B.M.S. COLLEGE OF ENGINEERING BENGALURU

Autonomous Institute, Affiliated to VTU

Lab Record

Software Engineering and Object-Oriented Modeling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

ARJUN PRABHAKARAN

1BM22CS053

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 Analysis and Design(22CS6PCSEO) laboratory has been carried out by ARJUN PRABHAKARAN (1BM22CS053) during the 5th Semester Oct24-Jan2025.

Signature of the Faculty Incharge:

NAME OF THE FACULTY: ANUSHA S

Assistant Professor

Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

1. Hotel Management System

Problem Statement

The goal is to develop a Hotel Management System that efficiently manages hotel operations, including room bookings, customer check-ins and check-outs, payments, and inventory. The system should streamline processes for both hotel staff and guests, ensuring an easy-to-use interface for reservations, billing, and tracking room availability. It should help reduce manual errors, improve customer service, and enhance overall operational efficiency.

Software Requirement Specification(SRS)

1. Introduction

1.1 Purpose

This document outlines the requirements for a Hotel Management System (HMS) to automate and streamline hotel operations, including reservations, guest management, billing, and housekeeping.

1.2 Scope

The system will manage reservations, check-ins/outs, billing, room availability, and inventory. It will serve hotels of all sizes, offering features for front desk management, housekeeping, restaurant billing, and reporting.

1.3 Overview

The document covers:

- **General Description**: Users, features, and benefits.
- **Functional Requirements**: Core system functionalities.
- **Interface Requirements**: External system interactions.
- **Performance Requirements**: System performance expectations.
- **Design Constraints**: Any technical limitations.
- **Non-functional Requirements**: Security, usability, etc.
- **Preliminary Budget & Time**: Estimated project budget and timeline.

2. General Description

2.1 User Characteristics

- **Hotel Managers**: Oversee operations, generate reports.
- **Receptionists**: Handle bookings, check-ins/outs.
- **Housekeeping**: Update room statuses.
- **Accounting**: Manage billing and financial reports.

- **Guests**: Make bookings, check-in/check-out.

2.2 Features

- **Reservation Management**: Book and manage rooms.
- **Guest Profiles**: Store guest info and preferences.
- **Room Management**: Track room status (available, occupied).
- **Billing & Payments**: Handle payments and invoicing.
- **Reporting**: Generate performance and occupancy reports.
- **Inventory**: Track hotel supplies and resources.
- **Housekeeping**: Update room cleaning status.
- **User Authentication**: Secure login and access control.

2.3 Benefits

- **Efficiency**: Reduces manual tasks and errors.
- **Guest Experience**: Personalizes services for guests.
- **Real-time Data**: Provides up-to-date information.
- **Cost Management**: Optimizes resources and inventory.
- **Scalability**: Supports hotel chains and large operations.

3. Functional Requirements

- **Reservation Management**: Create, modify, cancel, and confirm bookings.
- **Guest Profiles**: Store and retrieve guest data.
- **Room Management**: Track room status (occupied, vacant).
- **Billing**: Generate bills and process payments.
- **Housekeeping**: Update and track cleaning tasks.

4. Interface Requirements

- **Web Interface**: Accessible to receptionists and managers.
- **Payment Gateway Integration**: Process online payments securely.
- **External Booking System Integration**: Sync with OTAs (Online Travel Agencies) like Booking.com.

5. Performance Requirements

- **Response Time**: System must respond to user actions within 2 seconds.
- **Scalability**: Support up to 500 rooms and 100 concurrent users.

6. Design Constraints

- **Platform**: The system will be cloud-based for remote access.
- **Security**: Must comply with GDPR and PCI DSS standards for data protection.

