

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



Lab Record

**Object Oriented Modeling**

*Submitted in partial fulfillment for the 5<sup>th</sup> Semester Laboratory*

Bachelor of Engineering  
in  
Computer Science and Engineering

*Submitted by:*

**Ankit Singh Bhatti**

1BM22CS353

Department of Computer Science and Engineering  
B.M.S. College of Engineering  
Bull Temple Road, Basavanagudi, Bangalore 560 019  
Mar-June 2024

**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 Ankit Singh Bhatti (1BM22CS353) during the 5<sup>th</sup> Semester Oct 24 - Jan 25.

Signature of the Faculty Incharge:

Prof. Prameetha Pai  
Assistant Professor  
Department of Computer Science and Engineering  
B.M.S. College of Engineering, Bangalore

## Table of Contents

- |                               |
|-------------------------------|
| 1. Hotel Management System    |
| 2. Credit Card Processing     |
| 3. Library Management System  |
| 4. Stock Maintenance System   |
| 5. Passport Automation System |

GITHUB LINK: [https://github.com/AnkitSB19/OOMD\\_-1BM22CS353](https://github.com/AnkitSB19/OOMD_-1BM22CS353)

# 1. Hotel Management System

## Problem Statement:

Design a software system to support a computerized hotel management network, catering to a chain of hotels operating under a single brand. Each hotel in the chain maintains its own local computer to manage bookings, room availability, and customer data. Reception desks at individual hotels communicate directly with their local computer systems to handle check-ins, check-outs, and in-house services. Front-desk staff input guest and reservation details.

An online booking portal communicates with a central server that coordinates reservations across all hotels, ensuring real-time updates for room availability. The online system allows guests to search for hotels, check room availability, make reservations, and process payments securely. Additionally, guests should be able to manage their reservations and request additional services through the portal.

The system requires robust recordkeeping, scalability to support multiple hotels, and strict security measures to protect customer data and transactions. It must handle concurrent booking requests and avoid double-booking. Each hotel will provide and maintain its software for local operations, while you are tasked with designing the software for the central reservation system, the online booking portal, and the network connecting individual hotels to the central server. The cost of the shared system will be apportioned to hotels based on the volume of online reservations they receive.

---

# Software Requirement Specification Document

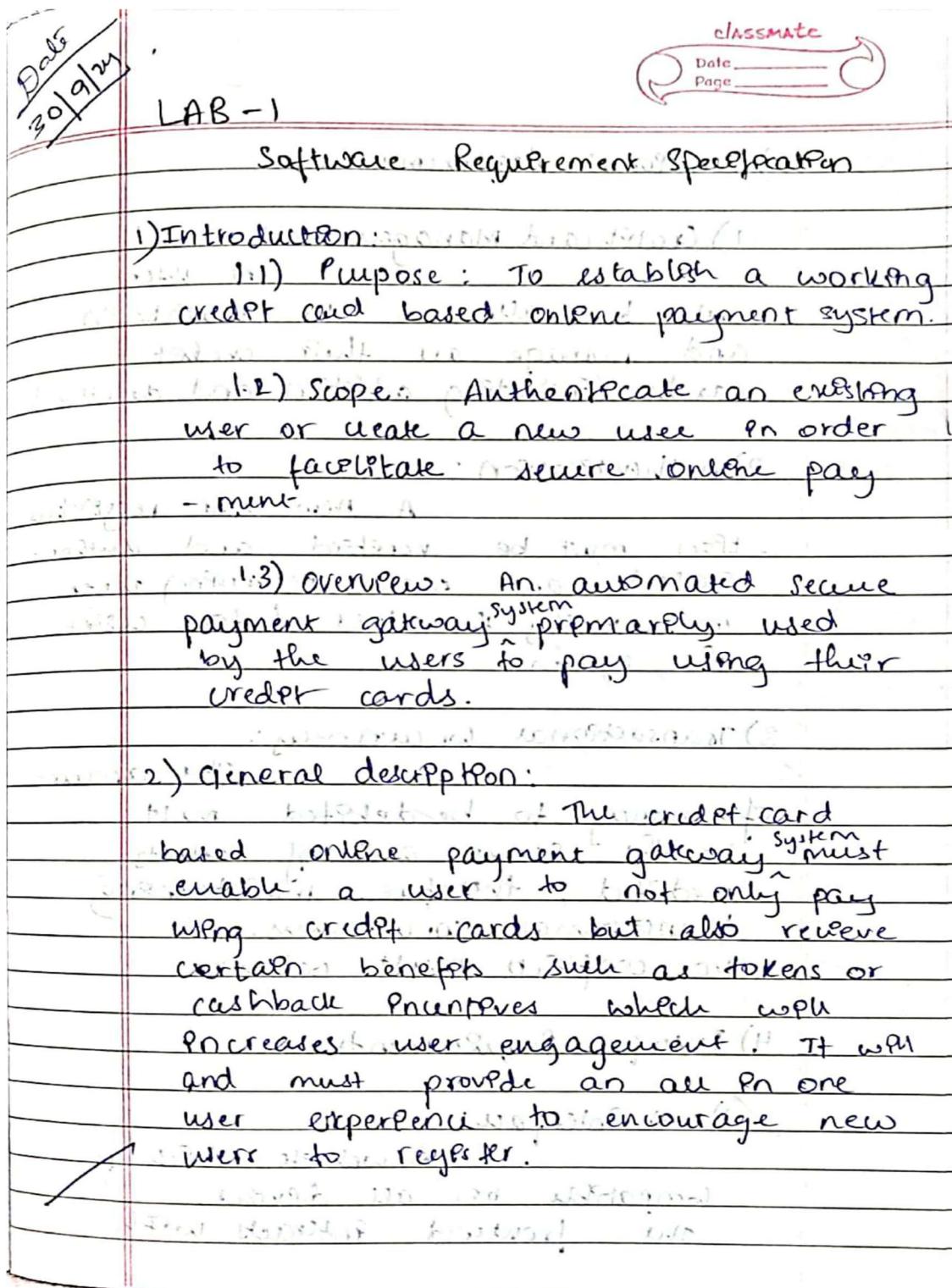


Fig 1.1

<p>3) Functional Requirements:</p> <ul style="list-style-type: none"> <li>1) Credit Card Management: A user must be allowed to maintain and manage all their credit cards. (including addition and deletion)</li> <li>2) Authentication: A new user registration must be verified and authenticated and each returning user must verify themselves before using the system.</li> <li>3) Transactional Concurrency: The amount of money to be debited must be the same amount of money credited; therefore maintaining transactional concurrency.</li> <li>4) Interface Requirements:</li> <li>1) Web Interface: A website which is compatible on all devices. The frontend interacts with</li> </ul>	<p>the user and the backend consists of a database which stores all previous transactions and details of the user.</p> <p>5) Performance Requirements:</p> <ul style="list-style-type: none"> <li>→ Speed: Transactions must be completed quickly and must not take a lot of time.</li> <li>→ Availability: Huge number of transactions must not cause the website server to crash.</li> </ul> <p>b) Design Constraints:</p> <p>The entire website must be made by adhering to the given RBI guidelines of the online payments and the software to be used for the same will must be implemented using a MEAN stack.</p> <p>7) Non-Functional Requirements:</p> <ul style="list-style-type: none"> <li>1) Security</li> <li>2) Compatibility</li> </ul>
--	---

Fig 1.2

<p>→ Data Integrity → Scalable → Improvable → Upgradable → Portable</p> <p>8) Preliminary Schedule and Budget:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; padding: 2px;">Week 1</td><td style="text-align: center; padding: 2px;">Week 2</td><td style="text-align: center; padding: 2px;">Week 3</td><td style="text-align: center; padding: 2px;">Week 4</td><td style="text-align: center; padding: 2px;">Week 5</td><td style="text-align: center; padding: 2px;">Week 6</td><td style="text-align: center; padding: 2px;">Week 7</td><td style="text-align: center; padding: 2px;">Week 8</td><td style="text-align: center; padding: 2px;">Week 9</td><td style="text-align: center; padding: 2px;">Week 10</td><td style="text-align: center; padding: 2px;">Week 11</td><td style="text-align: center; padding: 2px;">Week 12</td></tr> <tr> <td style="text-align: center; padding: 2px;">T1</td><td style="text-align: center; padding: 2px;">T2</td><td style="text-align: center; padding: 2px;">T3</td><td style="text-align: center; padding: 2px;">T4</td><td style="text-align: center; padding: 2px;">T5</td><td style="text-align: center; padding: 2px;">T6</td><td style="text-align: center; padding: 2px;">T7</td><td style="text-align: center; padding: 2px;">T8</td><td style="text-align: center; padding: 2px;">T9</td><td style="text-align: center; padding: 2px;">T10</td><td style="text-align: center; padding: 2px;">T11</td><td style="text-align: center; padding: 2px;">T12</td></tr> </table> <p>The entire workload is divided into 7 tasks and each task is to be completed as per the schedule given above.</p> <p>A total of 5 milestones must be achieved.</p> <p>In total, the entire project must be completed at the end of 10 weeks.</p>	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	<p>Budget:</p> <p>Total Lines of code: 220,000, to be written for the system.</p> <p>Organization average productivity: \$120/hr.</p> <p>Cost per Line of code: <math>220000 \times \\$120 = \\$264000</math></p> <p>Total cost: <math>220000 \times \\$120 = \\$264000</math></p> <p><del>Total cost = \$10333333.3</del></p>
Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12														
T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12														

Fig 1.3

## Class Diagram

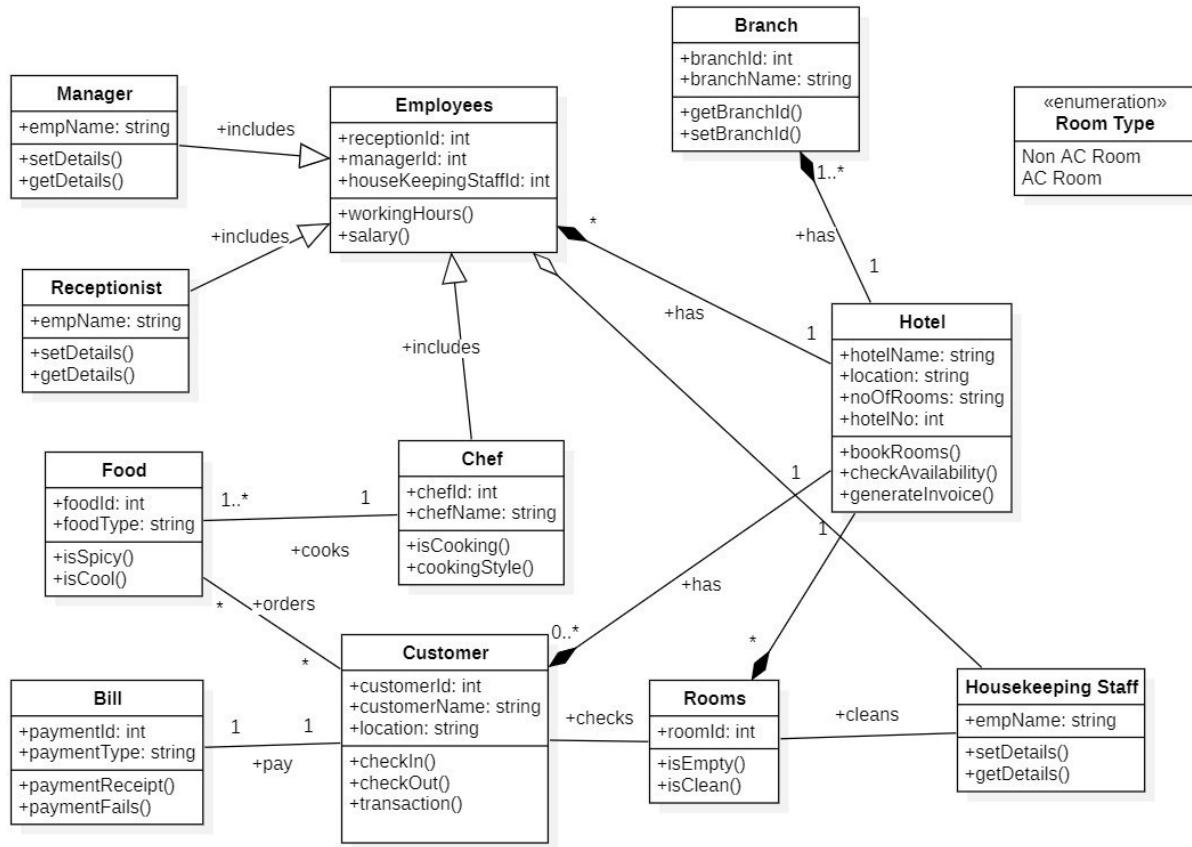


Fig 1.4

The class diagram illustrates the structure and interactions in a **Hotel Management System**, highlighting key roles, operations, and relationships:

- Employees:** A parent class encompassing all employee types like **Manager**, **Receptionist**, **Chef**, and **Housekeeping Staff**, each with specific attributes and methods while sharing common properties like working hours and salary.
- Branch and Hotel:** The **Branch** class represents the organizational hierarchy, with each branch having one or more hotels. The **Hotel** class contains details like location, number of rooms, and functionality to book rooms, check availability, and generate invoices.
- Rooms:** Each room is identified by a unique ID, its status (clean/empty), and belongs to a **Room Type** (AC or Non-AC). Rooms are maintained by the **Housekeeping Staff**.
- Customer:** Customers interact with the system to check in, check out, and handle transactions. They also book rooms and order food.

5. **Food and Chef:** Customers can place food orders, and chefs are responsible for cooking based on customer preferences.
6. **Bill:** Manages payment-related processes, including generating receipts and handling failed payments.

The system ensures smooth coordination among the components, with clearly defined relationships, such as:

- **Branch** contains **Hotels**, and **Hotels** manage **Rooms**.
  - **Customers** book **Rooms**, interact with **Receptionists**, and make payments through the **Bill** system.
  - **Housekeeping Staff** maintain **Rooms**, while **Chefs** prepare ordered **Food**.
-

## State Diagram

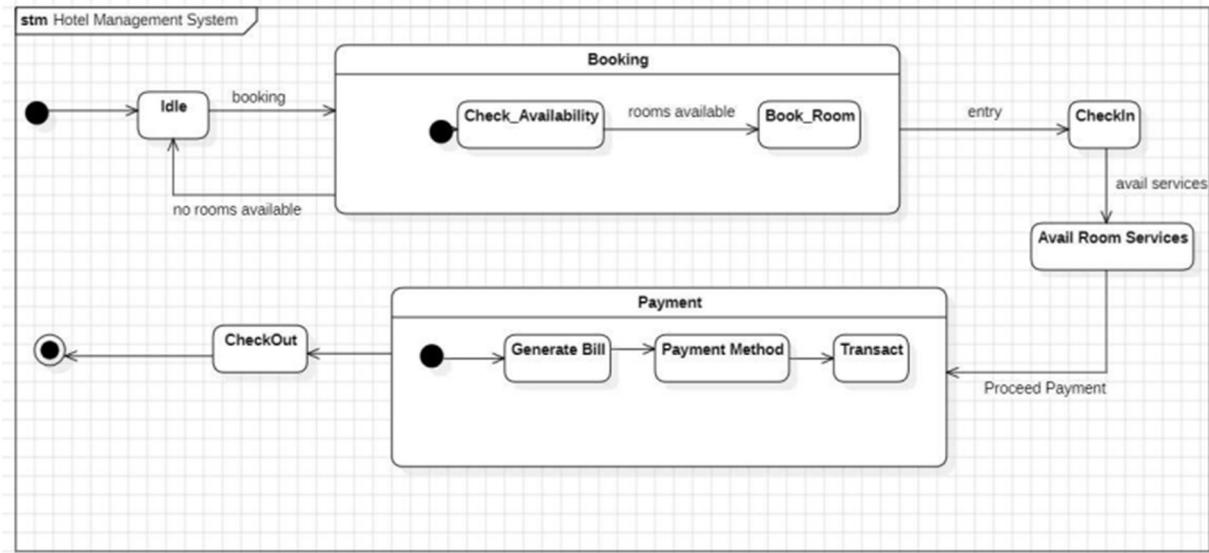


Fig 1.5

The statechart diagram illustrates the workflow for managing hotel bookings and payments within a Hotel Management System. It outlines the states and transitions involved in the booking and payment processes.

### States:

#### Booking Process

- Idle**: The initial state where the system is waiting for a booking request.
- Check Availability**: The state where the system checks for room availability based on the customer's request.
- Book\_Room**: The state where a room is booked for the customer.
- Checkin**: The state where the customer checks in to the hotel.
- Avail Room Services**: The state where the customer avails of additional room services.

#### Payment Process:

- CheckOut**: The state where the customer checks out of the hotel.
- Generate Bill**: The state where the system generates a bill for the customer's stay and services.
- Payment Method**: The state where the customer selects the preferred payment method.
- Transact**: The state where the payment transaction is being processed.
- Processed Payment**: The final state where the payment is successfully processed.

### **Transitions:**

- From **Idle** to **Check Availability**: When a booking request is received.
- From **Check Availability** to **Book\_Room**: If rooms are available.
- From **Check Availability** to **Idle**: If no rooms are available.
- From **Book\_Room** to **Checkin**: When the customer arrives for check-in.
- From **Checkin** to **Avail Room Services**: If the customer requests additional services.
- From **Checkin** to **Payment**: When the customer is ready to check out.
- From **CheckOut** to **Generate Bill**: When the customer checks out.
- From **Generate Bill** to **Payment Method**: After the bill is generated.
- From **Payment Method** to **Transact**: When the customer selects a payment method.
- From **Transact** to **Processed Payment**: When the payment is successfully processed.

The statechart diagram provides a clear and concise overview of the hotel booking and payment process, highlighting the different stages involved and the possible transitions between states.

