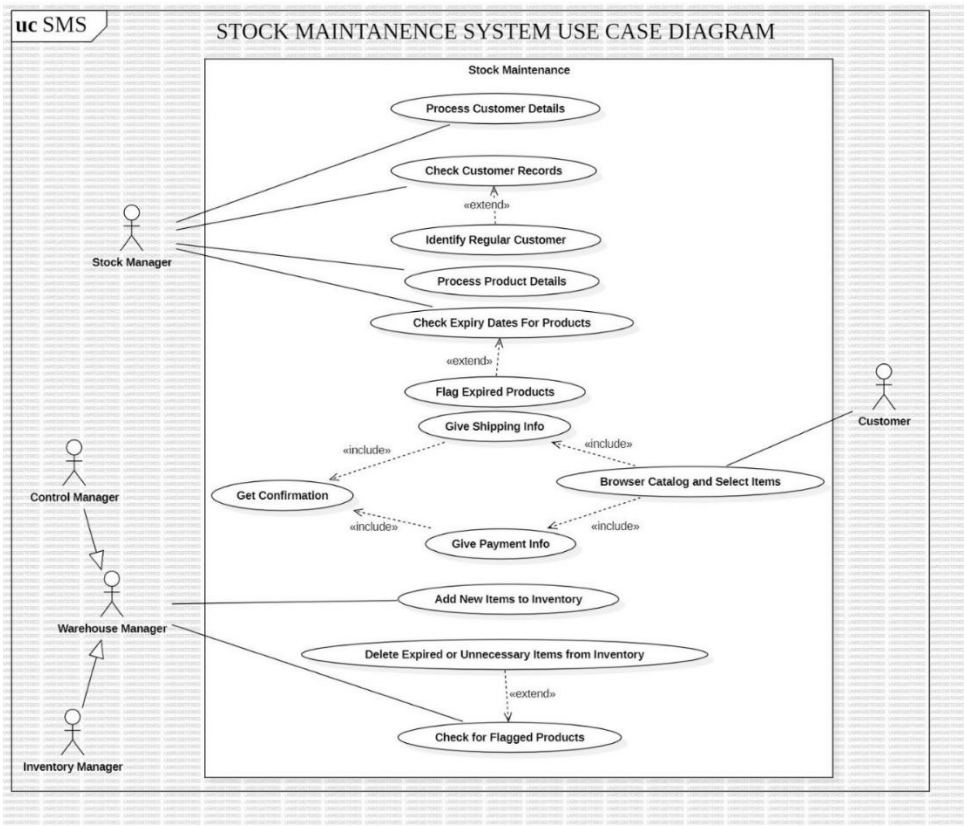


Figure 15 advanced

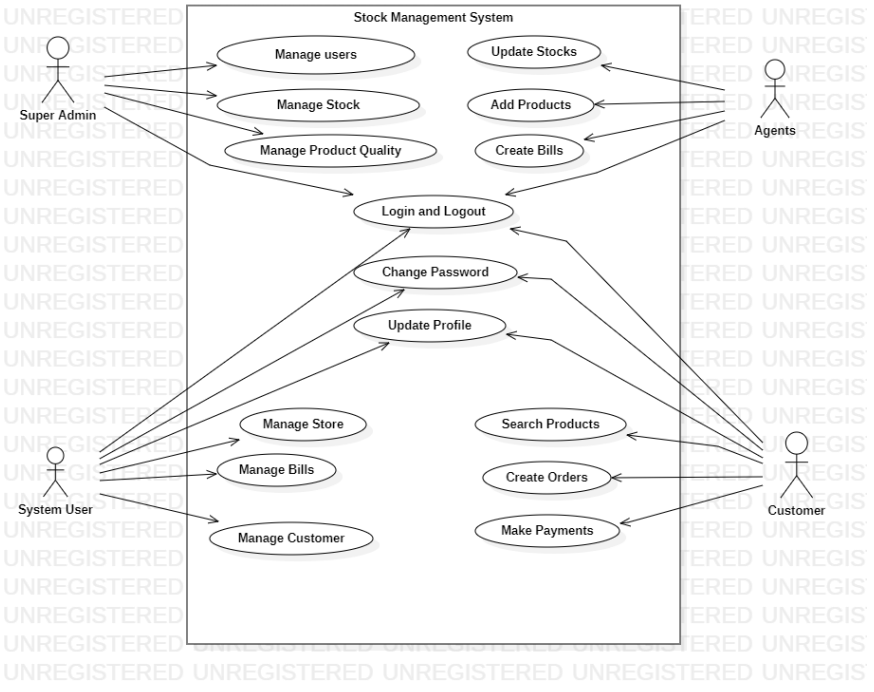


Figure 16 simple

Actors:

- **Super Admin:** This actor has the highest level of privileges and can perform all system administration tasks.
- **Agents:** These actors have specific permissions and can perform tasks related to stock management, such as adding products, creating bills, and managing orders.
- **System User:** This actor represents users who can interact with the system for various purposes, such as searching for products and creating orders.
- **Customer:** This actor represents end-users who purchase products from the system.

Use Cases:

- **Manage Users:** This use case allows the Super Admin to manage user accounts, including creating, modifying, and deleting user roles and permissions.
- **Update Stocks:** This use case enables the management of stock levels, including receiving shipments, adjusting inventory, and handling stock adjustments.
- **Manage Stock:** This use case allows for various stock-related operations, such as tracking stock levels, setting reorder points, and managing stock locations.
- **Manage Product Quality:** This use case enables the management of product quality, including quality checks, returns, and handling defective products.
- **Add Products:** This use case allows the authorized actors to add new products to the system, including product details, pricing, and inventory.
- **Create Bills:** This use case allows authorized actors to create bills for customers, including calculating taxes and discounts.
- **Login and Logout:** This use case enables all actors to log in to the system and log out securely.
- **Change Password:** This use case allows users to change their passwords for security purposes.
- **Update Profile:** This use case allows users to update their personal information.

- **Manage Store:** This use case allows for managing store-related information, such as store locations, contact details, and operating hours.
- **Manage Bills:** This use case enables the management of bills, including viewing bill history, generating reports, and handling payment issues.
- **Manage Customer:** This use case allows for managing customer information, including adding new customers, updating customer details, and managing customer orders.
- **Search Products:** This use case allows users to search for products based on various criteria, such as product name, category, price, and availability.
- **Create Orders:** This use case allows customers to create orders for products.
- **Make Payments:** This use case allows customers to make payments for their orders.

Relationships:

- No explicit "include" or "extend" relationships are shown in this diagram. However, it's possible that some use cases might include sub-use cases or have extensions under specific conditions. For example, "Manage Stock" might include sub-use cases like "Adjust Inventory" and "Set Reorder Points."