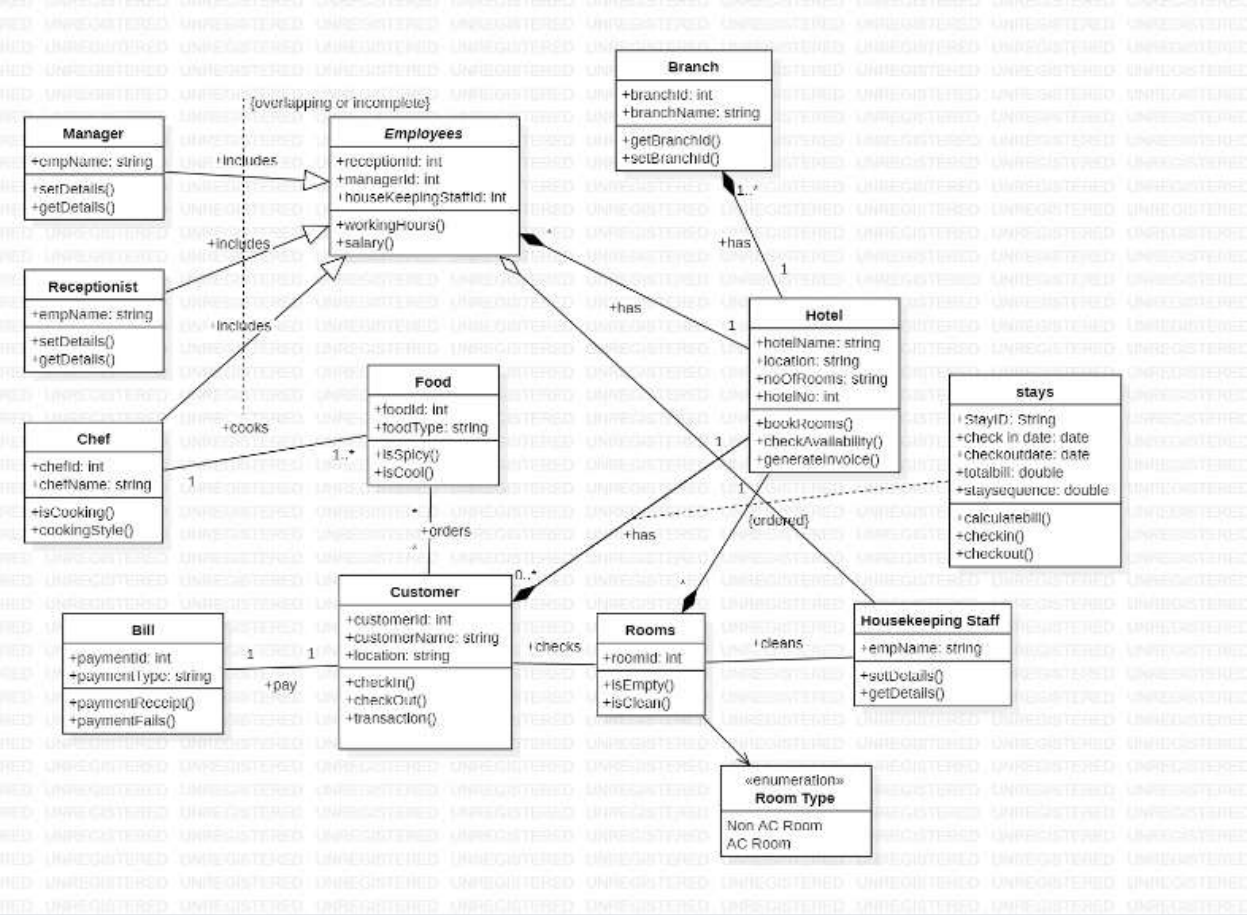
7. Non-functional Requirements

- **Reliability**: 99.9% uptime.
- **Usability**: Intuitive UI/UX for staff and guests.
- **Security**: Role-based access control and encryption of sensitive data.
- **Backup**: Daily backups of all critical data.

8. Preliminary Budget and Time

- **Budget**: Estimated at $50,000 for development, testing, and deployment.
- **Timeline**: Project completion in 6 months.

Class Diagram



Description

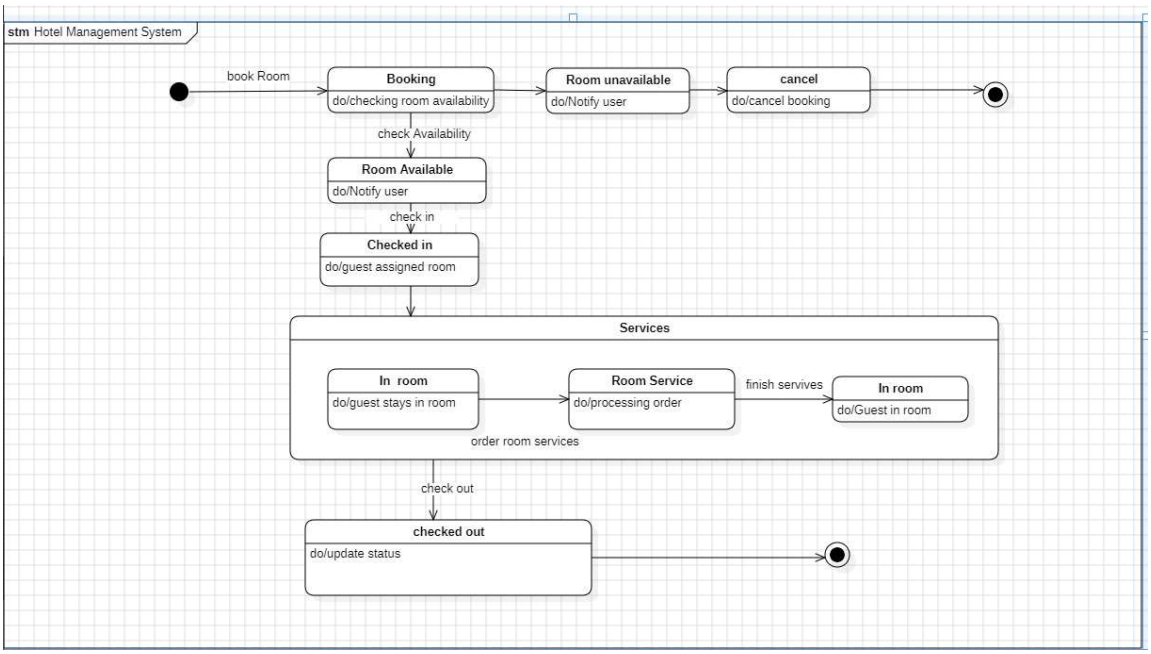
Room: Contains attributes like RoomNo (for the room number), RoomType (e.g., Single, Double), and details (e.g., features like bed size, view).