---

## Use Case Diagram

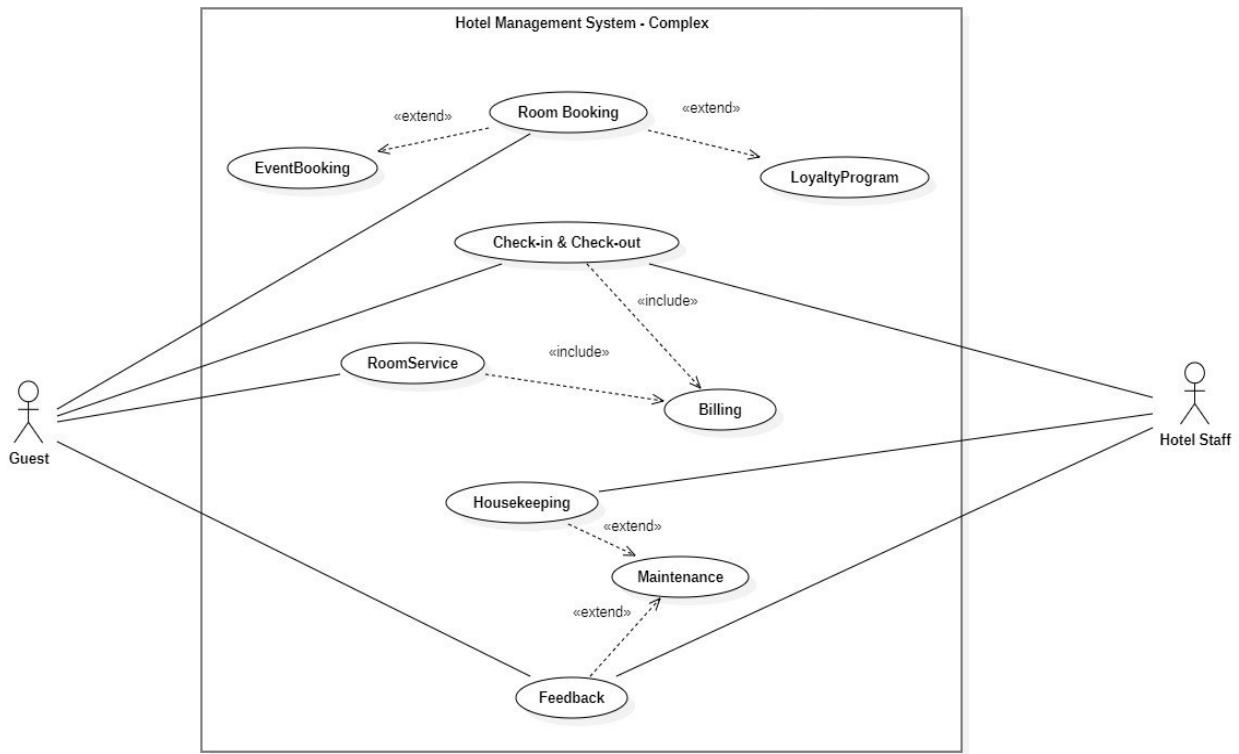


Fig 1.6

The use case diagram for the Hotel Management System represents the interactions between key actors—**Guest** and **Hotel Staff**—and the various services offered by the system. It highlights the system's modularity and functionality for managing hotel operations and guest interactions. Below is a detailed breakdown:

### Actors:

#### 1. Guest:

The primary end-user interacting with the system to access services such as booking rooms, managing reservations, requesting services, and providing feedback.

#### 2. Staff:

Internal users who manage and fulfill the services requested by guests, such as room management, billing, housekeeping, and maintenance.

## **Use Cases:**

### **1. Room Booking:**

- Guests can book rooms online or at the reception.
- Extends to optional services like **Event Booking** (e.g., conferences, parties) and **Loyalty Program** (for frequent customers).

### **2. Check-in & Check-out:**

- Guests check in upon arrival and check out upon departure.
- Includes generating bills and managing payments.

### **3. Billing:**

- Essential for processing payments, generating receipts, and managing failed transactions.
- Directly connected to the check-in and check-out process and integrated with room services and additional guest services.

### **4. Room Service:**

- Guests can request services such as ordering food, laundry, or other in-room amenities.
- Connected to the billing system to ensure accurate charge calculation.

### **5. Housekeeping:**

- Managed by hotel staff to ensure rooms are clean and ready for use.
- Extends to **Maintenance** for addressing any issues like repairs or technical faults.

### **6. Feedback:**

- Guests provide feedback regarding their experience.
- Helps the hotel improve its services, covering aspects like housekeeping, maintenance, and overall guest satisfaction.

## **Relationships:**

- **Extend:**
  - Room booking extends to event booking and loyalty programs for additional functionalities.
  - Housekeeping extends to maintenance when specific repair tasks are required.
- **Include:**
  - Check-in & check-out includes billing to ensure a smooth guest departure process.
  - Room service includes billing for accurate financial management.

## **System Features and Benefits:**

- **Guest-Centric Services:** Enables seamless booking, personalized room service, and effortless payment processes for guests.
  - **Staff Operations:** Simplifies tasks for hotel staff, ensuring efficient coordination between housekeeping, maintenance, and billing.
  - **Feedback Mechanism:** Provides valuable insights for enhancing customer satisfaction.
-

## Sequence Diagram

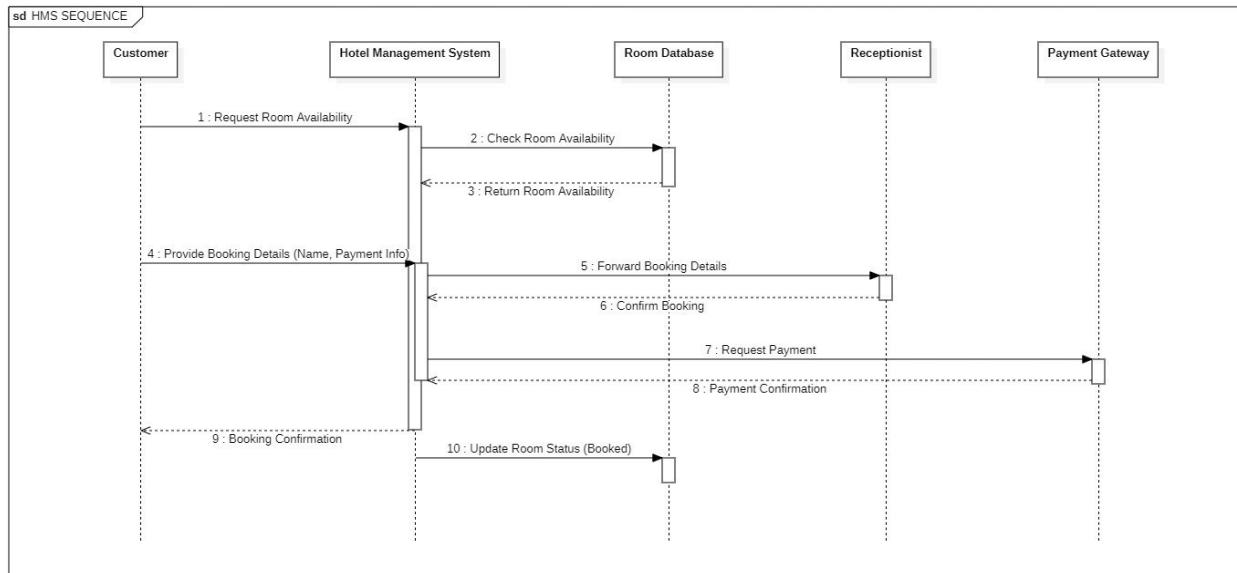


Fig 1.7

The sequence diagram for the Hotel Management System demonstrates the step-by-step process of room booking, showcasing interactions among key entities: Customer, Hotel Management System, Room Database, Receptionist, and Payment Gateway. Below is a detailed description:

### Actors:

#### 1. Customer:

Initiates the room booking process by requesting room availability and providing booking details.

#### 2. Hotel Management System (HMS):

Serves as the central platform managing requests, verifying room availability, and coordinating between other entities.

#### 3. Room Database:

Validates room availability and updates room status after successful booking.

#### 4. Receptionist:

Confirms booking details and oversees the interaction between the system and the customer.

## **5. Payment Gateway:**

Processes payment transactions and confirms payment completion.

### **Key Interactions:**

#### **1. Request Room Availability:**

The customer begins by sending a request to the HMS to check available rooms.

#### **2. Check and Return Room Availability:**

HMS queries the Room Database to validate the availability of rooms and sends the information back to the customer.

#### **3. Provide Booking Details:**

The customer provides essential booking information such as name and payment details to proceed with the booking.

#### **4. Forward Booking Details:**

HMS forwards the received booking details to the Receptionist for confirmation.

#### **5. Confirm Booking:**

The Receptionist validates the booking and notifies the HMS of the confirmation.

#### **6. Request Payment:**

The HMS communicates with the Payment Gateway to initiate the payment process.

#### **7. Payment Confirmation:**

Upon successful payment, the Payment Gateway sends confirmation back to the HMS.

#### **8. Update Room Status:**

HMS updates the Room Database, marking the room as booked.

#### **9. Booking Confirmation:**

The customer receives confirmation of their booking, completing the sequence.

### **Features and Benefits:**

- Centralized System Coordination:**

The HMS ensures seamless communication between the customer, room database, receptionist, and payment gateway.

- **Real-Time Room Availability Updates:**  
The Room Database efficiently tracks and updates room status to prevent double booking.
  - **Secure Payment Processing:**  
Integration with the Payment Gateway ensures safe and reliable transaction handling.
  - **Simplified Customer Experience:**  
Customers experience a streamlined process, from checking room availability to receiving booking confirmation.
-

## Activity Diagram

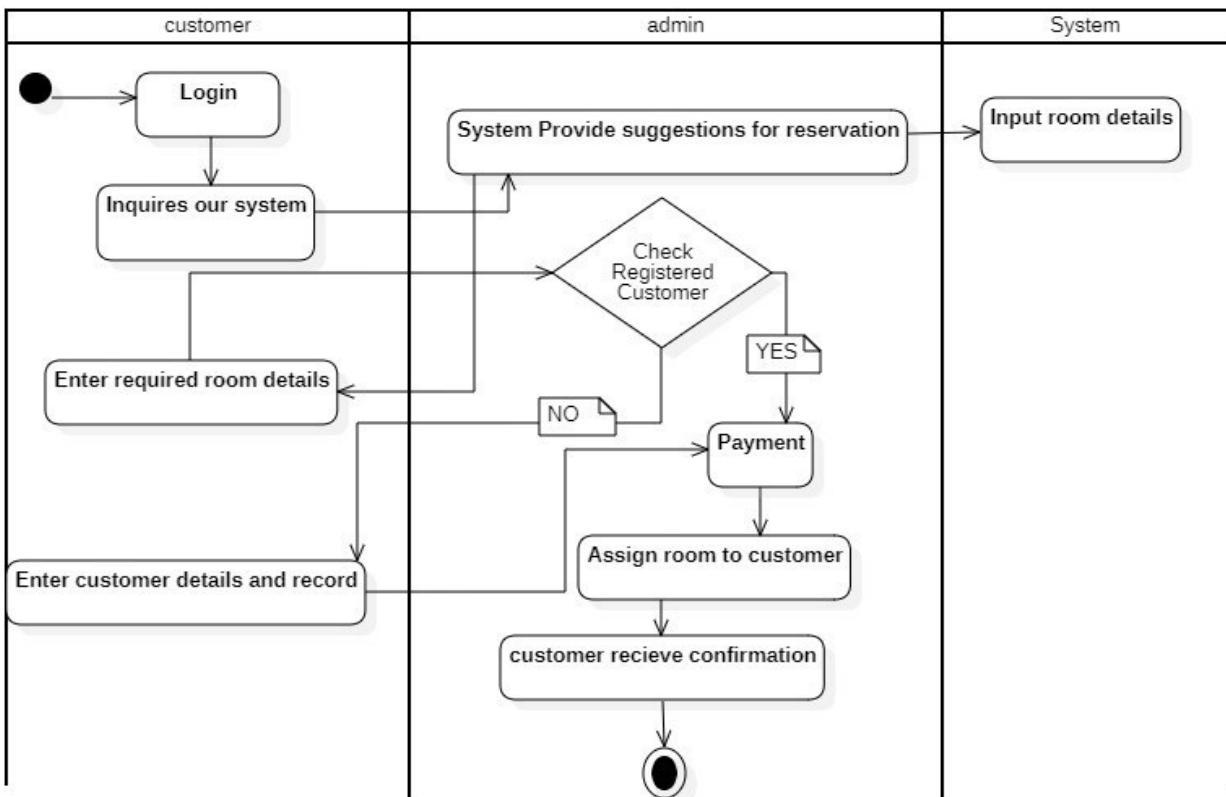


Fig 1.8

The activity diagram illustrates the process flow for room reservation in a hotel management system. It highlights the interactions between the customer, admin, and system components involved in the booking process.

### Actors:

- **Customer:** The end-user who initiates the room reservation request.
- **Admin:** The system administrator responsible for managing room details and customer data.
- **System:** Represents the hotel management system itself, encompassing various functionalities.

### Key Interactions:

1. **Customer Login:** The process starts with the customer logging into the system.

2. **System Provides Suggestions for Reservation:** Based on the customer's preferences or previous inquiries, the system suggests available rooms and their details.
3. **Input Room Details:** The customer selects a desired room and enters the necessary details, such as check-in/check-out dates and room preferences.
4. **Check Registered Customer:** The system verifies if the customer is already registered in the system.
  - o **YES:** If the customer is registered, the system proceeds to the payment step.
  - o **NO:** The customer is prompted to enter their details for registration.
5. **Payment:** The customer makes the payment for the reservation.
6. **Assign Room to Customer:** Upon successful payment, the system assigns the selected room to the customer.
7. **Customer Receives Confirmation:** The system sends a confirmation message to the customer, completing the reservation process.

#### **Features and Benefits:**

- **Centralized System Management:** The diagram showcases how the system coordinates various activities, from room availability checks to payment processing.
- **Efficient Customer Interaction:** The system provides a user-friendly interface for customers to easily inquire, select, and book rooms.
- **Secure Payment Processing:** The integration of payment functionality ensures secure and reliable transactions.
- **Enhanced Customer Experience:** The streamlined process and timely confirmations contribute to a positive customer experience.

#### **Additional Considerations:**

- **Error Handling:** The diagram could be further enhanced by incorporating error handling mechanisms to address potential issues like invalid payment or room unavailability.
- **Cancellation and Modification:** The process could be extended to include functionalities for cancelling or modifying reservations.

## 2. Credit Card Processing System

### Problem Statement:

Design a software system to support a computerized credit card processing network for a large retail organization with multiple stores operating under a single brand. Each store maintains its own point-of-sale (POS) terminal to manage transactions locally. POS terminals at individual stores communicate directly with their local computer systems to handle sales, refunds, and void transactions. Sales associates input customer and transaction details.

An online payment gateway communicates with a central server that coordinates transaction processing and authorization across all stores, ensuring real-time updates for transaction status and inventory levels. The online system allows customers to make purchases online, process payments securely, and access their transaction history.

The system requires robust recordkeeping, scalability to support multiple stores, and strict security measures to protect customer data and transactions. It must handle concurrent transaction requests and avoid fraudulent activities. Each store will provide and maintain its software for local POS operations, while you are tasked with designing the software for the central authorization server, the online payment gateway, and the network connecting individual stores to the central server. The cost of the shared system will be apportioned to stores based on the volume of transactions they process.

---

# Software Requirement Specification Document

LAB - 2	7/19/24	classmate
<p><b>Hotel Management System:</b></p> <p>1) Introduction:</p> <p>1.1) Purpose: To establish the functional and non-functional requirements for a Hotel Management System.</p> <p>1.2) Scope: The Hotel management system will automate the hotel booking process, managing customer details, and billing related to the clients.</p> <p>1.3) Overview: The Hotel management system will automate the entire booking system and management. The system will be quick and responsive for optimal user interaction.</p> <p>2) General Description:</p> <p>The system for Hotel management is a standalone model which allows the user to login and verify themselves as the first step after which the main functionalities are given to the user. The system features are as follows:</p>	<p>→ Room Booking System: Allows customers to book rooms online or via staff entry.</p> <p>→ Check-in/Check-out System: Track guest arrivals and departures.</p> <p>→ Staff Management: Manage employee data, roles and schedules.</p> <p>→ Billing System: Automate invoice generation and payment processing.</p> <p>3) Functional Requirements:</p> <p>1) User Registration: The system enables a new user to register themselves and create a profile and on the other hand also enables an existing user to login to the system.</p> <p>2) Check-in/Check-out System: A functionality which allows a user to check-in/check-out of from their respective rooms. This will also enable the staff with the same functionality to clean and maintain rooms.</p>	

Fig 2.1

<p>3) Billing and reporting: The overall bill payment and management of different bills must reflect requirements of a different functionality.</p> <p>4) Interface Requirements:</p> <ol style="list-style-type: none"> <li>1) User-System Interface: The user interface shall provide an intuitive navigation for booking, checking-in/out, and managing rooms.</li> <li>2) Hardware Interface: The system must consist of barcode scanners for check-in/out.</li> </ol> <p>5) Performance Requirements:</p> <ol style="list-style-type: none"> <li>1) Speed: The system must be quick to respond to user inputs.</li> <li>2) Memory: The system requires 1TB of memory in order to function efficiently.</li> </ol>	<p>6) Design Constraints: The software must consist of a frontend developed by using HTML, CSS, JavaScript. The backend must involve use of technologies like Node.js or Angular.</p> <p>7) Non-Functional Requirements:</p> <ul style="list-style-type: none"> <li>→ Portability: The system must be accessible through any device thus making it portable.</li> <li>→ Reliability: The system must be reliable in the sense that it must not crash due to huge traffic of users.</li> </ul> <p>8) Preliminary Schedule and Budget:</p> <table border="1"> <thead> <tr> <th>Phase</th> <th>Duration (weeks)</th> </tr> </thead> <tbody> <tr> <td>Project Initiation</td> <td>2</td> </tr> <tr> <td>Requirements Gathering</td> <td>4</td> </tr> <tr> <td>System Design</td> <td>3</td> </tr> <tr> <td>Development</td> <td>10</td> </tr> <tr> <td>Testing</td> <td>3</td> </tr> <tr> <td>Deployment</td> <td>2</td> </tr> </tbody> </table>	Phase	Duration (weeks)	Project Initiation	2	Requirements Gathering	4	System Design	3	Development	10	Testing	3	Deployment	2
Phase	Duration (weeks)														
Project Initiation	2														
Requirements Gathering	4														
System Design	3														
Development	10														
Testing	3														
Deployment	2														

Fig 2.2

classmate  
Date \_\_\_\_\_  
Page \_\_\_\_\_

Budget Breakdown:		Estimated cost (L)
Personnel	cost	200,000
Software	for census	10,000
Infrastructure	rent	15,000
Security	and	10,000
Training		7500
Marketing		2500