4. SEQUENCE DIAGRAM

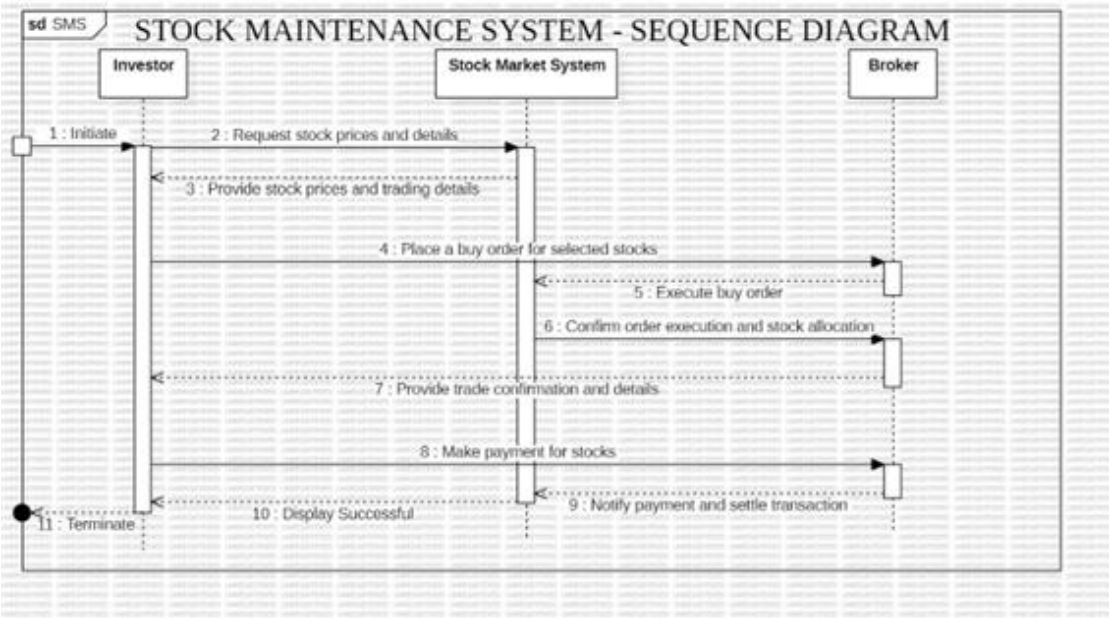


Figure 17 Advanced

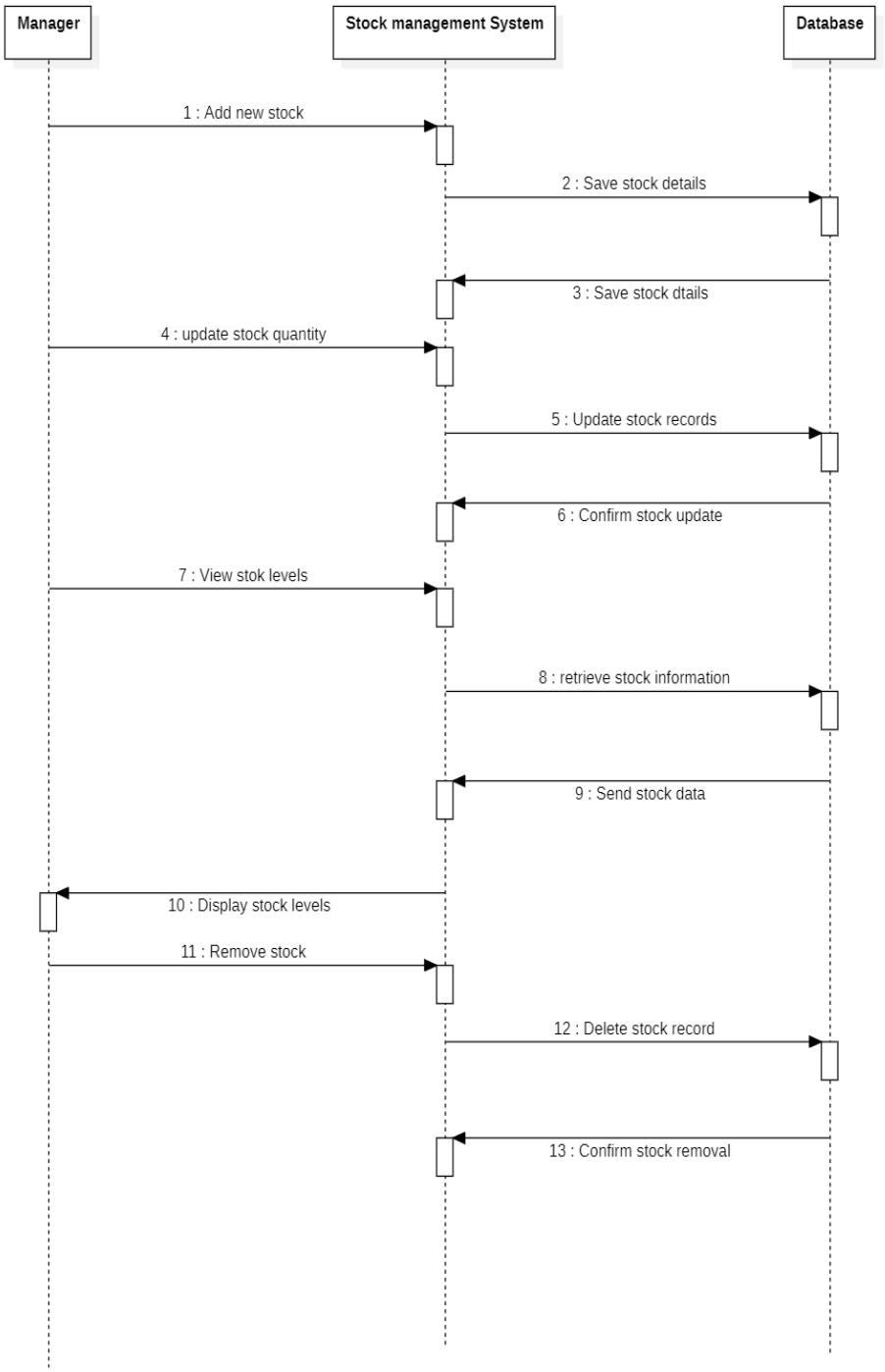


Figure 18 Simple

Overview:

1. Add New Stock: The manager initiates the process by adding a new stock item to the system.

2. Save Stock Details: The system saves the details of the new stock item in the database.

3. Save Stock Details: The system saves the details of the new stock item in the database.

4. Update Stock Quantity: The manager updates the quantity of an existing stock item.

5. Update Stock Records: The system updates the stock records in the database to reflect the new quantity.

6. Confirm Stock Update: The system confirms the successful update of the stock quantity to the manager.