~~Q~~ Planning Phase: Infrastructure

1. Plan areas of responsibility  
and enough staff available  
to do the job

2. Set up review cell to monitor  
work and make changes  
as needed

3. Set up with client for review of  
work done

4. Establish time schedule for each task

5. Assign tasks to staff

6. Assign tasks to staff

7. Assign tasks to staff

8. Assign tasks to staff

9. Assign tasks to staff

10. Assign tasks to staff

11. Assign tasks to staff

12. Assign tasks to staff

13. Assign tasks to staff

14. Assign tasks to staff

15. Assign tasks to staff

16. Assign tasks to staff

17. Assign tasks to staff

18. Assign tasks to staff

19. Assign tasks to staff

20. Assign tasks to staff

Fig 2.3

## Class Diagram

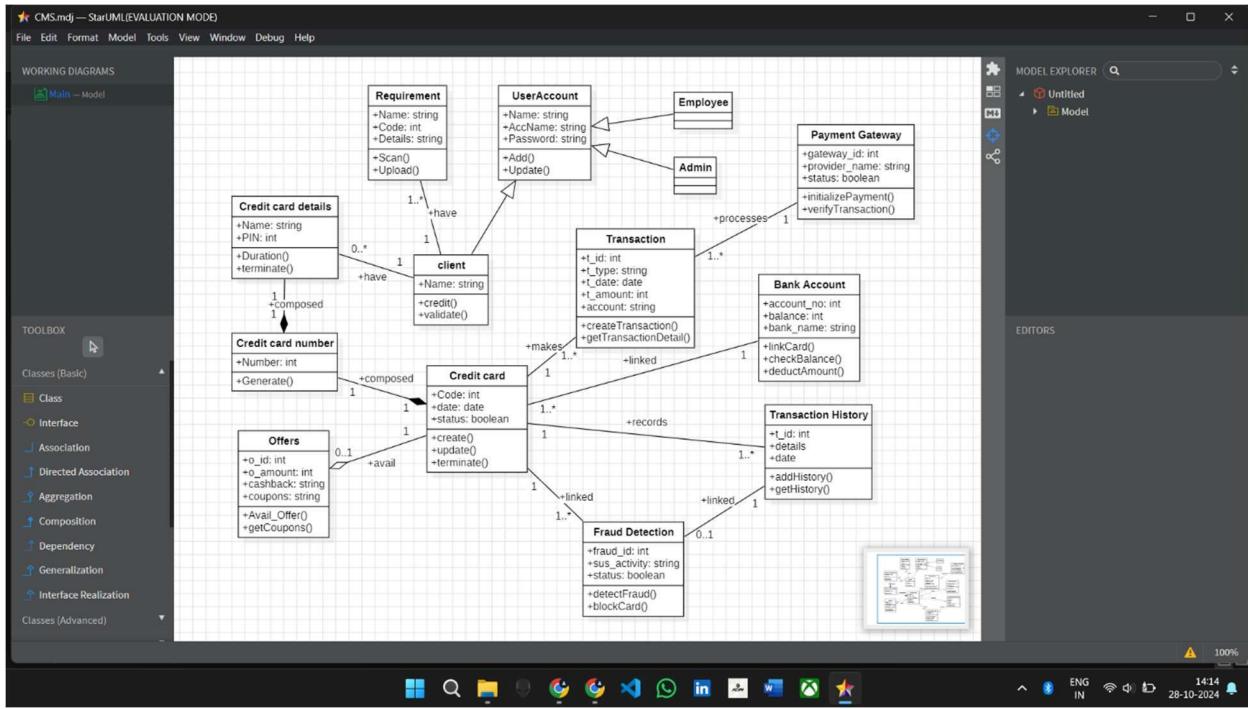


Fig 2.4

The class diagram illustrates the structure and interactions within a credit card processing system, highlighting key roles, operations, and relationships.

## Key Classes:

- **UserAccount:** Represents user accounts with attributes like Name, AccountName, and Password.
  - **Employee:** Represents employees with attributes like Name, AccountName, and Password.
  - **Admin:** A specialized type of Employee with administrative privileges.
  - **Credit Card:** Represents credit card details with attributes like Name, PIN, and Number.
  - **Bank Account:** Represents bank accounts with attributes like Account Number and Balance.
  - **Transaction:** Represents a single financial transaction, with attributes like TID, Type (e.g., purchase, refund), and Amount.
  - **Transaction History:** Records a collection of transactions associated with a specific account.
  - **Payment Gateway:** Handles the processing of credit card payments, including authorization and settlement.

- **Fraud Detection:** Implements algorithms and mechanisms to identify and prevent fraudulent transactions.

### Relationships:

- **Inheritance:** Employee inherits from UserAccount, and Admin inherits from Employee, indicating that Admins are a specific type of Employee.
- **Composition:** Credit Card is composed of Credit Card Number, meaning the Credit Card Number cannot exist independently.
- **Aggregation:** Transaction History is aggregated by Transaction, meaning a Transaction History can contain multiple Transactions.
- **Associations:** Various associations exist between classes like "client," "processes," "makes," "linked," and "records." These indicate relationships between different entities, such as a UserAccount making a Transaction or a Payment Gateway processing a Transaction.

The structure ensures smooth coordination between different components of the credit card processing system, with clearly defined relationships and responsibilities for each class.

---

## State Diagram

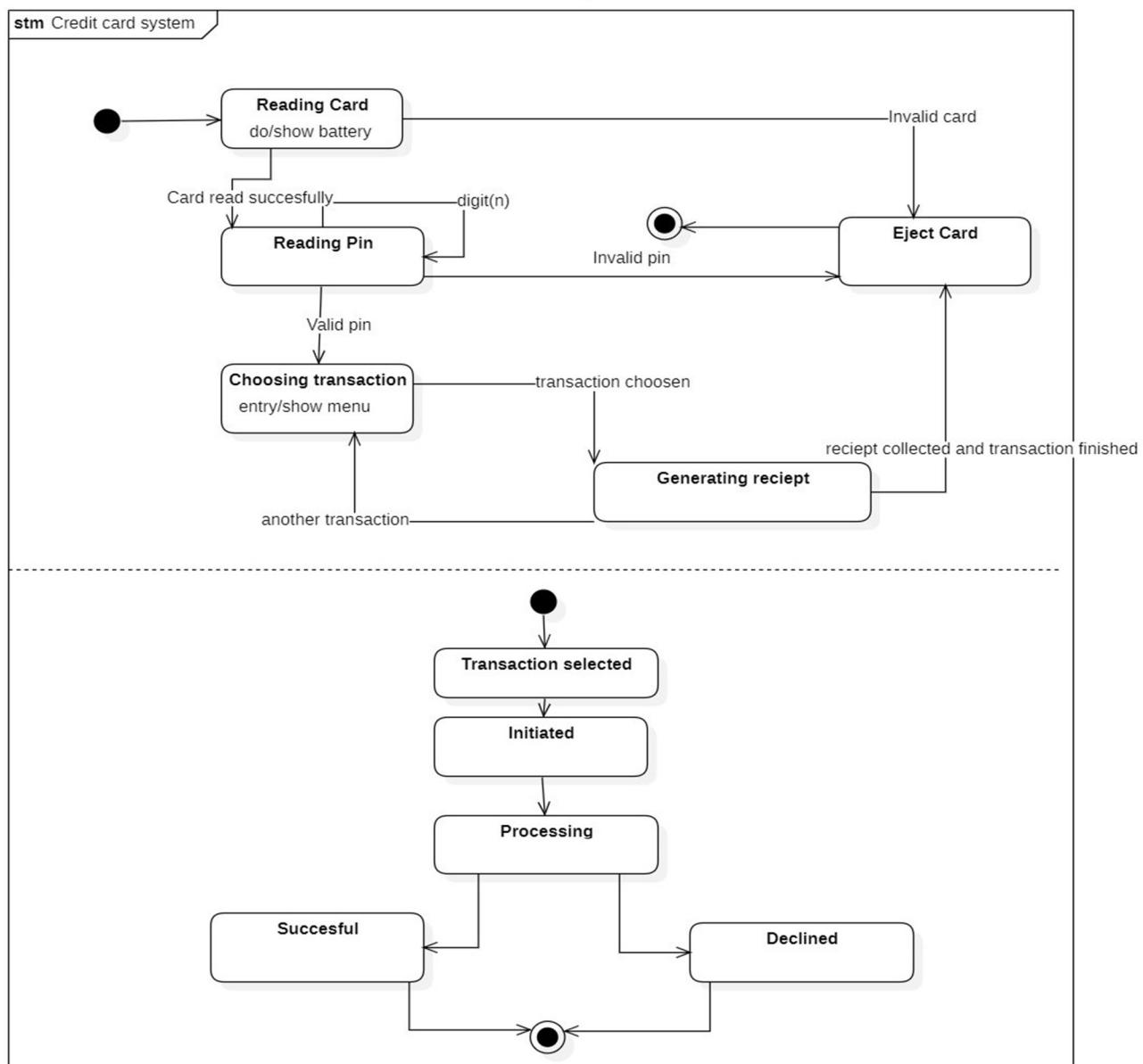


Fig 2.5

The statechart diagram illustrates the workflow of a credit card system. It outlines the various states and transitions involved in a typical credit card transaction.

### States:

- **Reading Card:** The initial state where the system reads the credit card information.
- **Reading Pin:** The state where the system prompts the user to enter their PIN.
- **Choosing Transaction:** The state where the user selects the type of transaction they wish to perform (e.g., purchase, cash withdrawal).

- **Generating Receipt:** The state where the system generates a receipt for the completed transaction.
- **Transaction Selected:** The state where the user has chosen the type of transaction.
- **Initiated:** The state where the transaction has been initiated and is being processed.
- **Processing:** The state where the transaction is being processed by the system.
- **Successful:** The state where the transaction has been successfully completed.
- **Declined:** The state where the transaction has been declined.

#### **Transitions:**

- **Reading Card:**
  - If the card is read successfully, the system transitions to the "Reading Pin" state.
  - If the card is invalid, the system transitions to the "Eject Card" state.
- **Reading Pin:**
  - If the entered PIN is valid, the system transitions to the "Choosing Transaction" state.
  - If the entered PIN is invalid, the system transitions to the "Eject Card" state.
- **Choosing Transaction:**
  - Once the user selects a transaction, the system transitions to the "Transaction Selected" state.
- **Transaction Selected:**
  - The system transitions to the "Initiated" state to begin processing the transaction.
- **Initiated:**
  - The system transitions to the "Processing" state to carry out the transaction.
- **Processing:**
  - If the transaction is successful, the system transitions to the "Successful" state.
  - If the transaction is declined, the system transitions to the "Declined" state.
- **Successful:**
  - The system transitions to the "Generating Receipt" state to generate a receipt for the successful transaction.
- **Generating Receipt:**
  - Once the receipt is generated, the system can either transition to the "Reading Card" state for another transaction or to the end state.

## Use Case Diagram

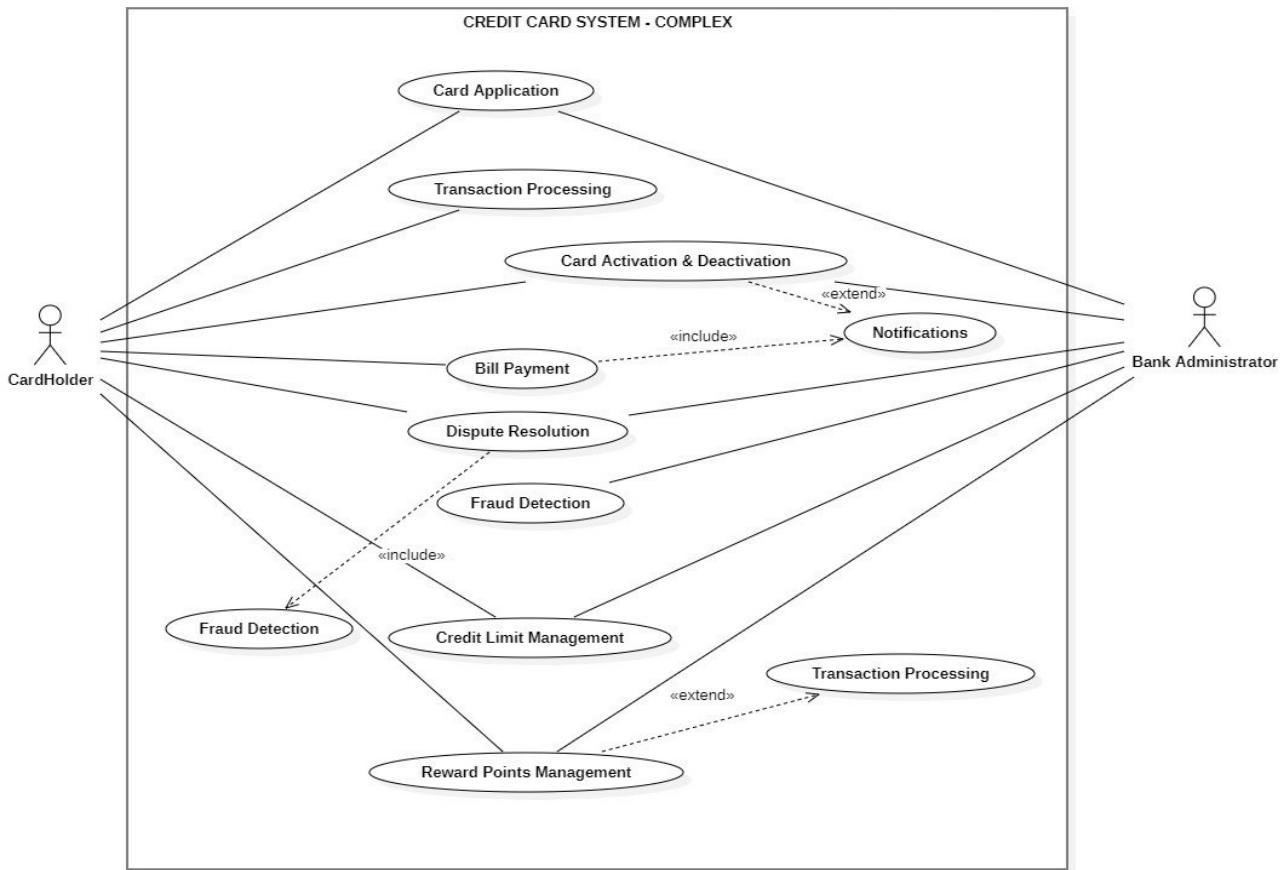


Fig 2.6

The use case diagram illustrates the various functionalities and interactions within a complex credit card system. It highlights the different use cases available to both Cardholders and Bank Administrators, as well as the relationships between these use cases.

### Actors:

- **CardHolder:** Represents the end-user of the credit card, who interacts with the system for various purposes.
- **Bank Administrator:** Represents the system administrator responsible for managing the credit card system and its functionalities.

### Use Cases:

- **Card Application:** The process of applying for a new credit card.
- **Transaction Processing:** Handles the processing of credit card transactions, including purchases, payments, and refunds.

- **Card Activation & Deactivation:** Allows for the activation and deactivation of credit cards.
- **Bill Payment:** Enables cardholders to make bill payments.
- **Dispute Resolution:** Handles disputes related to transactions or charges.
- **Fraud Detection:** Implements mechanisms to detect and prevent fraudulent activities.
- **Credit Limit Management:** Manages credit limits for individual cardholders.
- **Reward Points Management:** Manages the accumulation and redemption of reward points associated with credit card usage.
- **Notifications:** Sends notifications to cardholders and administrators regarding transactions, alerts, and other important information.

### **Relationships:**

- **Include:** The "Fraud Detection" use case is included in both "Transaction Processing" and "Dispute Resolution," indicating that fraud detection is an integral part of both processes.
- **Extend:** The "Notifications" use case extends "Card Activation & Deactivation," indicating that notifications can be sent as part of the card activation and deactivation process.

The use case diagram provides a comprehensive overview of the credit card system's functionalities, highlighting the interactions between different actors and use cases, as well as the relationships between various components of the system.

---

## Sequence Diagram

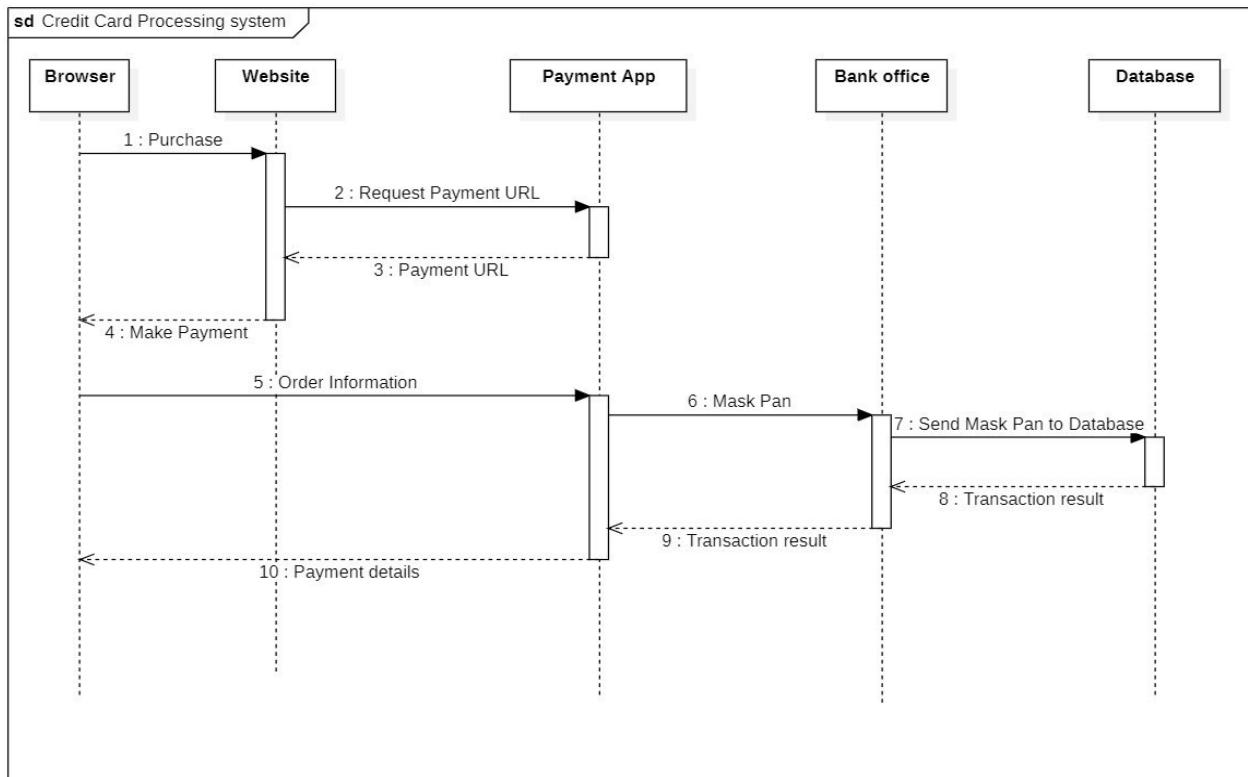


Fig 2.7

This sequence diagram illustrates the flow of interactions involved in a credit card processing system during an online purchase. It outlines the sequence of messages exchanged between different components of the system, including the Browser, Website, Payment App, Bank Office, and Database.