7. View Stock Levels: The manager views the current stock levels for various items.

8. Retrieve Stock Information: The system retrieves the stock information from the database.

9. Send Stock Data: The system sends the stock data to the manager.

10. Display Stock Levels: The system displays the stock levels to the manager.

11. Remove Stock: The manager removes a stock item from the system.

12. Delete Stock Record: The system deletes the corresponding stock record from the database.

13. Confirm Stock Removal: The system confirms the successful removal of the stock item to the manager

5. ACTIVITY DIAGRAM

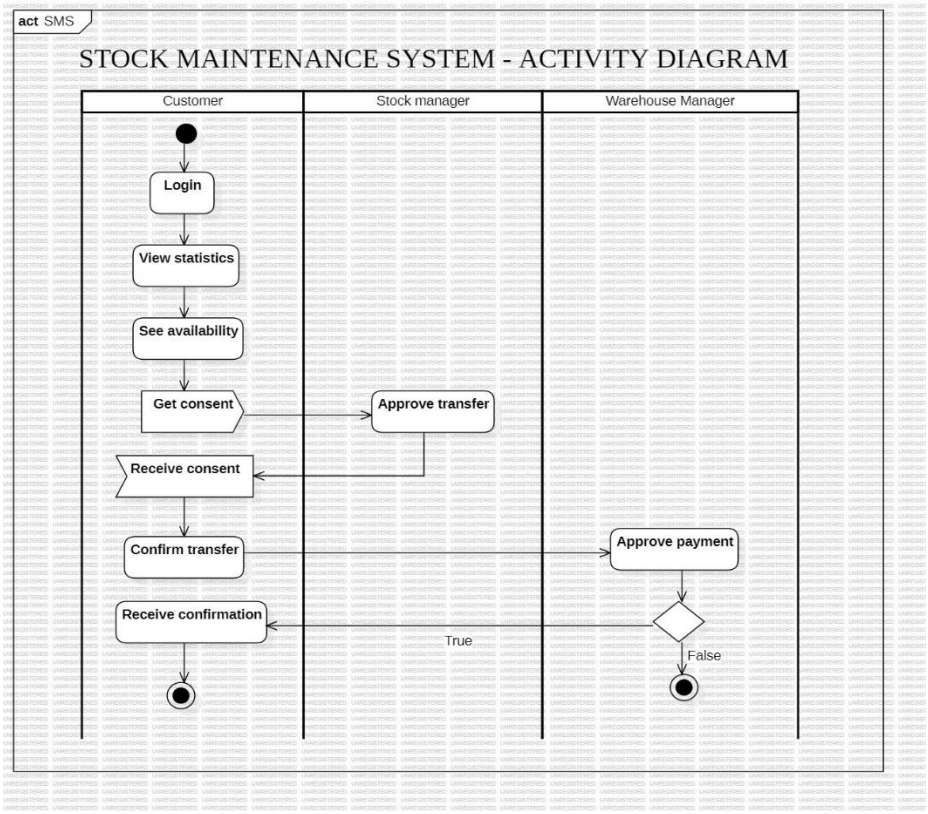


Figure 19 Advanced

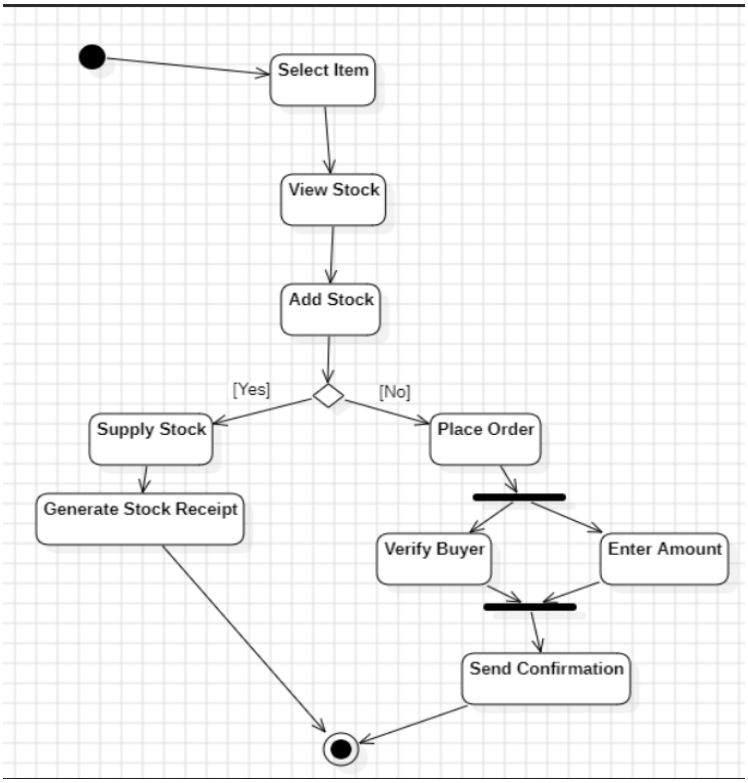
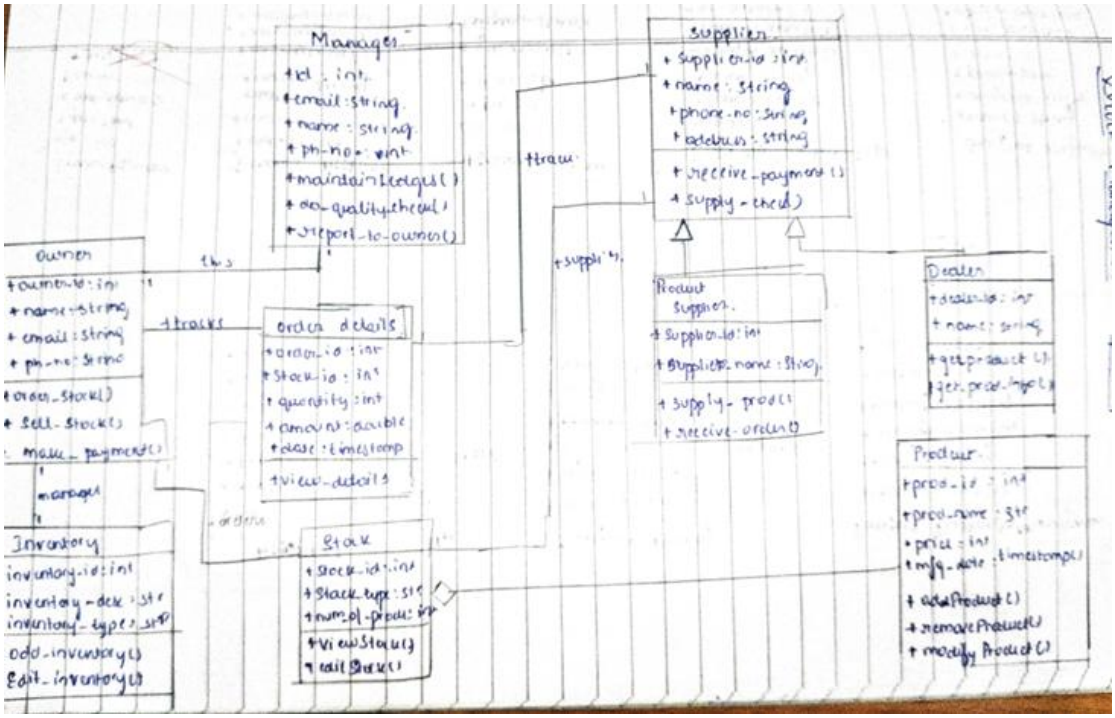
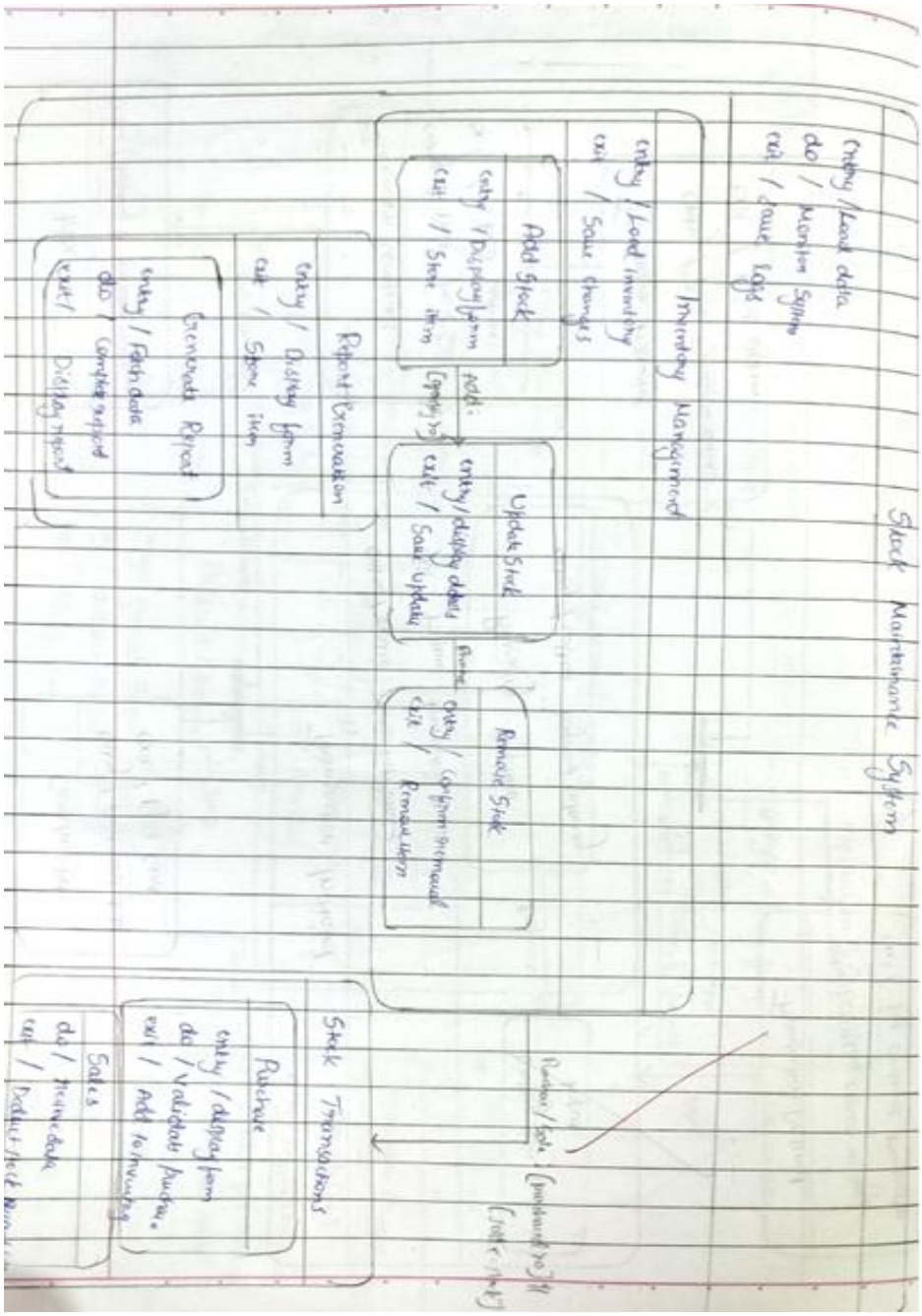