### Actors:

- **Browser:** Represents the user's web browser.
- **Website:** The online store where the user is making a purchase.
- **Payment App:** A third-party application used to process the payment.
- **Bank Office:** The bank's system that handles the actual transaction processing.
- **Database:** The central repository for storing transaction data.

### Sequence of Events:

1. **Purchase:** The user initiates a purchase on the website by selecting items and proceeding to checkout.
2. **Request Payment URL:** The website requests a payment URL from the Payment App.
3. **Payment URL:** The Payment App sends the payment URL back to the website.

4. **Make Payment:** The user is redirected to the Payment App using the provided URL.
5. **Order Information:** The Payment App receives order information from the website.
6. **Mask Pan:** The Payment App masks the credit card number (Pan) before sending it to the Bank Office for security purposes.
7. **Send Mask Pan to Database:** The Bank Office sends the masked Pan to the Database for processing.
8. **Transaction Result:** The Database processes the transaction and sends the result (success or failure) back to the Bank Office.
9. **Transaction Result:** The Bank Office sends the transaction result to the Payment App.
10. **Payment Details:** The Payment App sends the payment details (including the result) back to the website.

The sequence diagram provides a clear and concise overview of the steps involved in a typical online credit card transaction, highlighting the interactions between different components of the system and the flow of information during the process.

---

## Activity Diagram

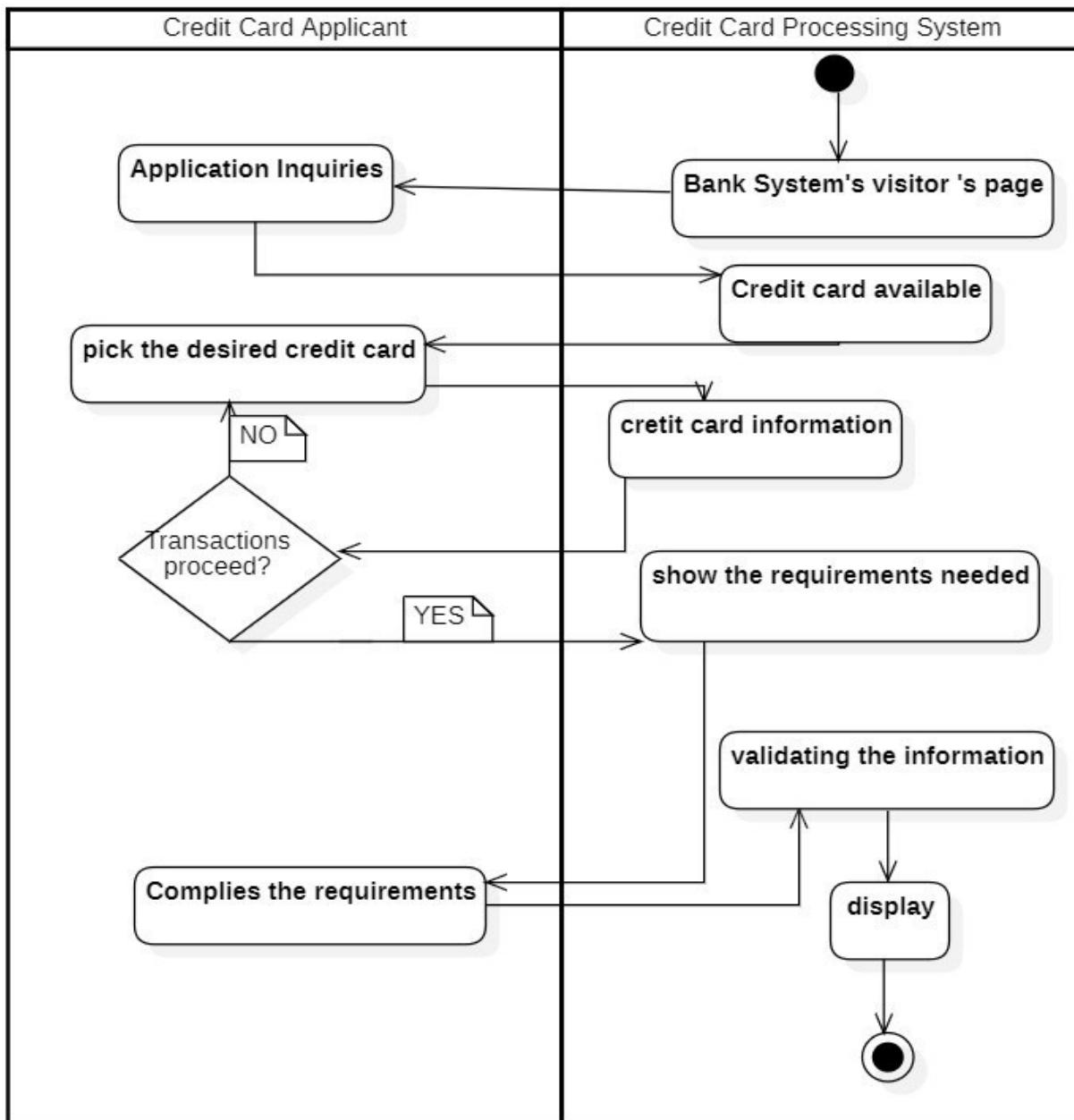


Fig 2.8

The activity diagram illustrates the process flow for a credit card application within a credit card processing system. It outlines the steps involved from the applicant's perspective and the system's response.

**Actors:**

- **Credit Card Applicant:** The individual who is applying for a credit card.
- **Credit Card Processing System:** The system that handles the application process.

### **Activities:**

- **Application Inquiries:** The applicant initiates the process by making inquiries about available credit cards.
- **Bank System's Visitor's Page:** The applicant is directed to the bank's system's visitor's page to view available credit card options.
- **Credit Card Available:** The applicant selects a desired credit card.
- **Credit Card Information:** The system provides the applicant with information about the selected credit card, including requirements and terms.
- **Transactions Proceed?** The system checks if the applicant meets the requirements for the selected credit card.
  - **NO:** The system informs the applicant that they do not meet the requirements and may suggest alternative options.
  - **YES:** The applicant proceeds with the application process.
- **Show the Requirements Needed:** The system displays the specific requirements that the applicant needs to fulfill.
- **Complies the Requirements:** The applicant provides the necessary information and documents to comply with the requirements.
- **Validating the Information:** The system validates the information provided by the applicant.
- **Display:** The system displays the outcome of the application process, indicating whether the application is approved or denied.

The activity diagram provides a clear and concise overview of the credit card application process, highlighting the key steps involved and the interactions between the applicant and the credit card processing system.

---

### **3. Library Management System**

#### **Problem Statement:**

Design a software system to support a computerized library management network for a large library system with multiple branches operating under a single administration. Each branch maintains its own local computer system to manage book lending, returns, and member records. Library staff at individual branches interact directly with their local computer systems to handle checkouts, check-ins, and member inquiries. Library staff input book and member details.

A central library management system communicates with a central server that coordinates book availability across all branches, ensures real-time updates for book locations, and manages member accounts centrally. The central system allows library members to search for books, place holds, renew loans, and access their borrowing history online.

The system requires robust recordkeeping, scalability to support multiple branches, and strict security measures to protect member data and library assets. It must handle concurrent requests for books and prevent overdue fines. Each branch will provide and maintain its software for local library operations, while you are tasked with designing the software for the central library management system, the online member portal, and the network connecting individual branches to the central server. The cost of the shared system will be apportioned to branches based on the volume of book transactions they process.

---

# Software Requirement Specification Document

classmate

Date \_\_\_\_\_

Page \_\_\_\_\_

## LIBRARY MANAGEMENT SYSTEM:

### 1) Introduction:

1.1) Purpose: To establish the functional and non-functional requirements for a library management system.

1.2) Scope: To create an automated system to manage the smooth functioning of all library operations, including cataloging, book lending, and member management.

### 1.3) Overview:

An independent system that manages all functions related to running a library. The system will allow users to search for books, borrow and return them, and maintain records of borrowing activities.

2) General description: The library system automates key functions like cataloging, borrowing, returning books and managing member records. It serves both staff and members, making

Fig 3.1

<p>library operations more efficient. Key features include an intuitive book catalog management system that allows librarians to add, update and categorize books efficiently. Users can search for books by title, author, or genre and reserve them if unavailable.</p> <p><b>3) Functional Requirements:</b></p> <ul style="list-style-type: none"> <li><b>• Book Catalog Management:</b> <ul style="list-style-type: none"> <li>Ability to add, update, or remove books from the system.</li> <li>Categorization of books by genre, author, publication date etc.</li> </ul> </li> <li><b>• User Management:</b> <ul style="list-style-type: none"> <li>Users can register and manage their profiles, while librarians handle user records, borrowing history and overdue fines.</li> </ul> </li> <li><b>• Reservation System:</b> <ul style="list-style-type: none"> <li>Users can reserve unavailable books, and the system notifies them when the books are ready for pick-up.</li> </ul> </li> </ul>	<p><b>• Search functionality:</b> Users can reserve unavailable books by title, author, genre or ISBN to quickly locate what they need.</p> <p><b>4) Interface Requirements:</b></p> <ul style="list-style-type: none"> <li><b>• User Interface:</b> The Library Management System will feature a web-based GUI built with modern front-end tech, for ex. HTML5, CSS3, JavaScript, that enables users to perform operations such as browsing, searching, and borrowing books.</li> <li><b>• Software Interface:</b> The System will integrate with a relational database management system (RDBMS), utilizing SQL for efficient storage and retrieval of book and user data.</li> <li><b>5) Performance Requirements:</b></li> <li><b>• Response Time:</b> The system must ensure optimal performance with a response time under 3 seconds for user queries.</li> </ul>
--	---

Fig 3.2

	classmate Date _____ Page _____
1) Scalability:	The Library Management System should scale effectively to support up to 10,000 books and 10,000 users, ensuring no significant performance degradation as the database grows.
2) Error Rate:	The system must maintain high reliability, with an error rate of less than 0.1% for critical operations, such as book inventory management, ensuring data integrity and accurate transactions.
3) Non-Functional Requirements:	<ul style="list-style-type: none"> <li>a) Security: The system should ensure that user data, including passwords, is stored in an encrypted format. Only authorized users (librarians) can modify book or user records.</li> <li>b) Portability: The system should be compatible across different operating systems, including Windows, macOS, and Linux.</li> <li>c) Reliability: The system should have an uptime of 99.9%.</li> </ul>
4) Design Requirements:	<ul style="list-style-type: none"> <li>a) Database Requirements: The system will utilize a relational database (MySQL or PostgreSQL) for robust storage and management.</li> <li>b) Platform Requirements: The Library System will be developed as a web-application, ensuring compatibility with all major browsers (including Chrome, Safari, etc.).</li> </ul>
	classmate Date _____ Page _____

Fig 3.3

•) System Design : Structuring the system

→ Architecture and detailed design for implementation.

•) Development : Actual coding & building of the system based on design specifications.

Schedule :	Duration (weeks)
Project Inspection	2
Requirements Gathering	4
System Design	3
Development	10
Testing	3
Deployment	2

Budget :

Category	Cost (\$)
Personnel cost	200,000
Software licenses	10,000
Infrastructure	15,000
Security	10,000
Training	7,500
Marketing	2500

Fig 3.3

## Class Diagram

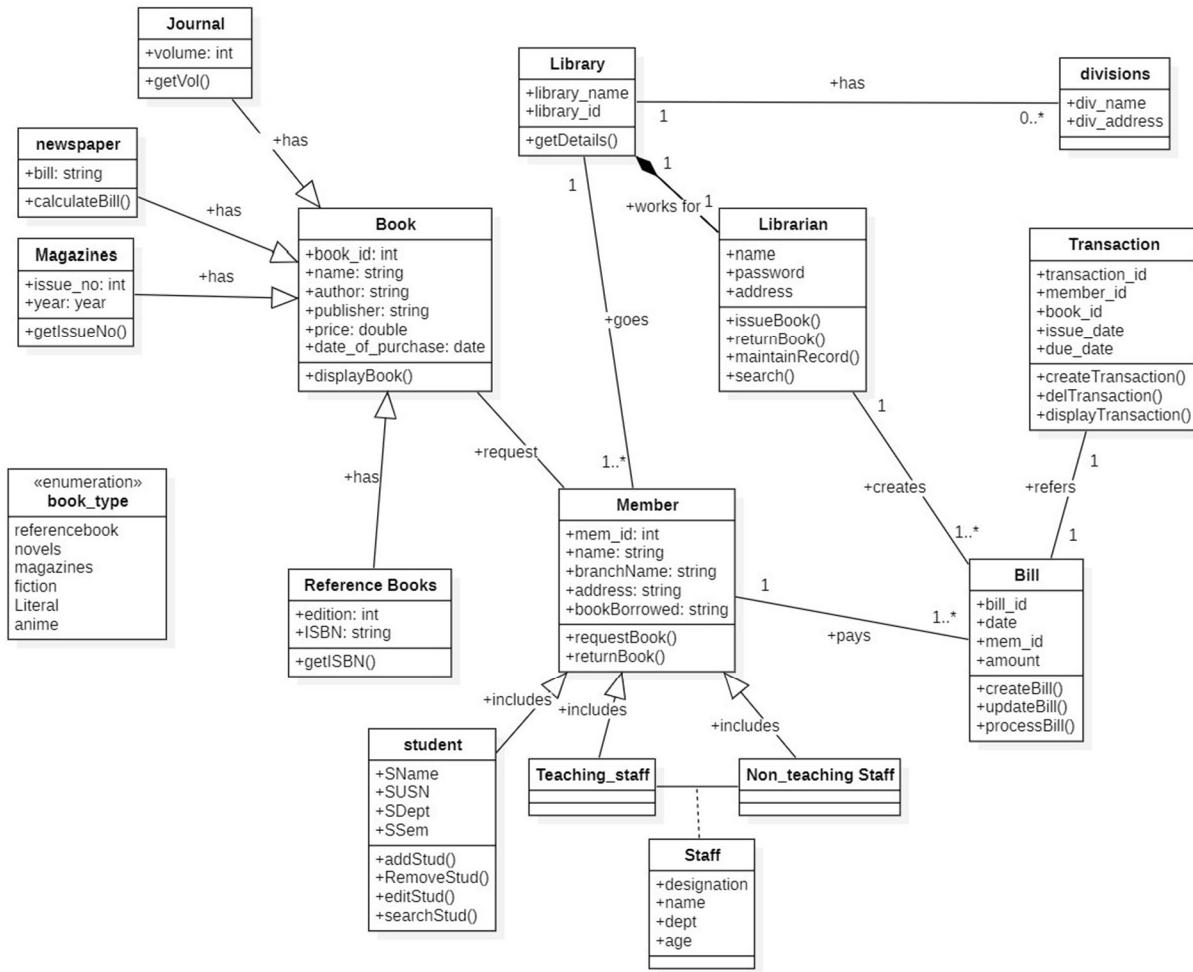


Fig 3.4

The class diagram illustrates the structure and relationships within a Library Management System. It highlights the key entities, their attributes, and the interactions between them.

### Key Classes

- **Library:** Represents a library with attributes like `library_name`, `library_id`, and has a relationship with **Divisions**.
- **Division:** Represents a division within a library, with attributes like `div_name` and `div_address`.
- **Book:** Represents a book with attributes like `book_id`, name, author, publisher, price, and `date_of_purchase`. It has a relationship with **BookType**.
- **BookType:** An enumeration class representing different types of books (e.g., reference, novels, magazines).

- **Member:** Represents library members with attributes like mem\_id, name, address, and branchName.
- **Librarian:** Represents library staff with attributes like name, password, and address.
- **Transaction:** Represents a transaction related to book borrowing or returning, with attributes like transaction\_id, book\_id, member\_id, issue\_date, and due\_date.
- **Bill:** Represents a bill for library services, with attributes like bill\_id, mem\_id, and amount.
- **Staff:** A parent class for different types of library staff (Teaching\_staff, Non\_teaching\_staff).
- **Student:** Represents student members with attributes like SName, SUSN, SDept, and SSem.

## **Relationships**

- **Library has Divisions:** A library can have multiple divisions.
- **Library has Books:** A library maintains a collection of books.
- **Book has BookType:** Each book belongs to a specific type.
- **Member requests Book:** Members can request to borrow books.
- **Member pays Bill:** Members pay for library services.
- **Librarian creates Transaction:** Librarians manage the book borrowing and returning process.
- **Transaction refers Bill:** Transactions can be associated with a bill.
- **Staff includes Teaching\_staff and Non\_teaching\_staff:** Teaching\_staff and Non\_teaching\_staff are types of Staff.
- **Student includes SName, SUSN, SDept, and SSem:** Students have additional attributes like student name, student ID, department, and semester.

The class diagram provides a clear and concise representation of the entities and their relationships within the library management system, laying the foundation for the design and implementation of the software.

---

## State Diagram

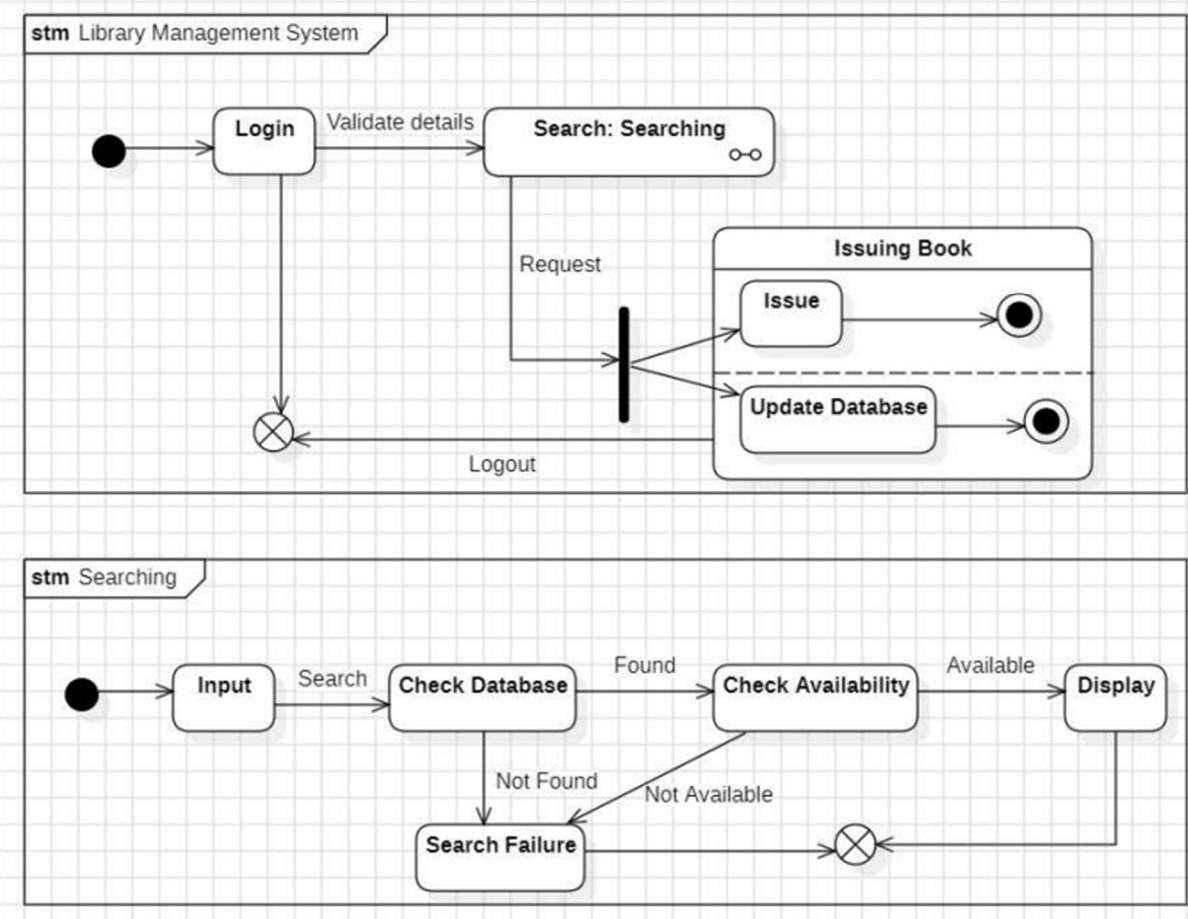


Fig 3.5

The statechart diagram illustrates the workflow within a Library Management System, focusing on the book searching and issuing processes. It outlines the states and transitions involved in these operations.