Figure 20 Simple

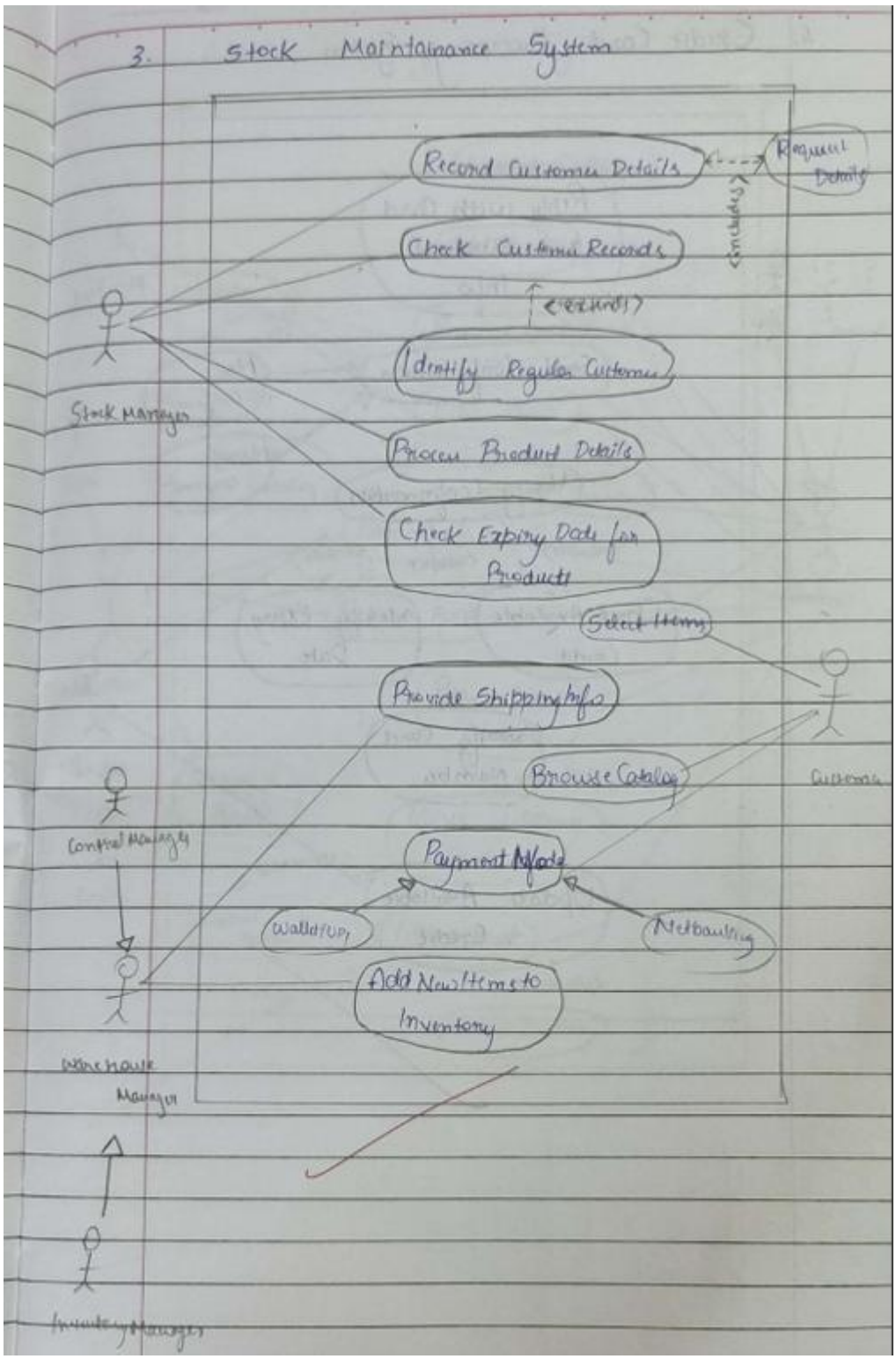
1. CLASS DIAGRAM



2. STATE DIAGRAM

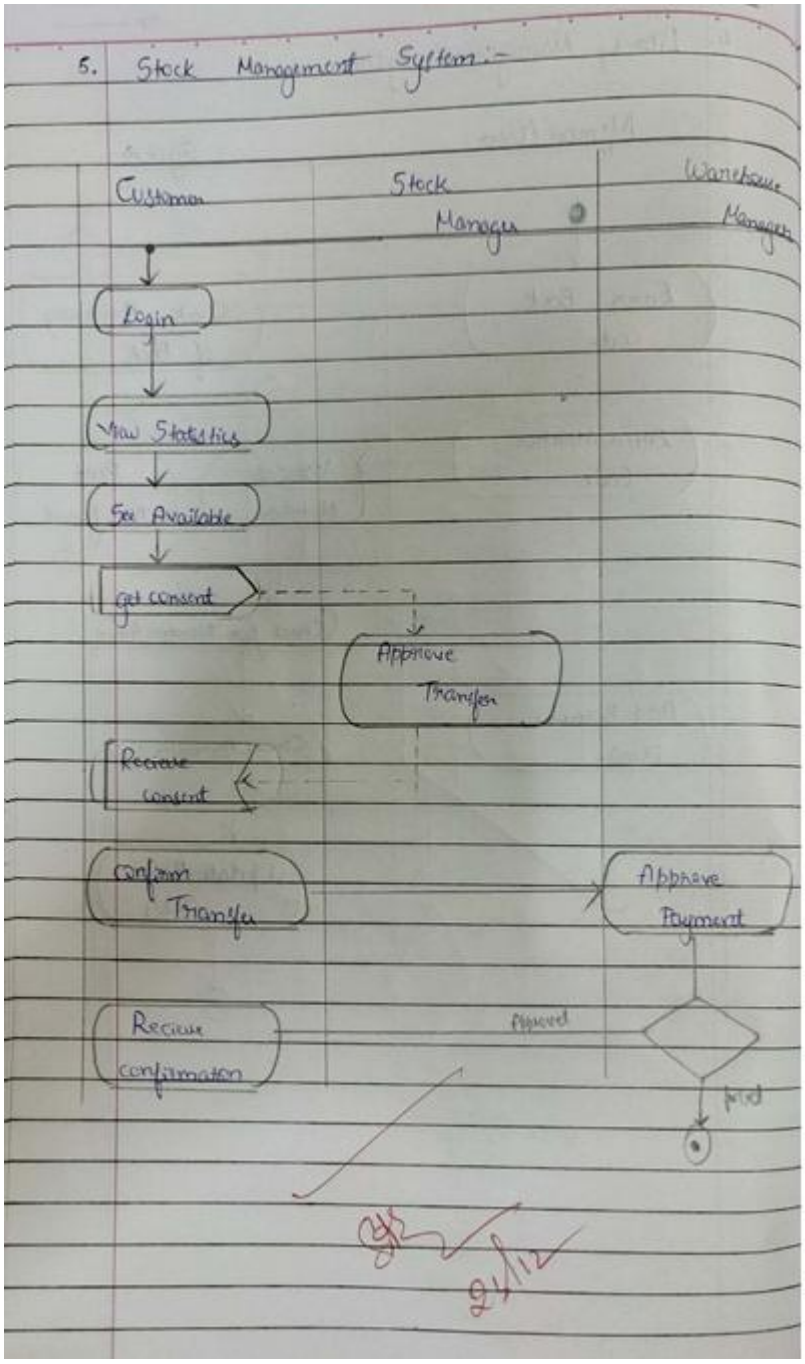


3. USE CASE DIAGRAM



4. SEQUENCE DIAGRAM

5. ACTIVITY DIAGRAM



CHAPTER – 5

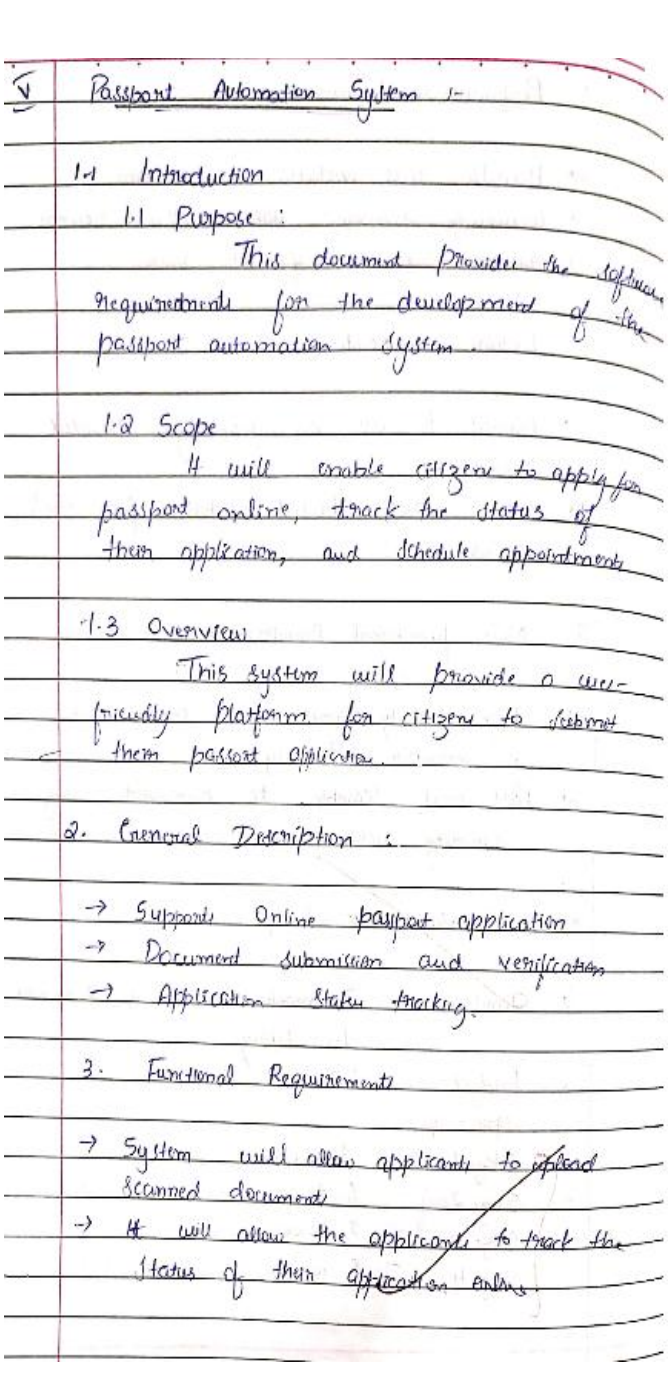
PASSPORT AUTOMATION SYSTEM

PROBLEM STATEMENT

The objective of this project is to design and implement a comprehensive Passport Automation System to streamline the processes of passport issuance, renewal, and verification. The system will address several key functionalities to improve efficiency and user experience.

- The system will provide an online platform for applicants to submit passport applications, update personal details, and upload required documents, simplifying the application process.
- It will facilitate real-time appointment scheduling for document verification, biometrics, and interviews at passport offices, ensuring timely processing.
- Automated document verification will cross-check submitted documents and applicant information with government databases to ensure accuracy and authenticity.
- The system will enable efficient processing of applications and provide real-time status updates to applicants at every stage, enhancing transparency.

SOFTWARE REQUIREMENTS SPECIFICATION



Passport Automation System :-

1. Introduction

1.1 Purpose :

This document Provides the software requirements for the development of the passport automation system.

1.2 Scope

It will enable citizens to apply for passport online, track the status of their application, and schedule appointments

1.3 Overview

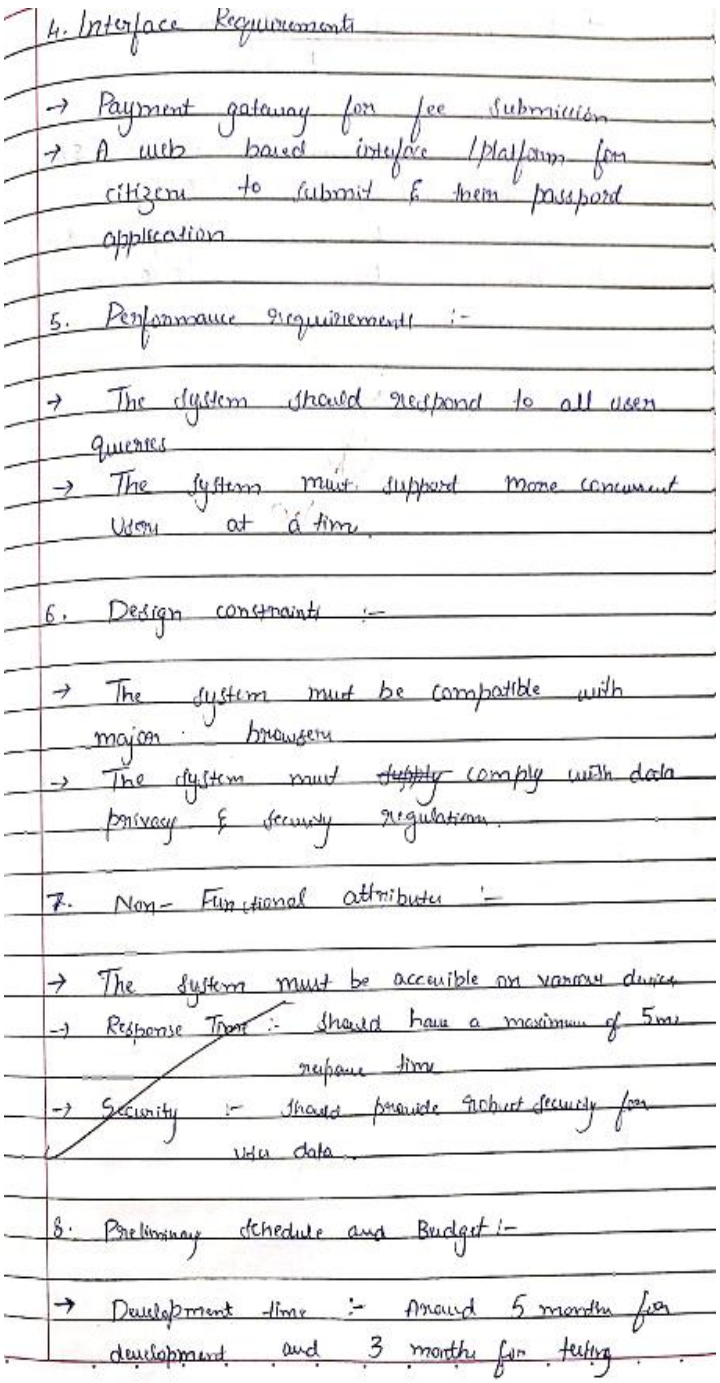
This system will provide a user-friendly platform for citizens to submit their passport application.

2. General Description :

→ Supports Online passport application
→ Document submission and verification
→ Application status tracking.

3. Functional Requirements

→ System will allow applicants to upload scanned documents
→ It will allow the applicants to track the status of their application online.



4. Interface Requirements

→ Payment gateway for fee submission
→ A web based interface / platform for citizens to submit & their passport application

5. Performance requirements :-

→ The system should respond to all user queries
→ The system must support more concurrent users at a time.

6. Design constraints :-

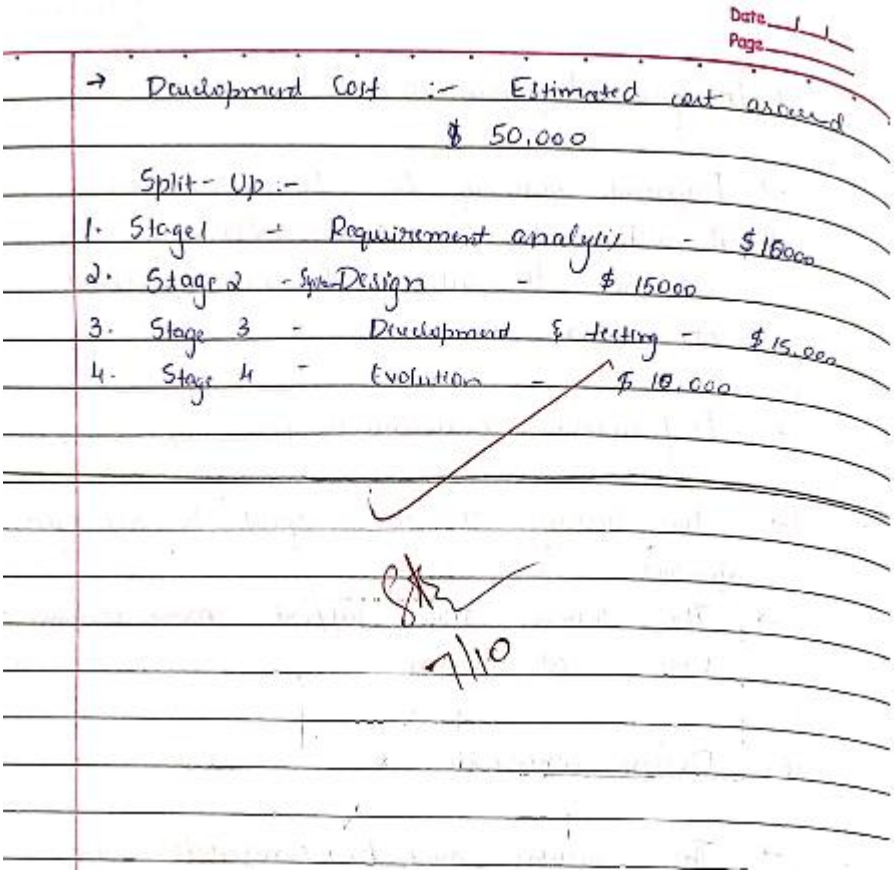
→ The system must be compatible with major browsers
→ The system must ~~supply~~ comply with data privacy & security regulations.