### Top Level (Library Management System)

- States:
  - **Login:** The initial state where the user logs into the system.
  - **Search: Searching:** The state where the system is actively searching for books based on the user's query.
  - **Issuing Book:** The state where the book is being issued to the user.
- Transitions:
  - From **Login** to **Search: Searching:**: After successful login, the system transitions to the searching state.

- From **Search: Searching** to **Issuing Book**: If a book is found and available, the system transitions to the issuing state.
- From **Issuing Book** to **Issue**: The book is issued to the user.
- From **Issuing Book** to **Update Database**: The system updates the database to reflect the book issue.
- From any state to **Logout**: The user logs out of the system.

### **Bottom Level (Searching)**

- **States:**
  - **Input**: The initial state where the user enters the search criteria.
  - **Search**: The state where the system is searching the database based on the input criteria.
  - **Check Database**: The state where the system checks the database for matching books.
  - **Found**: The state where a book matching the criteria is found.
  - **Check Availability**: The state where the system checks if the found book is available for issue.
  - **Available**: The state where the found book is available for issue.
  - **Display**: The state where the system displays the book details to the user.
  - **Not Found**: The state where no books matching the criteria are found.
  - **Not Available**: The state where the found book is not available for issue.
  - **Search Failure**: The final state when the search fails to find any available books.
- **Transitions:**
  - From **Input** to **Search**: When the user enters the search criteria.
  - From **Search** to **Check Database**: The system starts checking the database.
  - From **Check Database** to **Found**: If a book is found.
  - From **Check Database** to **Not Found**: If no book is found.
  - From **Found** to **Check Availability**: The system checks the availability of the found book.
  - From **Check Availability** to **Available**: If the book is available.
  - From **Check Availability** to **Not Available**: If the book is not available.
  - From **Available** to **Display**: The system displays the book details to the user.

- From **Not Found** or **Not Available** to **Search Failure**: The search process ends with a failure.

The statechart diagram provides a clear and concise overview of the book searching and issuing processes within the Library Management System, highlighting the different states and transitions involved in these operations.

---

## Use Case Diagram

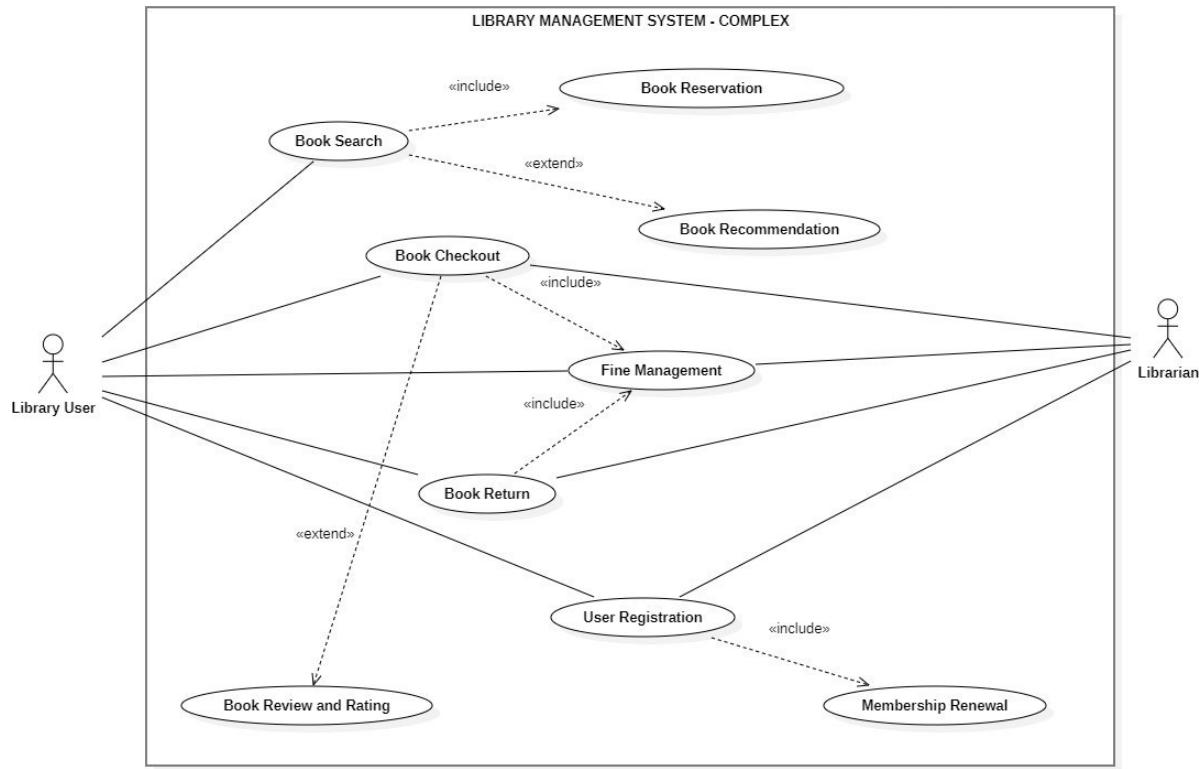


Fig 3.6

The use case diagram illustrates the various functionalities and interactions within a complex Library Management System. It highlights the different use cases available to both Library Users and Librarians, as well as the relationships between these use cases.

### Actors:

- **Library User:** Represents the end-user of the library system, who interacts with the system for various purposes.
- **Librarian:** Represents the library staff responsible for managing the library system and its functionalities.

### Use Cases:

- **Book Search:** Enables users to search for books in the library's catalog.
- **Book Reservation:** Allows users to reserve books that are currently checked out.
- **Book Checkout:** Handles the process of checking out books to library users.
- **Book Return:** Handles the process of returning borrowed books.
- **Fine Management:** Manages the imposition and collection of fines for overdue books.
- **User Registration:** Enables new users to register for library membership.

- **Membership Renewal:** Allows existing members to renew their library memberships.
- **Book Review and Rating:** Enables users to write reviews and rate books.
- **Book Recommendation:** Provides personalized book recommendations to users.

#### **Relationships:**

- **Include:** The "Book Reservation" use case includes the "Book Search" use case, indicating that book reservation involves searching for available books.
- **Include:** The "Book Checkout" use case includes the "Fine Management" use case, indicating that fines may be assessed during the checkout process.
- **Include:** The "Book Return" use case includes the "Fine Management" use case, indicating that fines may be assessed during the return process.
- **Include:** The "User Registration" use case includes the "Membership Renewal" use case, indicating that membership renewal is part of the user registration process.
- **Extend:** The "Book Recommendation" use case extends the "Book Search" use case, indicating that book recommendations can be provided as an additional feature of the book search functionality.

The use case diagram provides a comprehensive overview of the library management system's functionalities, highlighting the interactions between different actors and use cases, as well as the relationships between various components of the system.

---

## Sequence Diagram

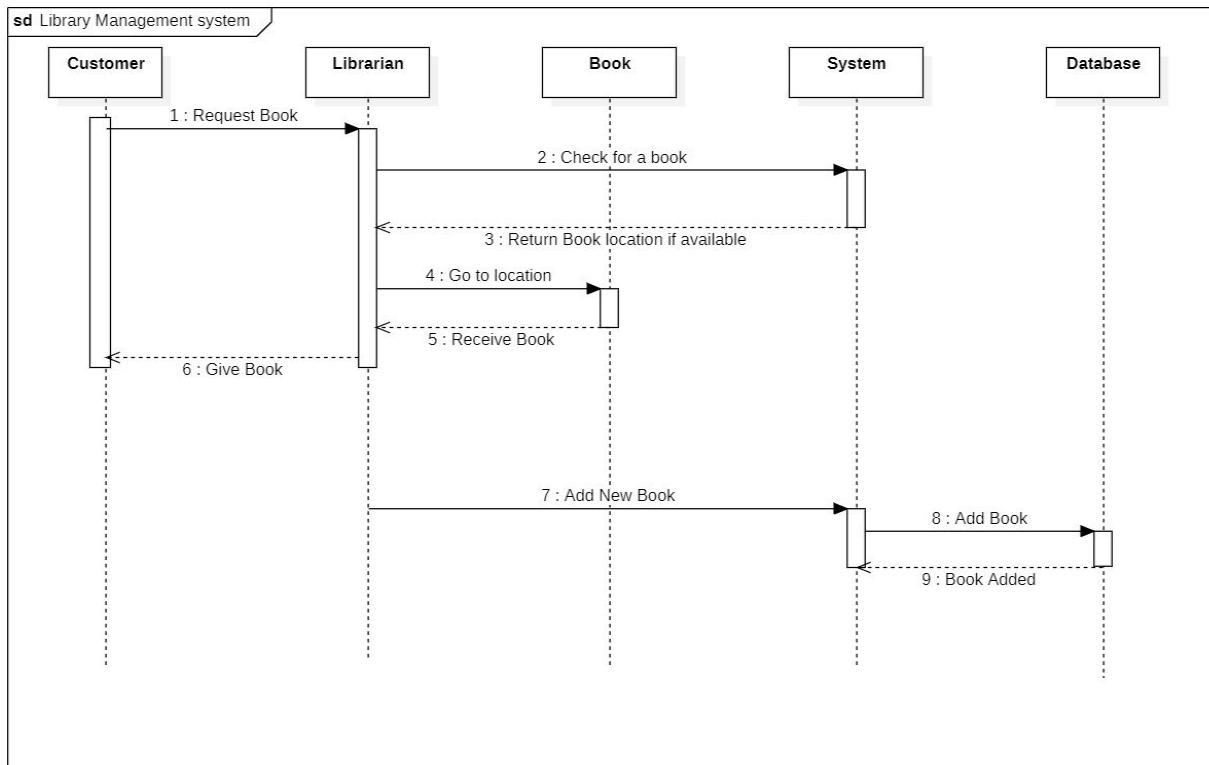


Fig 3.7

The sequence diagram illustrates the flow of interactions involved in a library management system, specifically focusing on the process of requesting and receiving a book. It outlines the sequence of messages exchanged between the Customer, Librarian, Book system, System, and Database.

### Actors:

- **Customer:** Represents the library user who requests a book.
- **Librarian:** Represents the library staff who assists the customer.
- **Book System:** Represents the system used to manage book information and availability.
- **System:** Represents the overall library management system.
- **Database:** Represents the central repository for storing book and user information.

### Sequence of Events:

1. **Request Book:** The Customer requests a specific book from the Librarian.
2. **Check for a book:** The Librarian checks the Book System for the availability of the requested book.
3. **Return Book location if available:** The Book System checks the Database for the location of the book and returns the information to the Librarian.

4. **Go to location:** The Librarian retrieves the book from the specified location.
5. **Receive Book:** The Librarian gives the requested book to the Customer.
6. **Give Book:** The Customer receives the book.
7. **Add New Book:** A new book is added to the library's collection.
8. **Add Book:** The System records the new book in the Database.
9. **Book Added:** The Database confirms the addition of the new book.

The sequence diagram provides a clear and concise overview of the steps involved in requesting and receiving a book from the library, highlighting the interactions between the different actors and the flow of information throughout the process.

---

## Activity Diagram

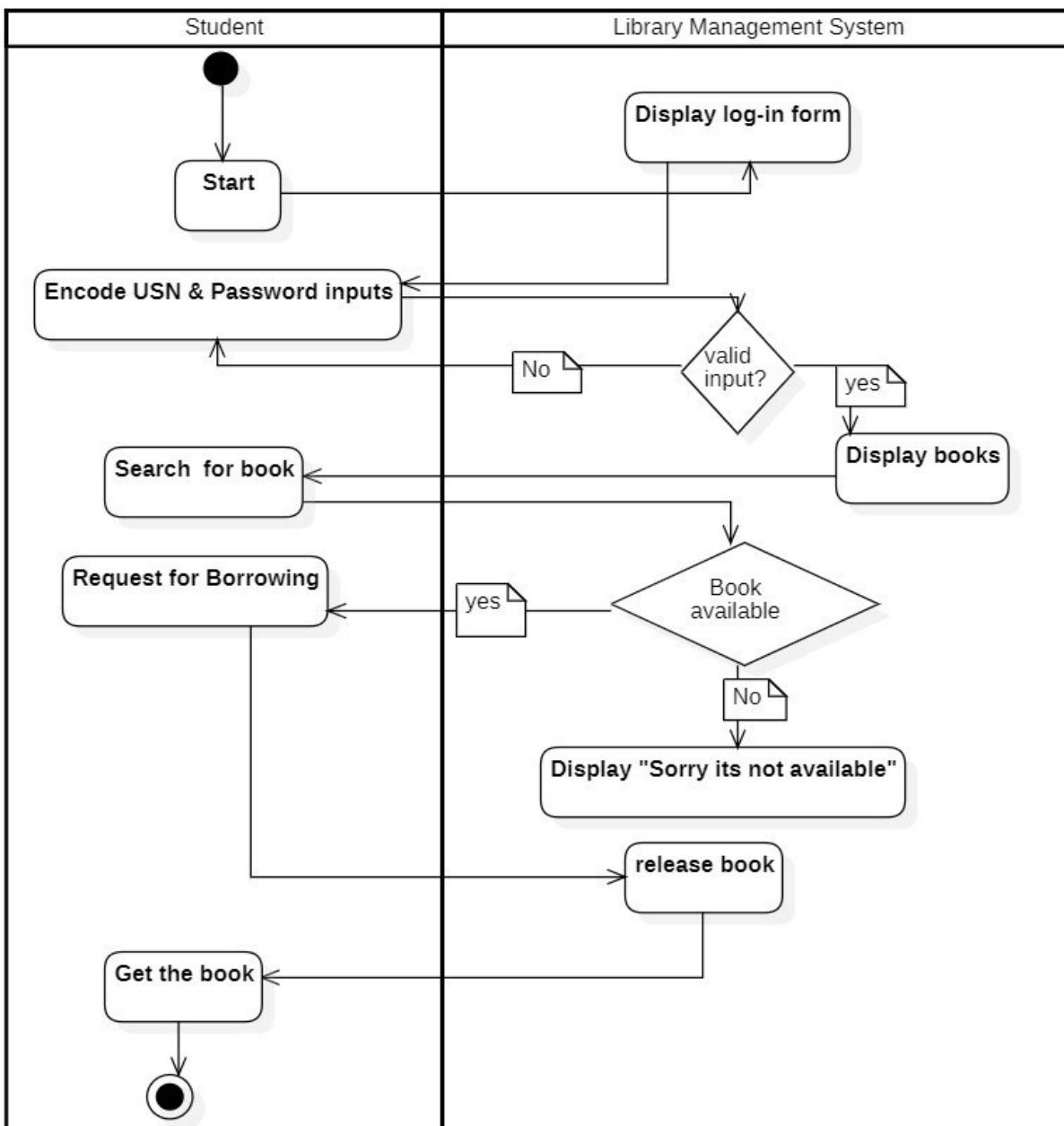


Fig 3.8

The activity diagram illustrates the process flow for a student borrowing a book from a Library Management System. It outlines the steps involved from the student's perspective and the system's response.

### Actors:

- **Student:** The individual who is borrowing the book.
- **Library Management System:** The system that handles the book borrowing process.

### **Activities:**

- **Start:** The process begins when the student initiates the book borrowing process.
- **Display log-in form:** The Library Management System displays a log-in form for the student to enter their credentials.
- **Encode USN & Password inputs:** The student enters their University Seat Number (USN) and password.
- **Valid input?:** The system validates the entered USN and password.
  - **No:** If the input is invalid, the system may display an error message or prompt the student to re-enter their credentials.
  - **Yes:** If the input is valid, the system proceeds to the next step.
- **Display books:** The system displays a list of available books to the student.
- **Search for book:** The student searches for the desired book from the list.
- **Book available?:** The system checks if the selected book is available for borrowing.
  - **No:** If the book is not available, the system displays a message like "Sorry, it's not available."
  - **Yes:** If the book is available, the system proceeds to the next step.
- **Request for Borrowing:** The student requests to borrow the selected book.
- **Get the book:** The system processes the request and allows the student to borrow the book.

The activity diagram provides a clear and concise overview of the book borrowing process, highlighting the key steps involved and the interactions between the student and the Library Management System.

---

## 4. Stock Maintenance System

### Problem Statement:

Design a software system to support a computerized stock maintenance network for a large manufacturing organization with multiple warehouses operating under a single brand. Each warehouse maintains its own local inventory management system to track stock levels, manage incoming and outgoing shipments, and generate reports. Warehouse staff at individual locations interact directly with their local systems to record stock movements, update inventory levels, and generate reports.

A central stock management system communicates with a central server that coordinates stock information across all warehouses, ensures real-time updates for stock availability, and manages stock replenishment orders. The central system allows for online order placement, order tracking, and inventory analysis.

The system requires robust recordkeeping, scalability to support multiple warehouses, and strict security measures to protect inventory data and prevent stock discrepancies. It must handle concurrent inventory updates and avoid stock-outs. Each warehouse will provide and maintain its software for local inventory management operations, while you are tasked with designing the software for the central stock management system, the online ordering platform, and the network connecting individual warehouses to the central server. The cost of the shared system will be apportioned to warehouses based on the volume of stock movements they process.