7. Non-Functional attributes :-

→ The system must be accessible on various devices
→ Response Time :- Should have a maximum of 5ms response time
→ Security :- Should provide robust security for user data.

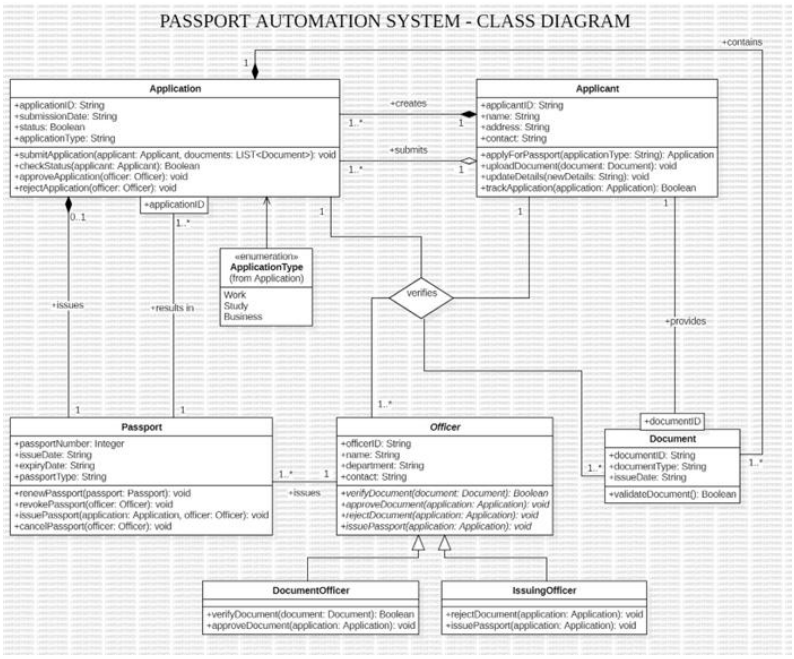
8. Preliminary schedule and Budget :-

→ Development time :- Around 5 months for development and 3 months for testing.


Date
Page
→ Development Cost :- Estimated cost accord
$ 50,000
Split - Up :-
1. Stage1 - Requirement analysis - $16000
2. Stage 2 - System Design - $ 15000
3. Stage 3 - Development & testing - $15,000
4. Stage 4 - Evolution - $ 10,000
7/10

UML DIAGRAMS

1. CLASS DIAGRAM



Key Classes and their Relationships:

1. **Applicant:**
    - Attributes: applicantID, name, address, contact
    - Methods: applyForPassport, uploadDocument, trackApplication
2. **Application:**
    - Attributes: applicationID, submissionDate, status, applicationType
    - Methods: submitApplication, checkStatus, approveApplication, rejectApplication
3. **Passport:**
    - Attributes: passportNumber, issueDate, expiryDate, passportType
    - Methods: renewPassport, revokePassport, cancelPassport
4. **Officer:**
    - Attributes: officerID, name, department, contact
    - Methods: verifyDocument, approveDocument, rejectDocument, issuePassport

5. **Document:**
    - Attributes: documentID, documentType, category, date, validated
    - Methods: validateDocument
6. **IssuingOfficer:**
    - Inherits from Officer
    - Methods: issuePassport
7. **DocumentOfficer:**
    - Inherits from Officer
    - Methods: verifyDocument, approveDocument, rejectDocument
8. **ApplicationType:**
    - Attributes: applicationType

Relationships:

- **Applicant submits Application:** An applicant can submit multiple applications.
- **Application contains Documents:** An application can have multiple documents.
- **ApplicationType is used in Application:** An application has an associated application type.
- **Officer verifies Documents:** An officer verifies the documents submitted with an application.
- **Officer issues Passport:** An officer issues passports.
- **IssuingOfficer inherits from Officer:** An issuing officer is a type of officer.
- **DocumentOfficer inherits from Officer:** A document officer is a type of officer.

2. STATE DIAGRAM

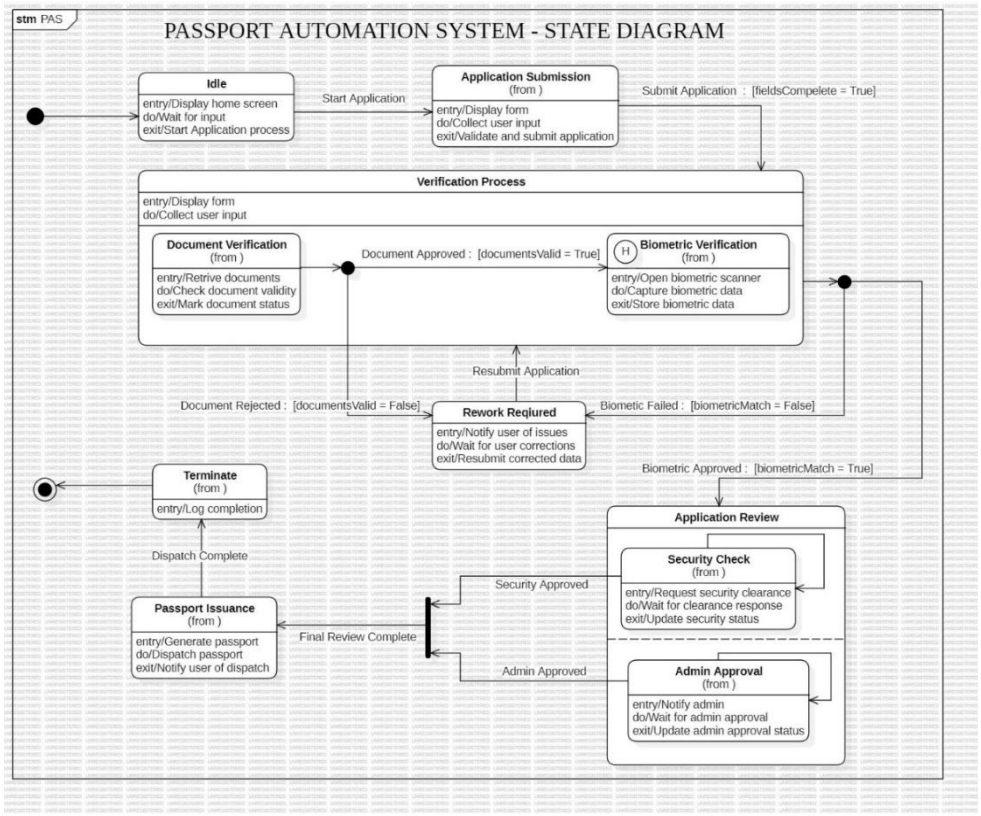


Figure 21 advanced

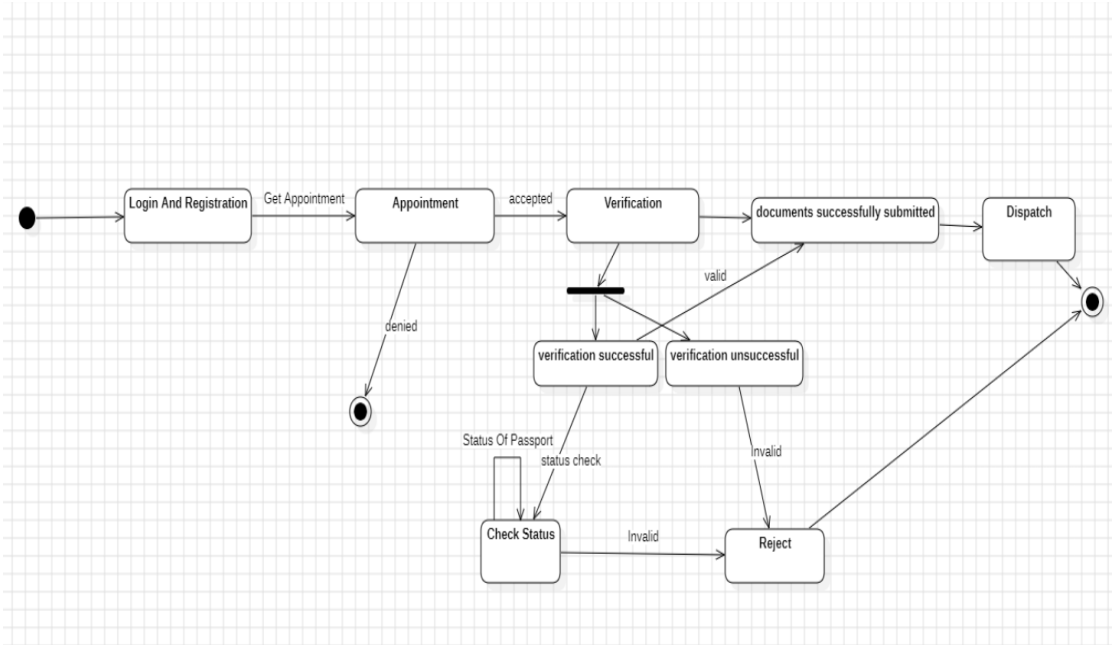


Figure 22 simple

Key States and Transitions

1. **Idle:**
    - Entry action: Displays the home screen.
    - Do action: Waits for user input.
    - Exit action: Starts the application process.
2. **Start Application:**
    - Entry action: Displays the application form.
    - Do action: Collects user input.
    - Exit action: Validates and submits the application.
3. **Application Submission**
    - Transition: Submit Application (fieldsComplete = True): Proceeds to the "Verification Process" state.
4. **Verification Process**
    - Entry action: Displays the verification form.
    - Do action: Collects user input.
    - Transitions:
        - Document Approved (documentsValid = True): Proceeds to "Biometric Verification".
        - Document Rejected (documentsValid = False): Proceeds to "Resubmit Application".
5. **Document Verification**
    - Entry action: Retrieves documents.
    - Do action: Checks document validity.
    - Exit action: Marks document status.
6. **Resubmit Application**
    - Entry action: Notifies the user of issues.
    - Do action: Waits for user corrections.
    - Exit action: Resubmits corrected data.
7. **Biometric Verification**
    - Entry action: Opens biometric scanner.
    - Do action: Captures biometric data.
    - Transitions:
        - Biometric Approved (biometricMatch = True): Proceeds to "Application Review".

        - Biometric Failed (biometricMatch = False): Proceeds to "Resubmit Application".