---

# Software Requirement Specification Document

classmate

Date \_\_\_\_\_  
Page \_\_\_\_\_

## STOCK MAINTENANCE SYSTEM

### 1) Introduction:

#### 1.1) Purpose:

The System aims to automate the management of stock inventory for businesses.

#### 1.2) Scope:

The System will allow businesses to manage stock records, monitor level of inventory, automate the reordering process and generate reports on stock transactions.

#### 1.3) Overview:

The System is designed to automate stock tracking and inventory management for businesses.

### 2) General Description:

The System will provide businesses with real-time stock information and automate key stock management processes such as reorder alerts, supplier-level tracking, and report generation.

Fig 4.1

### 3) Functional Requirements:

#### a) User Management:

The system will allow administrators to create, update and manage user accounts with different access levels.

#### b) Stock Management:

The system will allow users to add, update, delete stock items and also view current stock levels.

#### c) Reorder Alerts:

The system will automatically generate and send notifications when stock levels fall below the reorder threshold.

#### d) Reporting:

The system will generate reports on stock movements, including stock received, issued and current stock levels.

### 4) Interface Requirements:

#### a) User Interface:

The system will provide a web-based GUI for users to

manage stock, view reports and set reorder threshold.

#### b) Software Interface:

The system will integrate with a relational database (e.g., MySQL) to store stock & user data.

### 5) Performance Requirements:

#### a) Response Time:

The system should respond to user queries, such as stock searches, within 2 seconds under normal conditions.

#### b) Scalability:

The system should be able to handle up to 5000 stock items and 5000 users without performance degradation.

#### c) Error Rate:

The system should maintain an error rate of less than 0.05% for stock inventory operations.

#### d) Design Constraints:

##### a) Database Selection:

Fig 4.2

The system will use a relational database to store all staff and user data.

#### i) Platform Requirements:

The system will be a web application compatible with major browsers, including chrome, Firefox, etc.

#### ii) Non-Functional Requirements:

a) Security: The system will store credentials in an encrypted format (e.g. AES-256).

#### b) Portability:

The system will be compatible across different operating systems, including Windows, macOS and Linux.

#### c) Usability:

The system should have an uptime of 99.9%.

### 8) Preliminary Schedule & Budget:

#### Schedule:

Phase	Duration(weeks)
Project Inception	2
Requirements Gathering	4
System Design	3
Development	10
Testing	3
Deployment	2

Category	Estimated Cost(\$)
Personnel cost	150,000
Software Licenses	10,000
Infrastructure	12,000
Security	8,000
Training	5,000

Fig 4.3

## Class Diagram

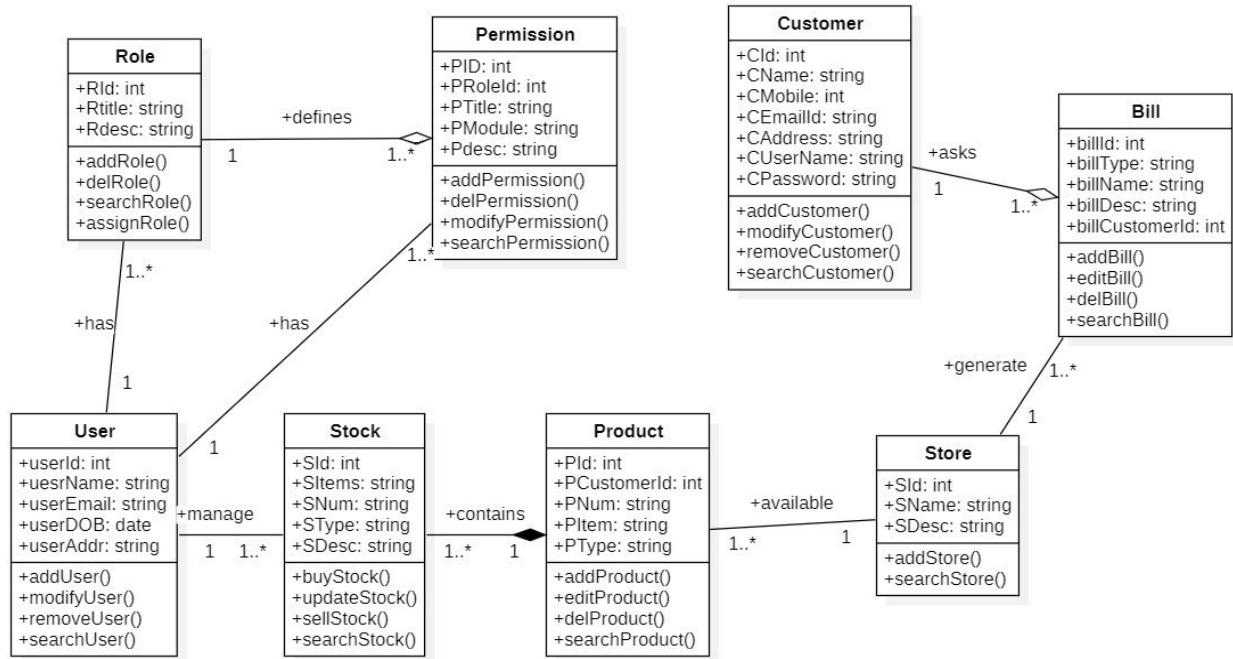


Fig 4.4

The class diagram illustrates the structure and relationships within a retail management system, highlighting key entities, their attributes, and the interactions between them.

### Key Classes

- **Role:** Represents different roles within the system (e.g., Manager, Cashier, Customer), with attributes like RoleID, RoleTitle, and RoleDesc.
- **Permission:** Represents permissions assigned to roles, with attributes like PermissionID, PermissionTitle, Module, and Access.
- **User:** Represents users of the system with attributes like UserID, UserName, Email, Date of Birth, and Address.
- **Customer:** Represents customers with attributes like CustomerID, Name, Mobile, Email, Address, and Username.
- **Stock:** Represents stock items with attributes like StockID, Items, Number, Type, and Description.
- **Product:** Represents products with attributes like ProductID, CustomerID, Number, Item, Type, and Price.
- **Store:** Represents stores with attributes like StoreID, Name, and Description.

- **Bill:** Represents bills generated for transactions, with attributes like BillID, BillType, BillName, BillDesc, and BillCustomerID.

## Relationships

- **Role defines Permission:** A Role can define multiple Permissions.
- **Role has User:** A User can be assigned to a specific Role.
- **Customer tasks Bill:** A Customer can generate multiple Bills.
- **Stock contains Product:** Stock items can contain multiple Products.
- **Store available Product:** A Store can have multiple Products available.

## Methods

Each class has methods for creating, modifying, deleting, and searching for objects within the class. For example:

- **Role:** addRole(), delRole(), searchRole(), assignRole()
- **Permission:** addPermission(), delPermission(), modifyPermission(), searchPermission()
- **User:** addUser(), modifyUser(), removeUser(), searchUser()
- **Customer:** addCustomer(), modifyCustomer(), removeCustomer(), searchCustomer()
- **Stock:** addStock(), updateStock(), sellStock(), searchStock()
- **Product:** addProduct(), editProduct(), delProduct(), searchProduct()
- **Store:** addStore(), searchStore()
- **Bill:** addBill(), editBill(), delBill(), searchBill()

The class diagram provides a clear and concise representation of the entities and their relationships within the retail management system, laying the foundation for the design and implementation of the software.

---

## State Diagram

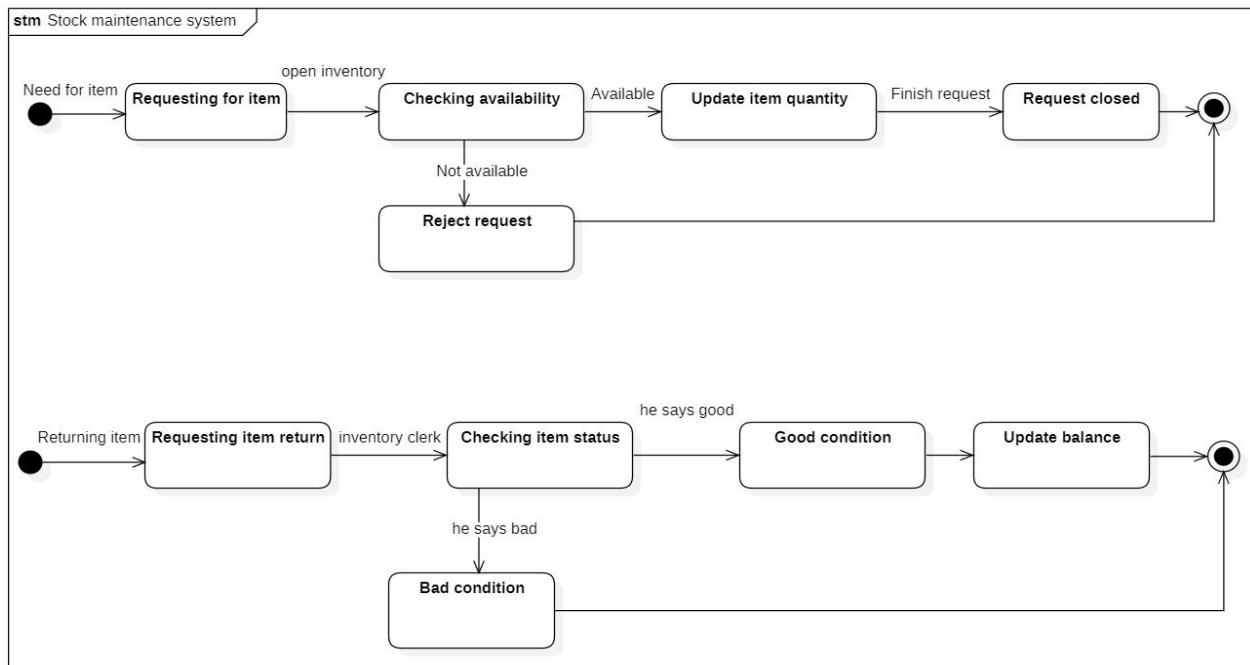


Fig 4.5

The statechart diagram illustrates the workflow for managing stock items within a Stock Maintenance System. It outlines the states and transitions involved in both requesting and returning items.

### States:

- Need for Item:** The initial state where a request for an item is made.
- Requesting for Item:** The state where the request for an item is being processed.
- Checking Availability:** The state where the system checks the availability of the requested item.
- Available:** The state where the requested item is available in stock.
- Update Item Quantity:** The state where the system updates the item quantity after the item is issued or returned.
- Finish Request:** The state where the request is successfully completed.
- Request Closed:** The final state after the request is processed.
- Reject Request:** The state where the request for an item is rejected due to unavailability.
- Returning Item:** The state where a request for returning an item is made.
- Requesting Item Return:** The state where the request for returning an item is being processed.

- **Checking Item Status:** The state where the condition of the returned item is being checked.
- **Good Condition:** The state where the returned item is in good condition.
- **Bad Condition:** The state where the returned item is in bad condition.
- **Update Balance:** The state where the stock balance is updated after the item is returned.

#### **Transitions:**

- From "Need for Item" to "Requesting for Item": When a request for an item is made.
- From "Requesting for Item" to "Checking Availability": After the request is received.
- From "Checking Availability" to "Available": If the item is available in stock.
- From "Checking Availability" to "Not Available": If the item is not available in stock.
- From "Available" to "Update Item Quantity": If the item is available, the quantity is updated.
- From "Update Item Quantity" to "Finish Request": After the quantity is updated.
- From "Finish Request" to "Request Closed": Upon successful completion of the request.
- From "Checking Availability" to "Reject Request": If the item is not available.
- From "Returning Item" to "Requesting Item Return": When a request for returning an item is made.
- From "Requesting Item Return" to "Checking Item Status": After the return request is received.
- From "Checking Item Status" to "Good Condition": If the returned item is in good condition.
- From "Checking Item Status" to "Bad Condition": If the returned item is in bad condition.
- From "Good Condition" to "Update Balance": If the item is in good condition, the stock balance is updated.
- From "Update Balance" to the final state: After the stock balance is updated.

The statechart diagram provides a clear and concise overview of the stock management process, highlighting the different stages involved in requesting and returning items, as well as the possible outcomes and transitions between states.

---

## Use Case Diagram

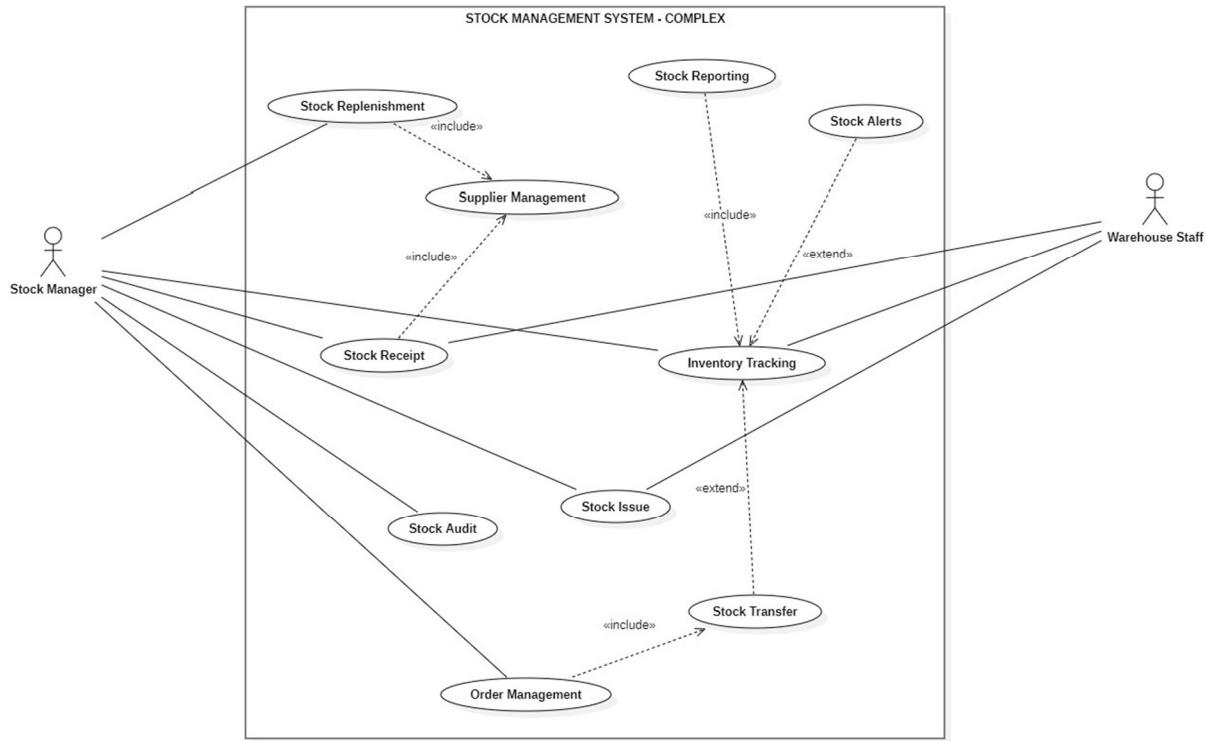


Fig 4.6

The use case diagram illustrates the various functionalities and interactions within a complex Stock Management System. It highlights the different use cases available to both Stock Managers and Warehouse Staff, as well as the relationships between these use cases.

### Actors:

- **Stock Manager:** Represents the individual responsible for managing the stock in the system.
- **Warehouse Staff:** Represents the staff members working in the warehouse who are involved in various stock-related activities.

### Use Cases:

- **Stock Replenishment:** Handles the process of ordering and receiving new stock items.
- **Stock Reporting:** Generates reports on stock levels, inventory trends, and other relevant metrics.
- **Stock Alerts:** Issues alerts for low stock levels, potential stockouts, and other critical inventory situations.
- **Supplier Management:** Manages relationships with suppliers, including adding, modifying, and deleting supplier information.

- **Stock Receipt:** Handles the process of receiving incoming stock shipments.
- **Inventory Tracking:** Tracks the movement of stock items within the warehouse, including receiving, issuing, and transferring stock.
- **Stock Issue:** Handles the process of issuing stock items from the warehouse, such as fulfilling orders or transferring stock to other locations.
- **Stock Audit:** Performs regular audits of stock levels to ensure accuracy and identify discrepancies.
- **Stock Transfer:** Handles the transfer of stock items between different locations within the warehouse or across different warehouses.
- **Order Management:** Manages orders for stock items, including order placement, processing, and fulfillment.

#### **Relationships:**

- **Include:** "Stock Replenishment" includes "Supplier Management," indicating that supplier management is a necessary part of the stock replenishment process.
- **Include:** "Stock Reporting" includes "Inventory Tracking," indicating that inventory tracking data is essential for generating stock reports.
- **Include:** "Stock Receipt" includes "Supplier Management," indicating that supplier information is required for receiving stock shipments.
- **Include:** "Stock Issue" includes "Inventory Tracking," indicating that stock issue transactions affect inventory levels.
- **Include:** "Stock Audit" includes "Inventory Tracking," as audits are performed on the tracked inventory data.
- **Include:** "Order Management" includes "Inventory Tracking," as order fulfillment relies on accurate inventory information.
- **Extend:** "Stock Alerts" extends "Inventory Tracking," indicating that stock alerts are generated based on changes in inventory levels.
- **Extend:** "Stock Issue" extends "Inventory Tracking," indicating that stock issuance affects inventory levels.
- **Extend:** "Stock Transfer" extends "Inventory Tracking," indicating that stock transfers affect inventory levels.

The use case diagram provides a comprehensive overview of the stock management system's functionalities, highlighting the interactions between different actors and use cases, as well as the relationships between various components of the system.