8. **Application Review**
    - Transitions:
        - Security Approved: Proceeds to "Admin Approval".
9. **Security Check**
    - Entry action: Requests security clearance.
    - Do action: Waits for clearance response.
    - Exit action: Updates security status.
10. **Admin Approval**

- Entry action: Notifies the admin.
- Do action: Waits for admin approval.
- Exit action: Updates admin approval status.

11. **Final Review Complete**

- Transition: Admin Approved: Proceeds to "Passport Issuance".

12. **Passport Issuance**

- Entry action: Generates the passport.
- Do action: Dispatches the passport.
- Exit action: Notifies the user of dispatch.

3. USE CASE DIAGRAM

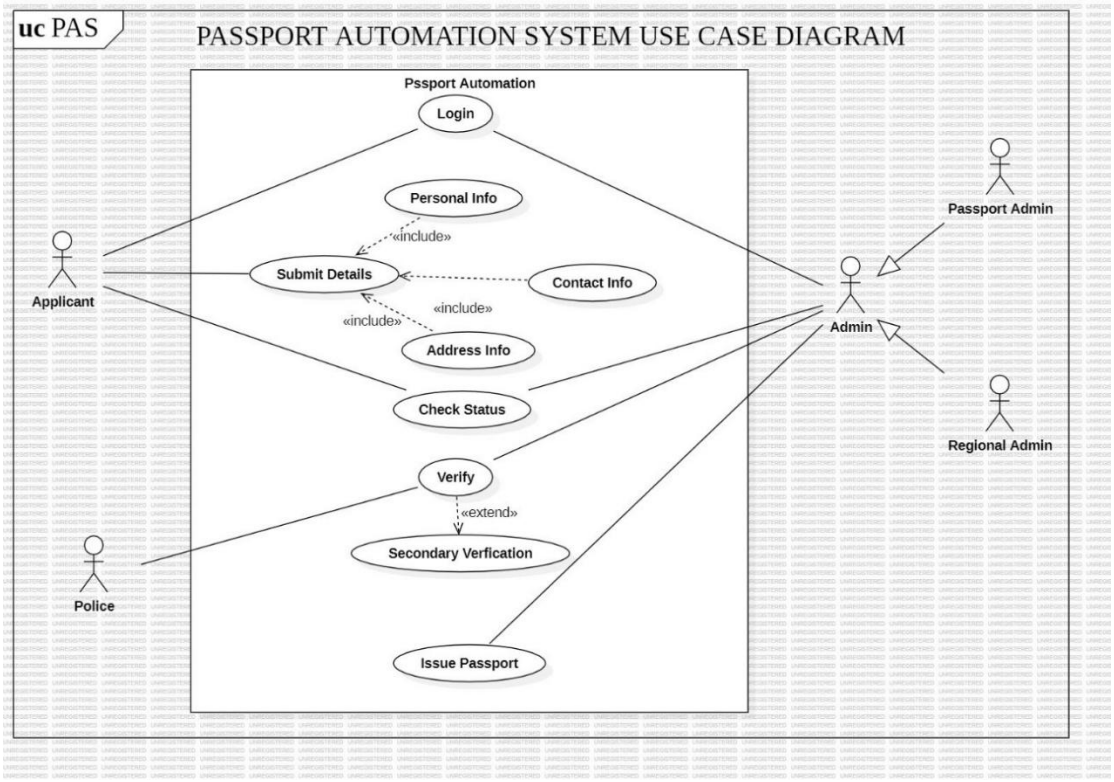


Figure 23 Advanced

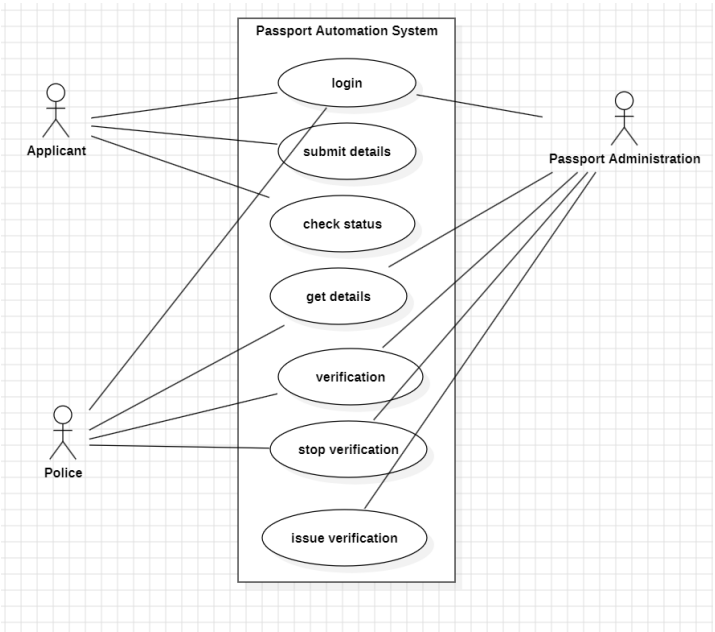


Figure 24 Simple

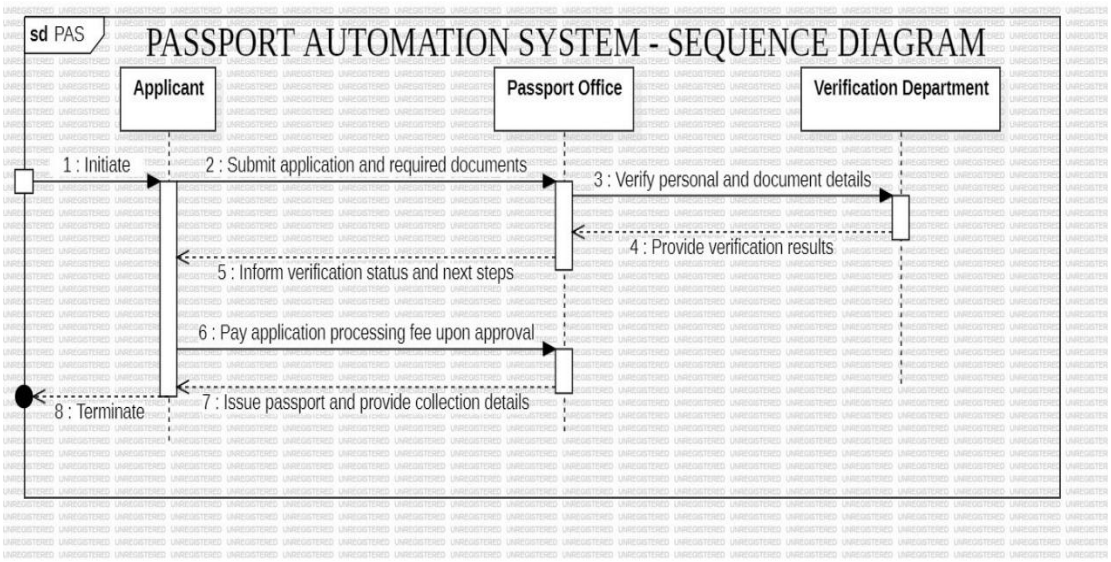
Actors:

- **Applicant:** The person applying for a passport.
- **Passport Admin:** Responsible for managing the passport application process.
- **Admin:** The administrator of the system, with higher level access and control.
- **Regional Admin:** An administrator with regional level access and responsibilities.
- **Police:** Involved in verifying the applicant's background and address.

Use Cases:

- **Passport Automation:** This is the main use case, encompassing all the functions related to passport application processing.
    - **Login:** The initial step for all users to access the system.
    - **Check Status:** Checking the status of the passport application.
    - **Verify:**
        - **Extends:** This use case can be extended to include additional verification steps, such as background checks or document verification.
        - **Secondary Verification:** A specialized verification process, possibly involving additional agencies like the police.
    - **Issue Passport:** The final step where the passport is issued to the applicant.

4. SEQUENCE DIAGRAM



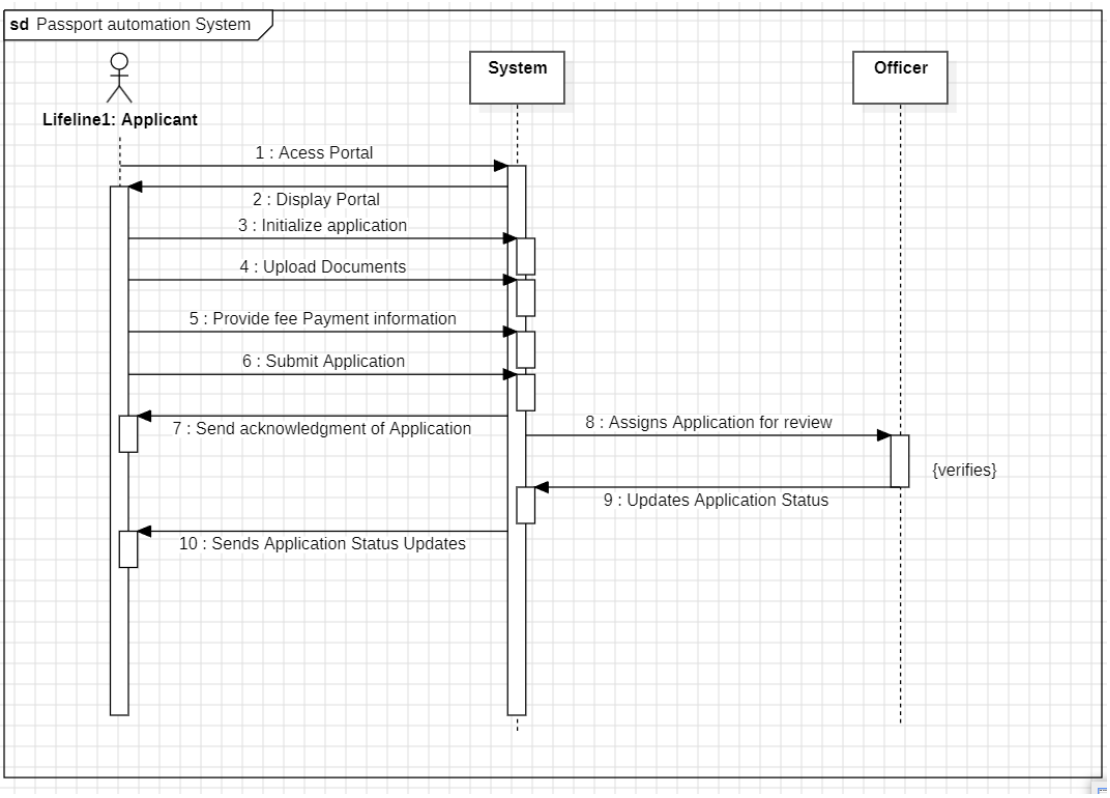


Figure 25 Simple Sequence diagram

Key Entities and Interactions:

1. **Applicant:**
    - Initiates the process by submitting an application and required documents (message 2).
    - Receives verification results and next steps (message 5).
    - Pays the application processing fee upon approval (message 6).
    - Receives the passport and collection details (message 7).
    - Terminates the interaction (message 8).
2. **Passport Office:**
    - Receives the application and documents (message 2).
    - Sends the documents to the Verification Department for verification (message 3).
    - Receives verification results (message 4).
    - Informs the applicant of verification status and next steps (message 5).
    - Receives the application processing fee (message 6).
    - Issues the passport and provides collection details to the applicant (message 7).
3. **Verification Department:**
    - Receives documents from the Passport Office (message 3).
    - Verifies personal and document details.
    - Sends verification results to the Passport Office (message 4).