---

## Sequence Diagram

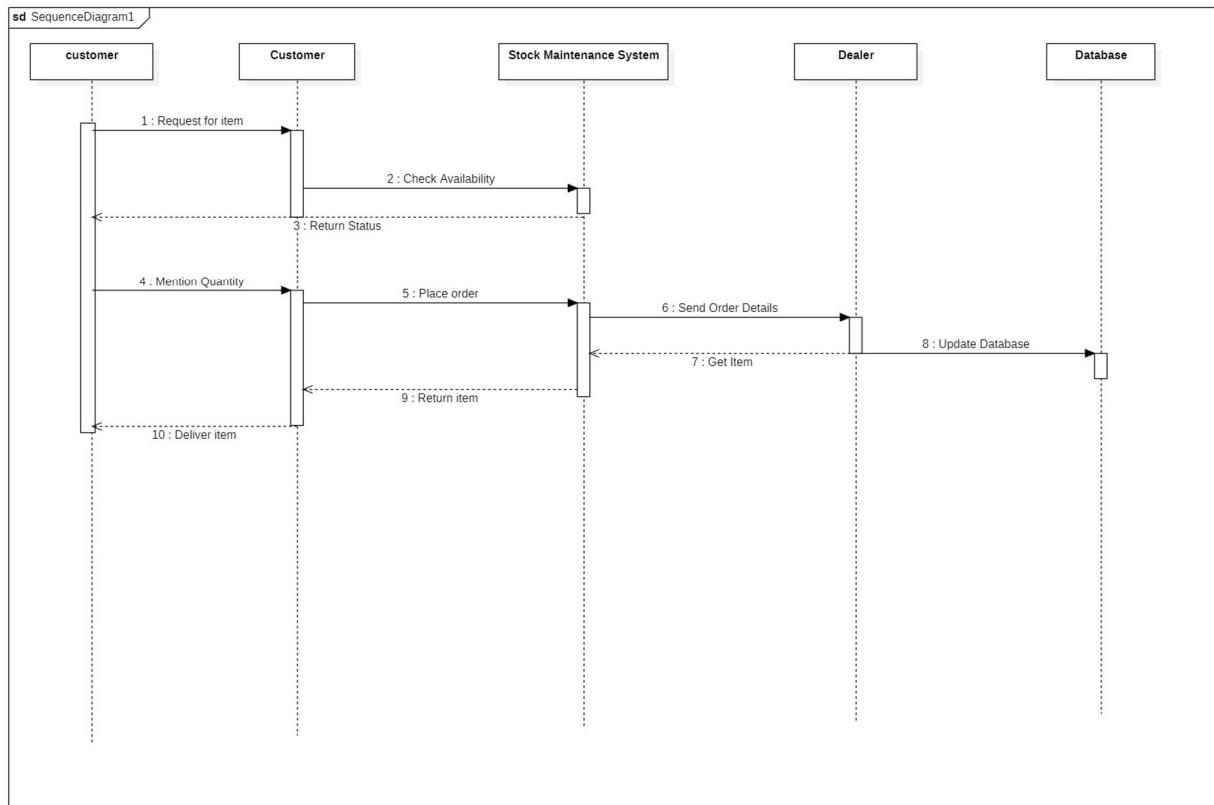


Fig 4.7

The sequence diagram illustrates the flow of interactions involved in a stock maintenance system, specifically focusing on the process of ordering an item. It outlines the sequence of messages exchanged between the Customer, Stock Maintenance System, Dealer, and Database.

### Actors:

- **Customer:** Represents the individual who is placing an order.
- **Stock Maintenance System:** Represents the system used to manage stock information and orders.
- **Dealer:** Represents the entity responsible for fulfilling the order.
- **Database:** Represents the central repository for storing stock and order information.

### Sequence of Events:

1. **Request for Item:** The Customer requests a specific item from the Stock Maintenance System.
2. **Check Availability:** The Stock Maintenance System checks the database for the availability of the requested item.

3. **Return Status:** The Stock Maintenance System sends the availability status (available or unavailable) back to the Customer.
4. **Mention Quantity:** The Customer specifies the desired quantity of the item.
5. **Place order:** The Customer places an order for the specified quantity of the item.
6. **Send Order Details:** The Stock Maintenance System sends the order details to the Dealer.
7. **Get Item:** The Dealer retrieves the ordered item from the stock.
8. **Update Database:** The Dealer updates the database to reflect the reduction in stock quantity.
9. **Return Item:** The Dealer delivers the ordered item to the Customer.
10. **Deliver Item:** The Customer receives the ordered item.

The sequence diagram provides a clear and concise overview of the steps involved in placing and fulfilling an order within the stock maintenance system, highlighting the interactions between the different actors and the flow of information throughout the process.

---

## Activity Diagram

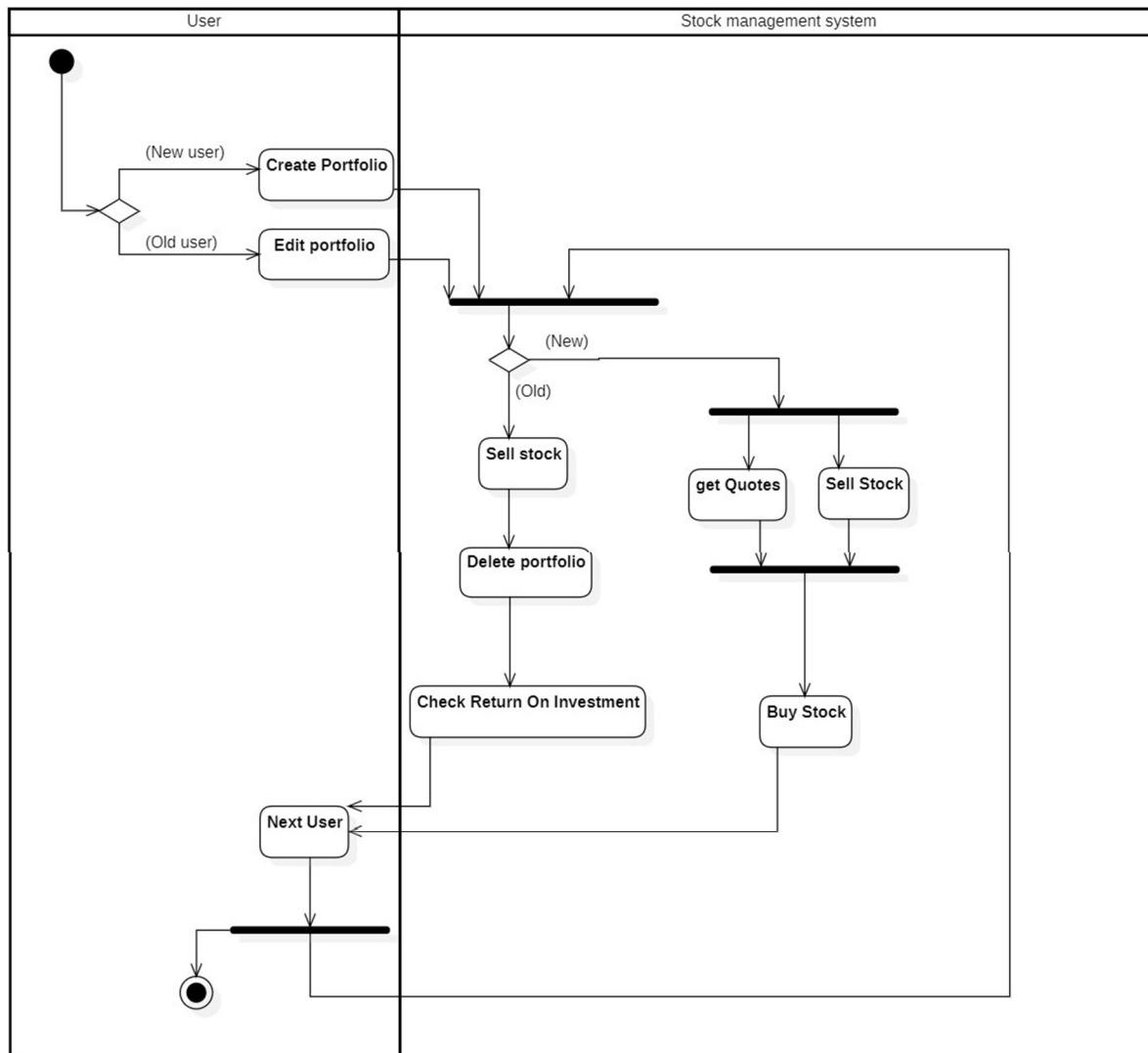


Fig 4.8

The activity diagram illustrates the workflow for a user interacting with a Stock Management System. It outlines the various actions a user can perform within the system.

### Actors:

- **User:** Represents the individual interacting with the Stock Management System.

### Activities:

- **Create Portfolio:** This activity allows new users to create a portfolio to track their investments.
- **Edit Portfolio:** Existing users can edit their existing portfolios by adding, removing, or modifying investments.

- **Sell Stock:** This activity enables users to sell stocks from their portfolio.
- **Get Quotes:** The system retrieves real-time stock quotes to assist users in making informed decisions.
- **Sell Stock:** The system processes the user's sell order.
- **Buy Stock:** This activity allows users to buy new stocks to add to their portfolio.
- **Delete Portfolio:** Users can delete their portfolio if they no longer wish to track it.
- **Check Return On Investment:** This activity calculates and displays the return on investment for the user's portfolio.
- **Next User:** The system transitions to the next user to begin their interaction with the system.

#### **Conditions:**

- The diagram includes conditional branches to differentiate between "New User" and "Old User" actions.

The activity diagram provides a clear and concise overview of the user interactions within the Stock Management System, highlighting the different actions available to users and the flow of the system.

---

## 5. Passport Automation System

### **Problem Statement:**

Design a software system to support a computerized Passport Automation System for a government agency. The system should streamline the passport application process, reducing manual intervention and ensuring efficiency. Passport offices maintain their own local computer systems to handle application intake, document verification, and initial data entry. Passport officers interact directly with their local systems to capture applicant information, verify documents, and update application status.

A central passport processing system communicates with a central server that coordinates application processing, manages document verification, and tracks passport issuance across all offices. The central system allows applicants to track their application status online, submit supporting documents electronically, and make online payments for passport fees.

The system requires robust data security, scalability to handle a large volume of applications, and strict adherence to government regulations and data privacy standards. It must handle concurrent applications, ensure data integrity, and prevent fraudulent activities. Each passport office will provide and maintain its software for local operations, while you are tasked with designing the software for the central passport processing system, the online applicant portal, and the network connecting individual passport offices to the central server. The cost of the shared system will be managed by the government agency.

---

# Software Requirement Specification Document

classmate Date _____ Page _____	
PASSPORT AUTOMATION SYSTEM	
1) Introduction:	
1.1) Purpose:	The purpose of this document is to describe the Software Requirements Specification (SRS) for the system.
1.2) Scope:	The system is designed to handle the entire lifecycle of passport processing, from online application submission to passport issuance.
1.3) Overview:	This document outlines the requirements for the Passport Automation System. It describes the system's functionality, performance requirements etc..
2) General Description:	The system is a web-based platform that simplifies the passport application process. It allows users to apply for passports, submit documents, schedule appointments and so much more.
3) Functional Requirements:	<ul style="list-style-type: none"><li>• User Registration and Login: Applicants can create accounts, log in, and reset passwords if needed.</li><li>• Passport Application: Users can fill out the application form, upload required docs, and submit them online.</li><li>• Appointment Scheduling: Applicants can schedule appointments for document verification or biometric data collection.</li><li>• Application Status Tracking: Users can track their application status in real-time.</li></ul>
4) Interface Requirements:	<ul style="list-style-type: none"><li>• User Interface: A user-friendly interface for applicants to submit applications, track status, and schedule appointments.</li><li>• Admin Interface: Integration with external govt. databases for identity verification.</li></ul>

Fig 5.1

### 5) Performance Requirements:

#### a) Response Time:

The system should respond to user requests within 3 seconds.

#### b) Error Rate:

The system should maintain an error rate of less than 0.1%.

### 6) Design Constraints:

#### a) Database Selection:

A relational database will be used to store applicant information, documents uploads etc..

#### b) Platform Requirements:

The system will be developed as a web application, compatible across multiple browsers.

### 7) Non-functional Requirements:

#### a) Security:

User data, including personal information, must be securely encrypted.

#### b) Usability:

The system should offer a simple, intuitive user interface for both

applicants and administrators.

### 8) Preliminary Schedule and Budget:

Phase	Duration (weeks)
Project Initiation	2
Requirement Gathering	4
System Desgn	3
Development	10
Testing	3
Deployment	2

#### Budget:

Category	Cost (\$)
Personnel cost	200,000
Software Licenses	10,000
Infrastructure	15,000
Security	10,000
Training	7500
Marketing	2,500

Fig 5.2

## Class Diagram

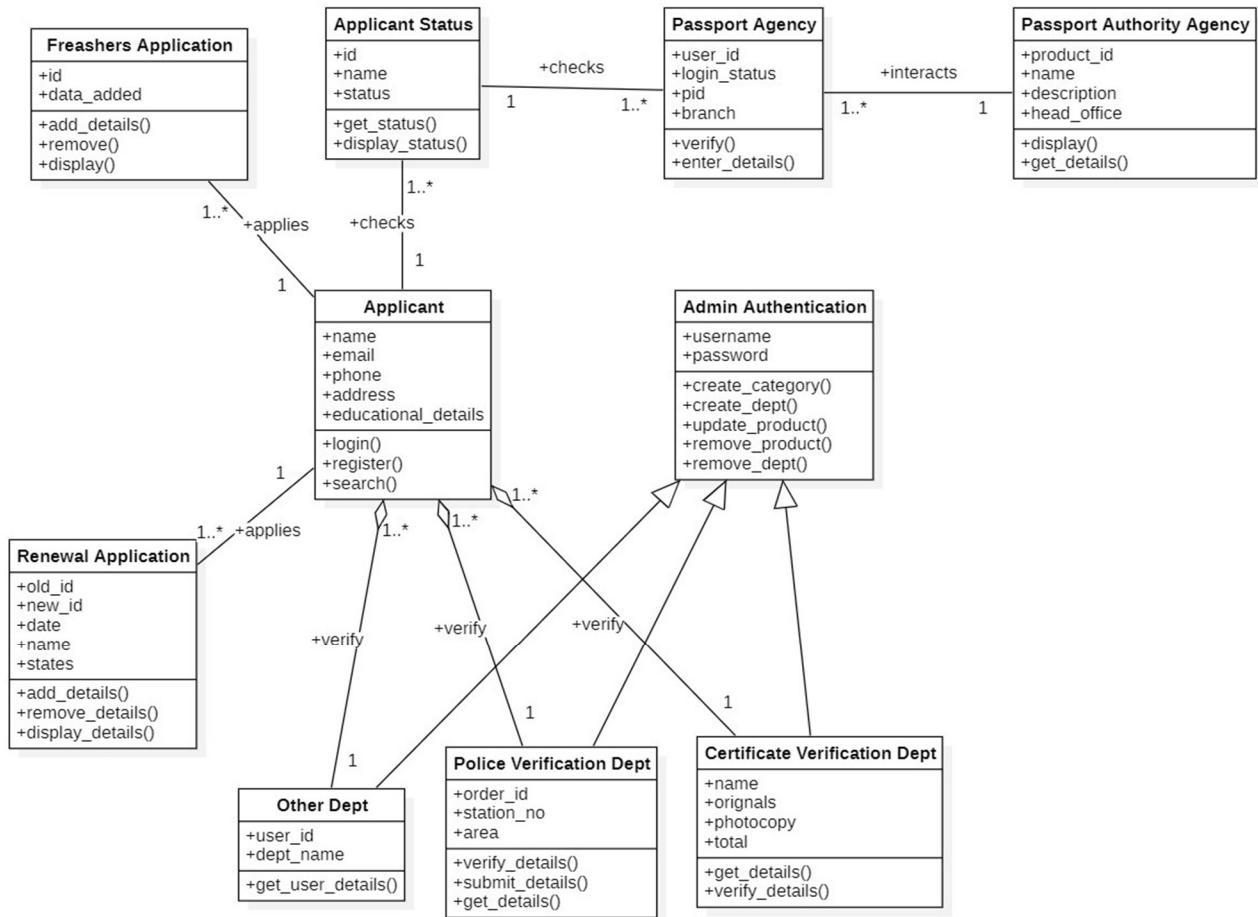


Fig 5.3

The class diagram illustrates the structure and relationships within a Passport Automation System. It highlights the key classes, their attributes, and the associations between them.

## Classes:

- **Freshers Application:** Represents new passport applications, with attributes like id, data\_added, and states.
  - **Applicant Status:** Tracks the status of an application, with attributes like id, name, and status.

- **Applicant:** Represents the individual applying for a passport, with attributes like name, email, phone, address, and educational details.
- **Renewal Application:** Represents applications for passport renewal, with attributes like old\_id, new\_id, date, and name.
- **Passport Agency:** Represents a passport issuing agency, with attributes like pid, name, branch, and head\_office.
- **Passport Authority Agency:** Represents the central authority responsible for passport issuance, with attributes like product\_id, name, and description.
- **Admin Authentication:** Represents the authentication mechanism for system administrators, with attributes like username and password.
- **Police Verification Dept:** Represents the department responsible for police verification, with attributes like order\_id, area, station\_no, and user\_id.
- **Certificate Verification Dept:** Represents the department responsible for certificate verification, with attributes like name, originals, photocopies, and total.
- **Other Dept:** Represents other departments involved in the passport application process.

### **Relationships:**

- **One-to-Many (1..\*)**
  - Freshers Application has one Applicant Status.
  - Applicant applies for one Freshers Application and many Renewal Applications.
  - Passport Agency interacts with many Passport Authority Agencies.
  - Admin Authentication interacts with all other classes in the system.
  - Applicant interacts with Police Verification Dept, Certificate Verification Dept, and Other Dept.
- **One-to-One (1..1)**
  - Applicant Status checks Freshers Application.
  - Applicant Status checks Renewal Application.

### **Methods:**

Each class would likely have methods for interacting with its attributes and related classes. For example:

- **Freshers Application:** add\_details(), remove(), display()
- **Applicant Status:** get\_status(), display\_status()
- **Applicant:** register(), search()
- **Admin Authentication:** create\_category(), create\_dept(), update\_product(), remove\_product(), remove\_dept()
- **Police Verification Dept:** verify\_details(), get\_details(), get\_user\_details(), submit\_details()
- **Certificate Verification Dept:** verify\_details(), get\_details()

The class diagram provides a clear and concise representation of the classes and their relationships within the Passport Automation System, laying the foundation for the object-oriented design and implementation of the system.