5. ACTIVITY DIAGRAM

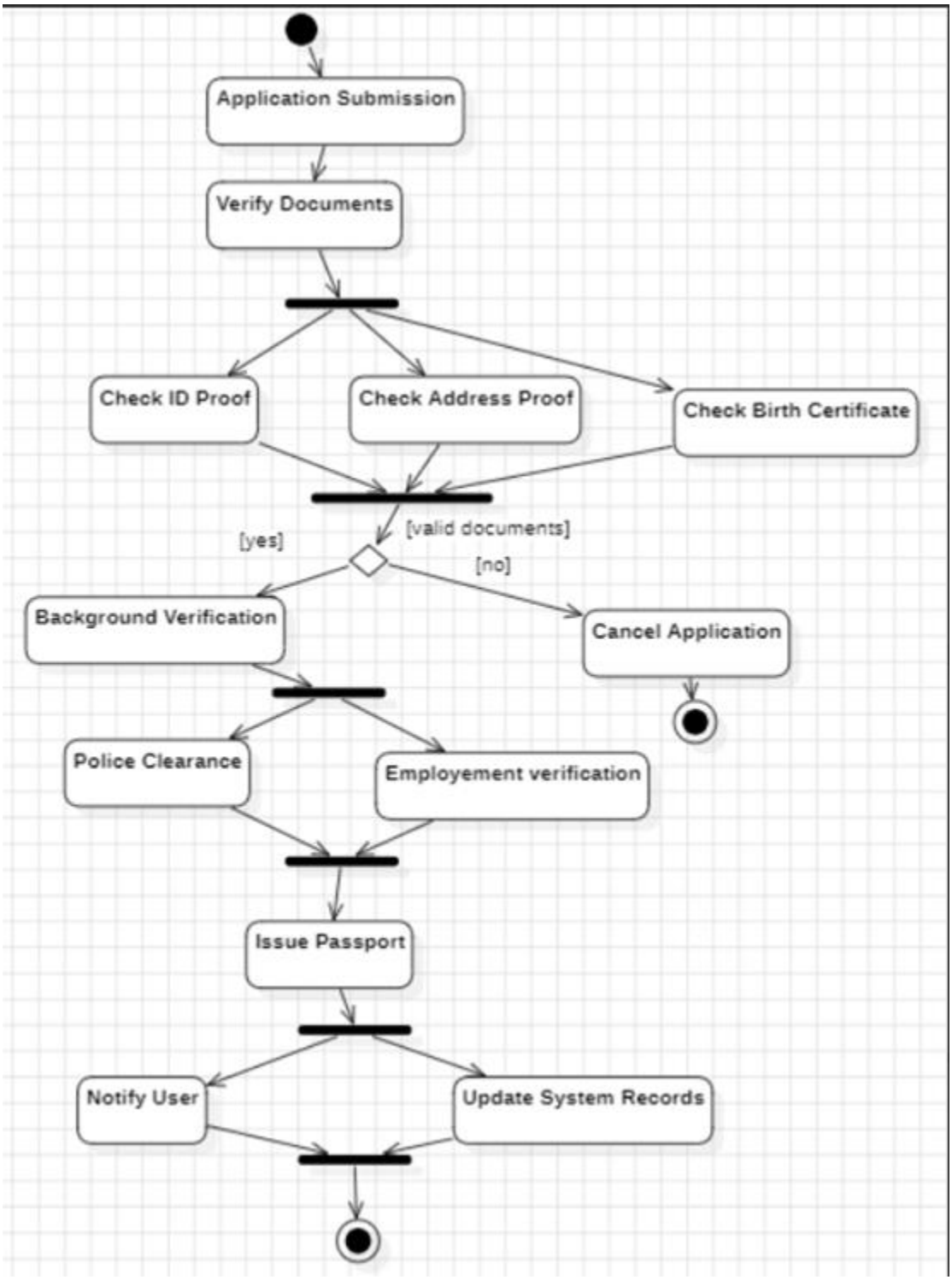


Figure 26 Simple Activity diagram

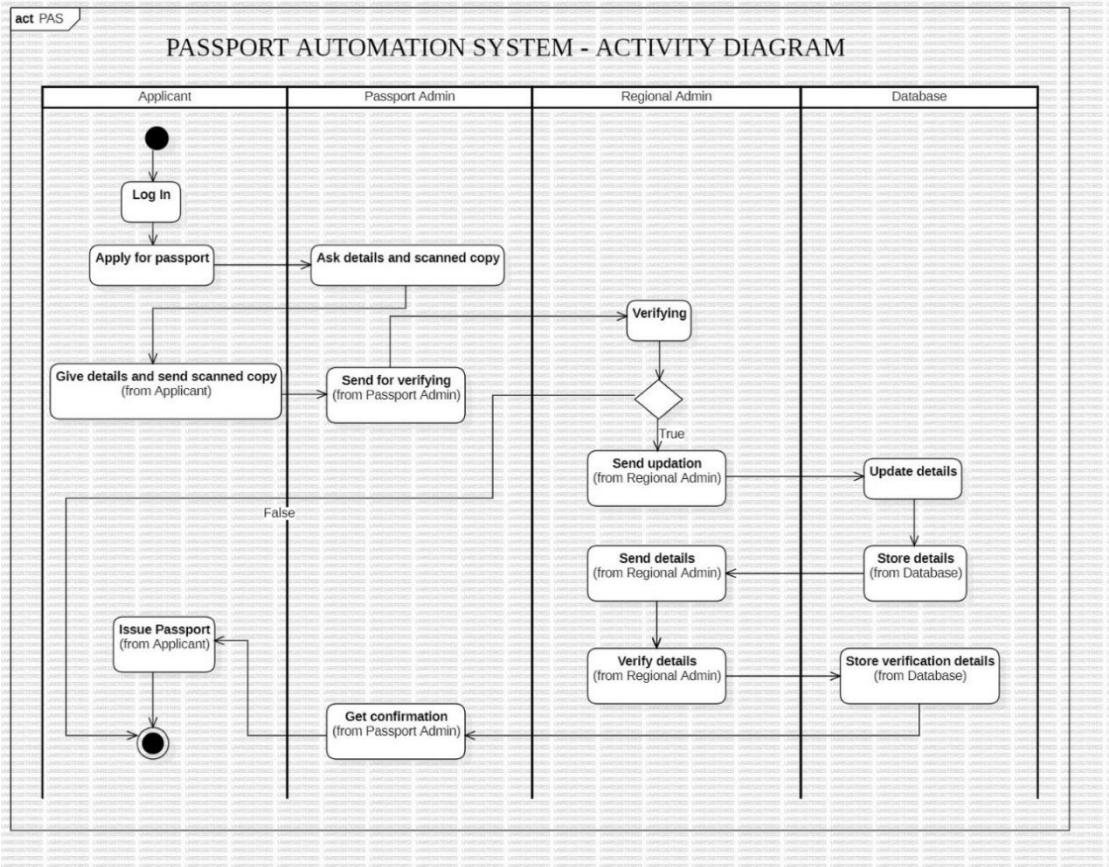


Figure 27 Advanced activity diagram

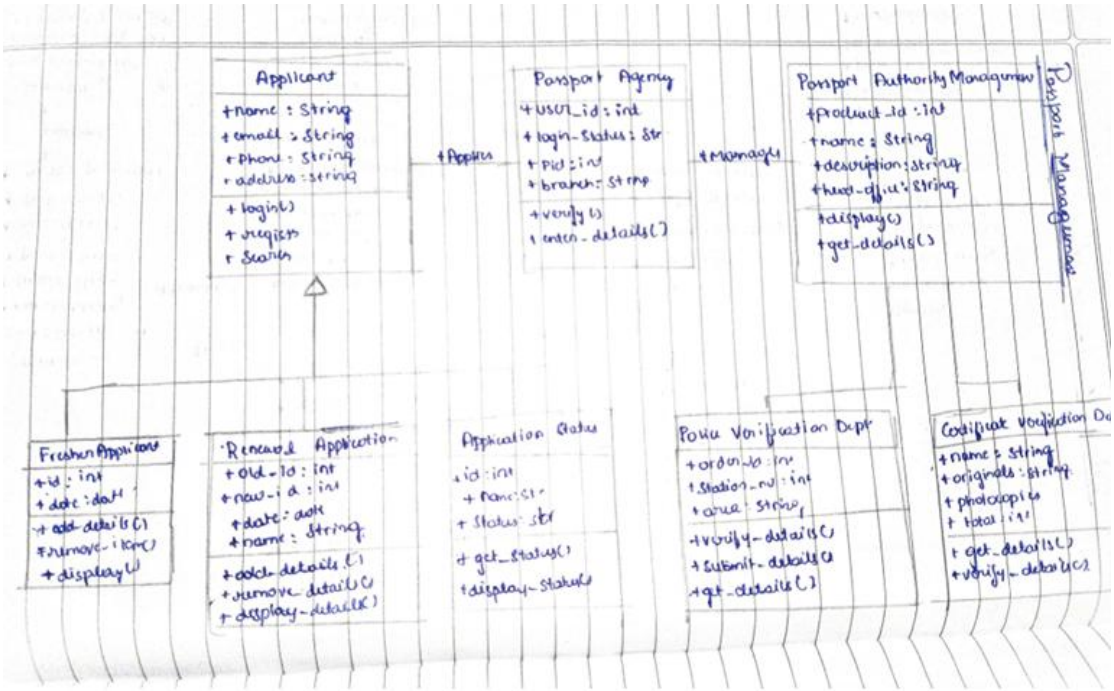
Key Activities and Flows:

1. **Applicant:**
    - Logs in to the system.
    - Applies for a passport.
    - Gives details and sends scanned copies of documents to the Passport Admin.
    - Issues the passport upon receiving confirmation.
2. **Passport Admin:**
    - Asks for details and scanned copies from the Applicant.
    - Sends the details for verification to the Regional Admin.
    - Receives confirmation from the Regional Admin.
    - Gets confirmation from the Applicant.
3. **Regional Admin:**

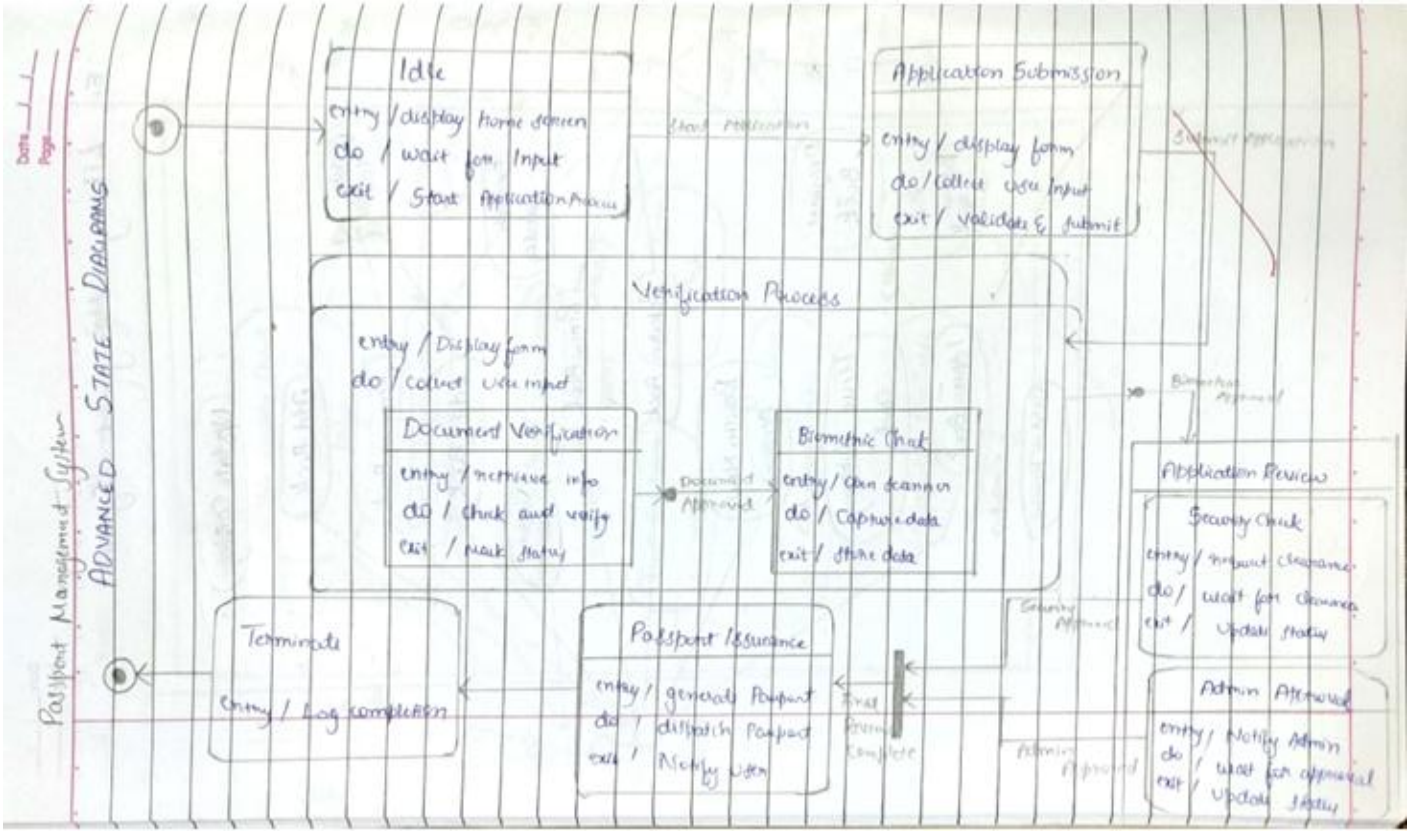
- Verifies the received details.
- Sends updates or details to the Database based on the verification outcome.
- Sends details to the Passport Admin upon completion.

4. **Database:**
    - Stores details received from the Regional Admin.
    - Stores verification details.

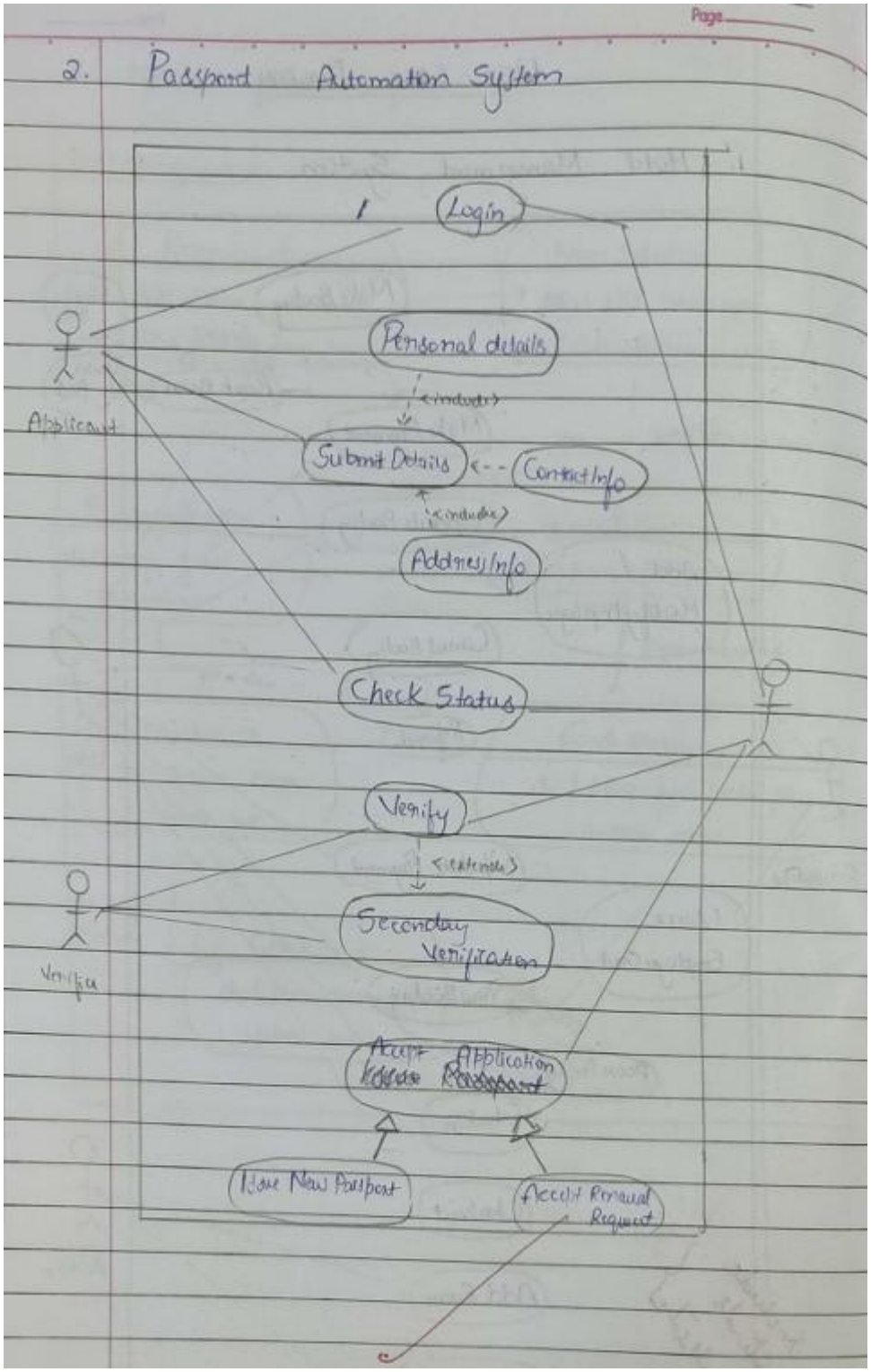
1. CLASS DIAGRAM



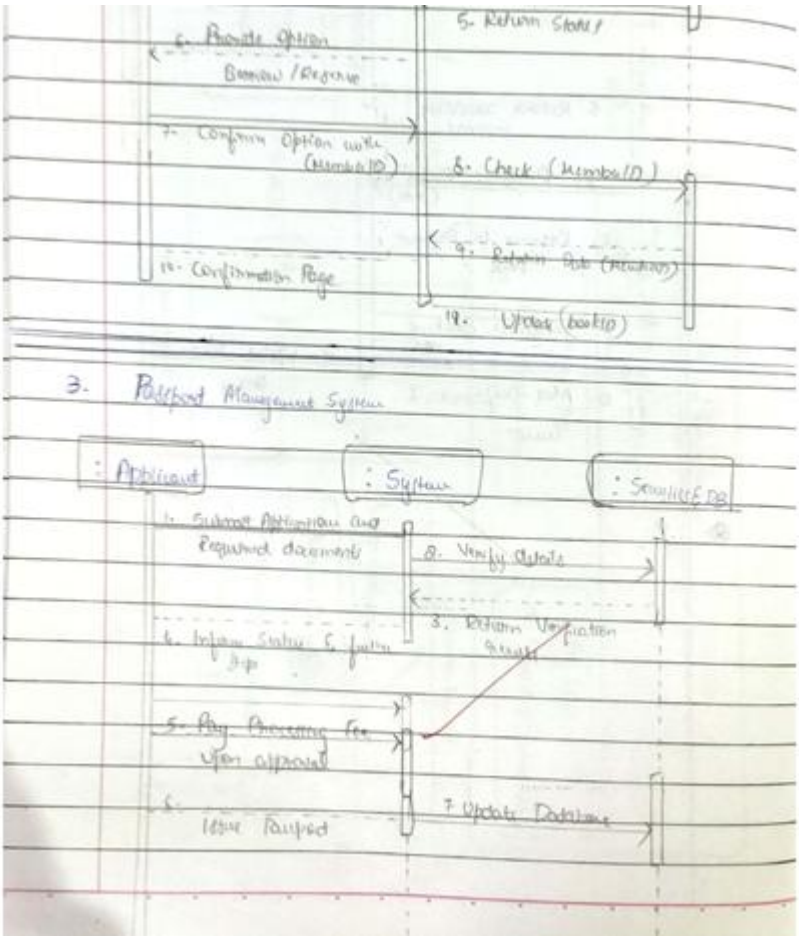
2. STATE DIAGRAM



3. USE CASE DIAGRAM



4. SEQUENCE DIAGRAM



5. ACTIVITY DIAGRAM