---

## State Diagram

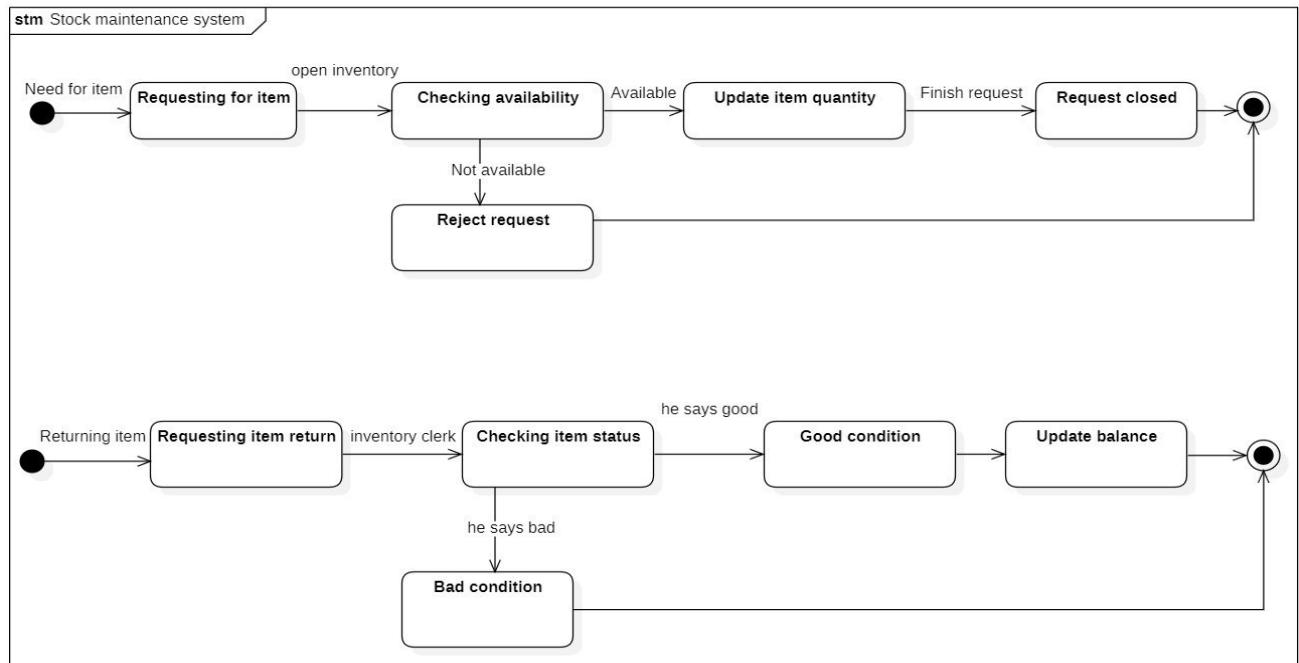


Fig 5.4

The statechart diagram illustrates the workflow for managing stock items within a Stock Maintenance System. It outlines the states and transitions involved in both requesting and returning items.

### States:

- **Need for Item:** The initial state where a request for an item is made.
- **Requesting for Item:** The state where the request for an item is being processed.
- **Checking Availability:** The state where the system checks the availability of the requested item.
- **Available:** The state where the requested item is available in stock.
- **Update Item Quantity:** The state where the system updates the item quantity after the item is issued or returned.
- **Finish Request:** The state where the request is successfully completed.
- **Request Closed:** The final state after the request is processed.
- **Reject Request:** The state where the request for an item is rejected due to unavailability.
- **Returning Item:** The state where a request for returning an item is made.

- **Requesting Item Return:** The state where the request for returning an item is being processed.
- **Checking Item Status:** The state where the condition of the returned item is being checked.
- **Good Condition:** The state where the returned item is in good condition.
- **Bad Condition:** The state where the returned item is in bad condition.
- **Update Balance:** The state where the stock balance is updated after the item is returned.

### **Transitions:**

- From "Need for Item" to "Requesting for Item": When a request for an item is made.
- From "Requesting for Item" to "Checking Availability": After the request is received.
- From "Checking Availability" to "Available": If the item is available in stock.
- From "Checking Availability" to "Not Available": If the item is not available in stock.
- From "Available" to "Update Item Quantity": If the item is available, the quantity is updated.
- From "Update Item Quantity" to "Finish Request": After the quantity is updated.
- From "Finish Request" to "Request Closed": Upon successful completion of the request.
- From "Checking Availability" to "Reject Request": If the item is not available.
- From "Returning Item" to "Requesting Item Return": When a request for returning an item is made.
- From "Requesting Item Return" to "Checking Item Status": After the return request is received.
- From "Checking Item Status" to "Good Condition": If the returned item is in good condition.
- From "Checking Item Status" to "Bad Condition": If the returned item is in bad condition.
- From "Good Condition" to "Update Balance": If the item is in good condition, the stock balance is updated.

The statechart diagram provides a clear and concise overview of the stock management process, highlighting the different stages involved in requesting and returning items, as well as the possible outcomes and transitions between states.

## Use Case Diagram

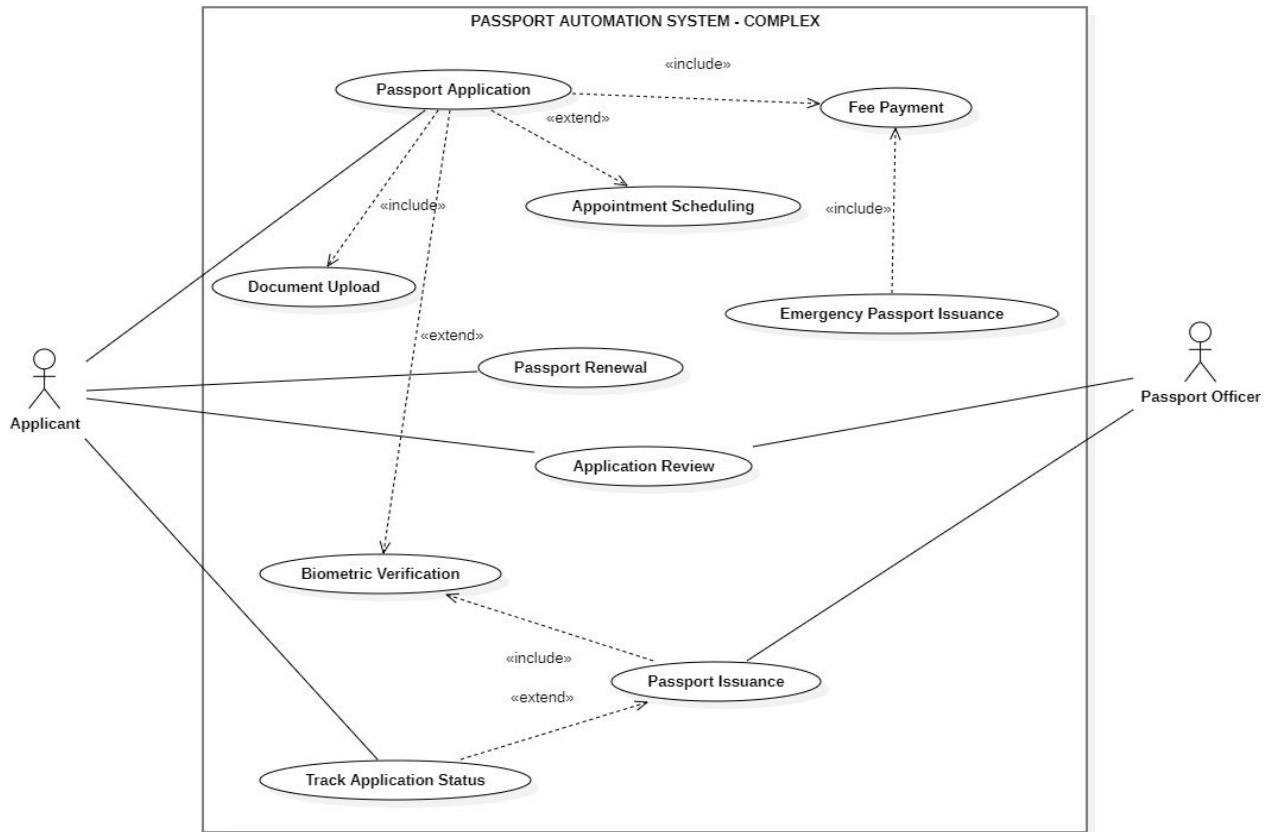


Fig 5.5

The use case diagram illustrates the various functionalities and interactions within a complex Passport Automation System. It highlights the different use cases available to both Applicants and Passport Officers, as well as the relationships between these use cases.

### **Actors:**

- **Applicant:** Represents the individual applying for a passport.
- **Passport Officer:** Represents the officer responsible for processing passport applications.

### **Use Cases:**

- **Passport Application:** The core process of applying for a passport, including submitting the application form and required documents.

- **Fee Payment:** The process of paying the required fees for the passport application.
- **Appointment Scheduling:** Scheduling an appointment for biometric verification and document submission.
- **Document Upload:** The process of uploading supporting documents electronically as part of the application.
- **Passport Renewal:** The process of renewing an existing passport.
- **Emergency Passport Issuance:** Handles emergency passport applications requiring expedited processing.
- **Application Review:** The process of reviewing the application and supporting documents.
- **Biometric Verification:** The process of capturing biometric data (fingerprints and photograph) of the applicant.
- **Passport Issuance:** The final stage of the process where the passport is issued to the applicant.
- **Track Application Status:** Allows applicants to track the status of their passport application online.

### **Relationships:**

- **Include:** "Fee Payment" is included in "Passport Application," indicating that fee payment is an integral part of the application process.
- **Include:** "Appointment Scheduling" is included in "Passport Application," indicating that scheduling an appointment is part of the application process.
- **Include:** "Document Upload" is included in "Passport Application," indicating that document upload is part of the application process.
- **Include:** "Emergency Passport Issuance" includes "Fee Payment," indicating that fee payment is required for emergency passport applications.
- **Include:** "Biometric Verification" is included in "Passport Issuance," indicating that biometric verification is necessary before passport issuance.
- **Include:** "Track Application Status" is included in all other use cases, indicating that applicants can track the status throughout the process.
- **Extend:** "Passport Renewal" extends "Passport Application," indicating that passport renewal follows a similar process to a new passport application with some variations.

- **Extend:** "Emergency Passport Issuance" extends "Passport Application," indicating that emergency passport issuance follows the general application process with expedited procedures.

The use case diagram provides a comprehensive overview of the passport automation system's functionalities, highlighting the interactions between different actors and use cases, as well as the relationships between various components of the system.

---

## Sequence Diagram

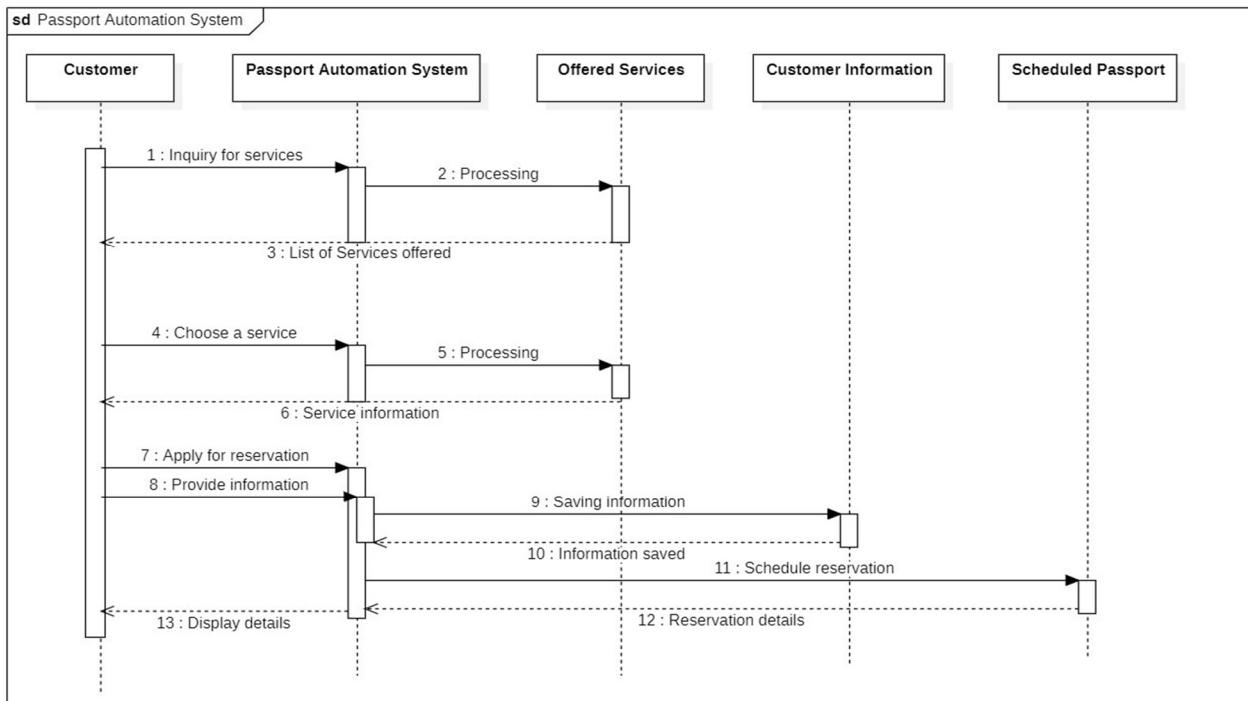


Fig 5.6

The sequence diagram illustrates the flow of interactions involved in a Passport Automation System, specifically focusing on the process of scheduling a passport service. It outlines the sequence of messages exchanged between the Customer, Passport Automation System, Offered Services, Customer Information, and Scheduled Passport components.

### Actors:

- **Customer:** Represents the individual who is requesting a passport service.
- **Passport Automation System:** Represents the system that manages the passport application process.
- **Offered Services:** Represents the module responsible for displaying available passport services.
- **Customer Information:** Represents the module responsible for collecting and storing customer information.

- **Scheduled Passport:** Represents the module responsible for scheduling appointments and storing reservation details.

### **Sequence of Events:**

1. **Inquiry for services:** The Customer initiates the process by inquiring about available passport services.
2. **Processing:** The Passport Automation System receives the inquiry and begins processing it.
3. **List of Services offered:** The Offered Services module provides a list of available passport services to the Passport Automation System.
4. **Choose a service:** The Customer selects a specific service from the list.
5. **Processing:** The Passport Automation System processes the service selection.
6. **Service information:** The Passport Automation System provides information about the selected service to the Customer.
7. **Apply for reservation:** The Customer applies for a reservation for the selected service.
8. **Provide information:** The Customer provides the necessary information for the reservation, such as personal details and appointment preferences.
9. **Saving information:** The Customer Information module receives and saves the customer's information.
10. **Information saved:** The Customer Information module confirms that the information has been saved.
11. **Schedule reservation:** The Scheduled Passport module schedules the reservation based on the customer's information and available slots.
12. **Reservation details:** The Scheduled Passport module provides the reservation details (date, time, location) to the Customer.
13. **Display details:** The Passport Automation System displays the reservation details to the Customer.

The sequence diagram provides a clear and concise overview of the steps involved in scheduling a passport service, highlighting the interactions between the different components of the system and the flow of information throughout the process.

## Activity Diagram

Action1

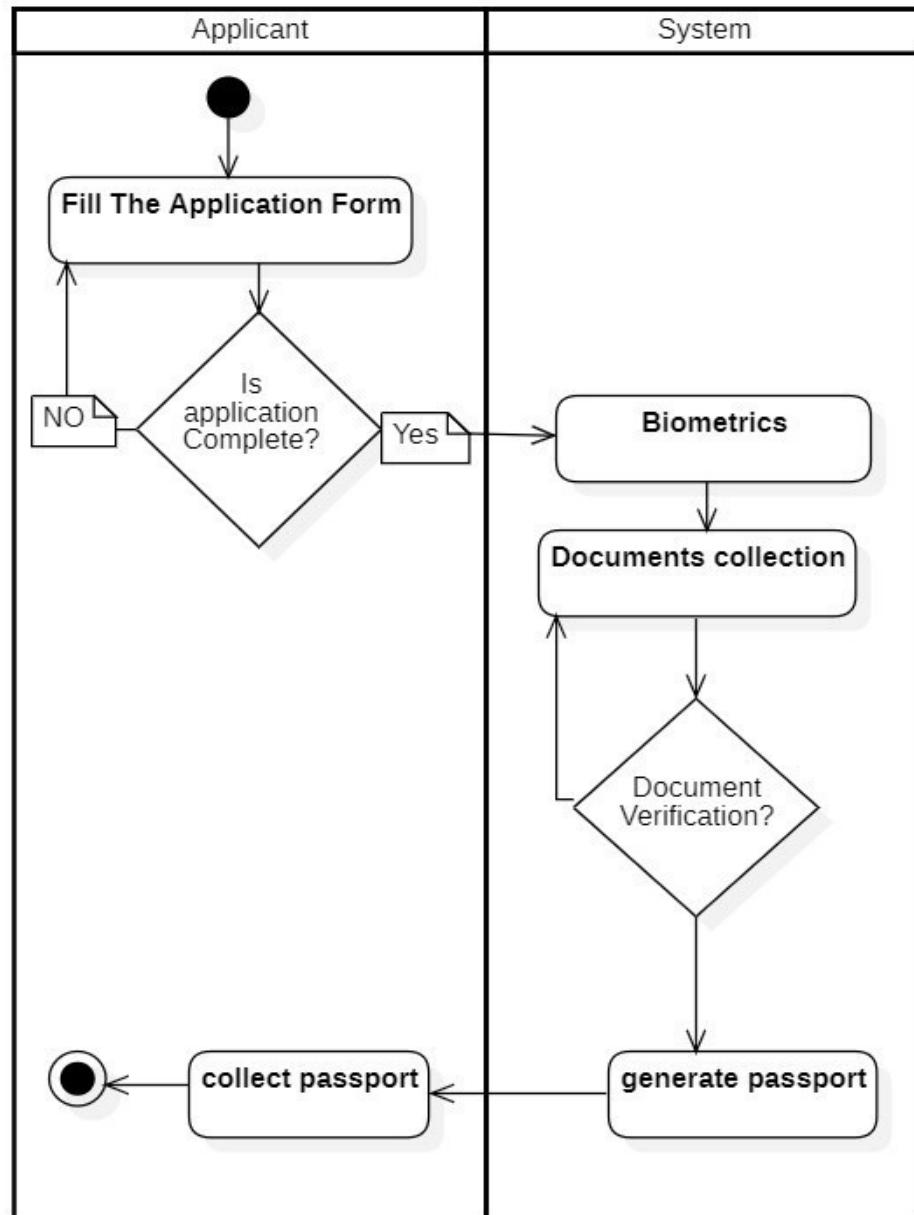


Fig 5.7

The activity diagram illustrates the workflow for processing a passport application. It outlines the steps involved from the applicant's perspective and the system's actions.

#### Actors:

- **Applicant:** The individual applying for the passport.
- **System:** Represents the passport processing system.

#### Activities:

- **Action1:** The initial state where the application process begins.
- **Fill The Application Form:** The applicant fills out the passport application form with the required information.
- **Is application Complete?:** The system checks if the applicant has filled out the application form completely.
  - **No:** If the application is incomplete, the applicant is prompted to complete the form.
  - **Yes:** If the application is complete, the process moves to the next step.
- **Biometrics:** The applicant undergoes biometric data capture (e.g., fingerprints, photograph).
- **Documents collection:** The applicant submits the required supporting documents (e.g., proof of address, identification).
- **Document Verification?:** The system verifies the authenticity of the submitted documents.
- **generate passport:** If all checks are successful, the system generates the passport.
- **collect passport:** The applicant collects the issued passport.

The activity diagram provides a clear and concise overview of the passport application process, highlighting the key steps involved and the interactions between the applicant and the system.

---