Guest: Holds details of the guest such as GuestID, Name, Age, and ContactDetails (email/phone).

Reservation: Manages bookings with attributes like makingID (reservation ID), RoomID (assigned room), and GuestID (associated guest).

Staff: Stores information about hotel staff, including StaffID, Name, Role (e.g., receptionist, housekeeper), and ContactDetails.

State Diagram



Description

Start: The process begins.
Book Room: The guest initiates a booking process.
Check Room Availability: The system checks if rooms are available for the selected dates.

- **Room Available**: If a room is available, the guest is notified and can proceed.
- **Room Unavailable**: If no rooms are available, the system notifies the user.

Check-in: The guest checks in to the hotel.

- **Guest Assigned Room**: The system assigns the guest to a room and updates the status.
- **In Room**: The guest enters and stays in the room.

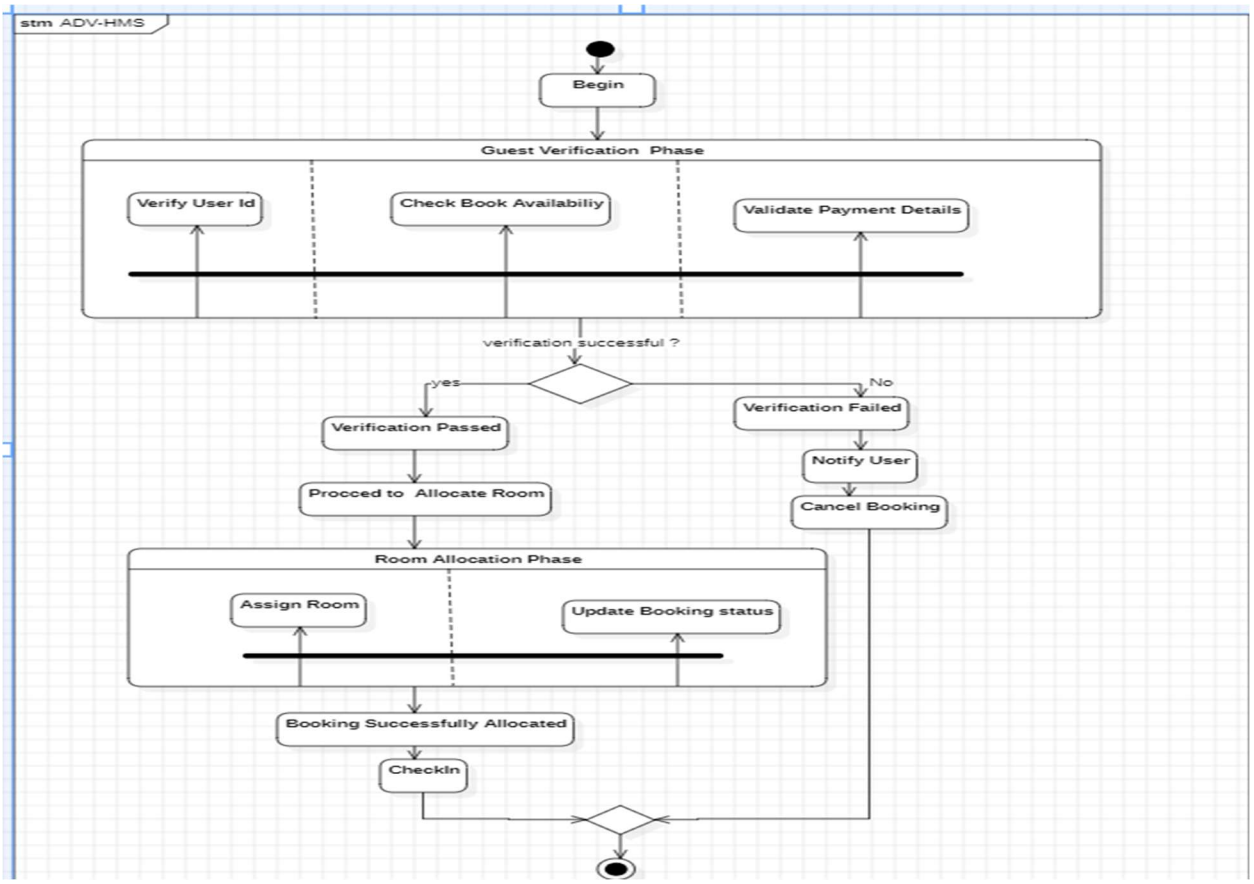
Room Service: The guest requests room service, which is processed by the system.

- **Processing Order**: The system processes the service order.
- **Finish Services**: Room service is completed, and the status is updated.

Cancel Booking: If the guest wants to cancel the booking, the system handles the cancellation.
Check-out: The guest checks out of the room.

- **Checked Out**: The system updates the guest status and room availability.

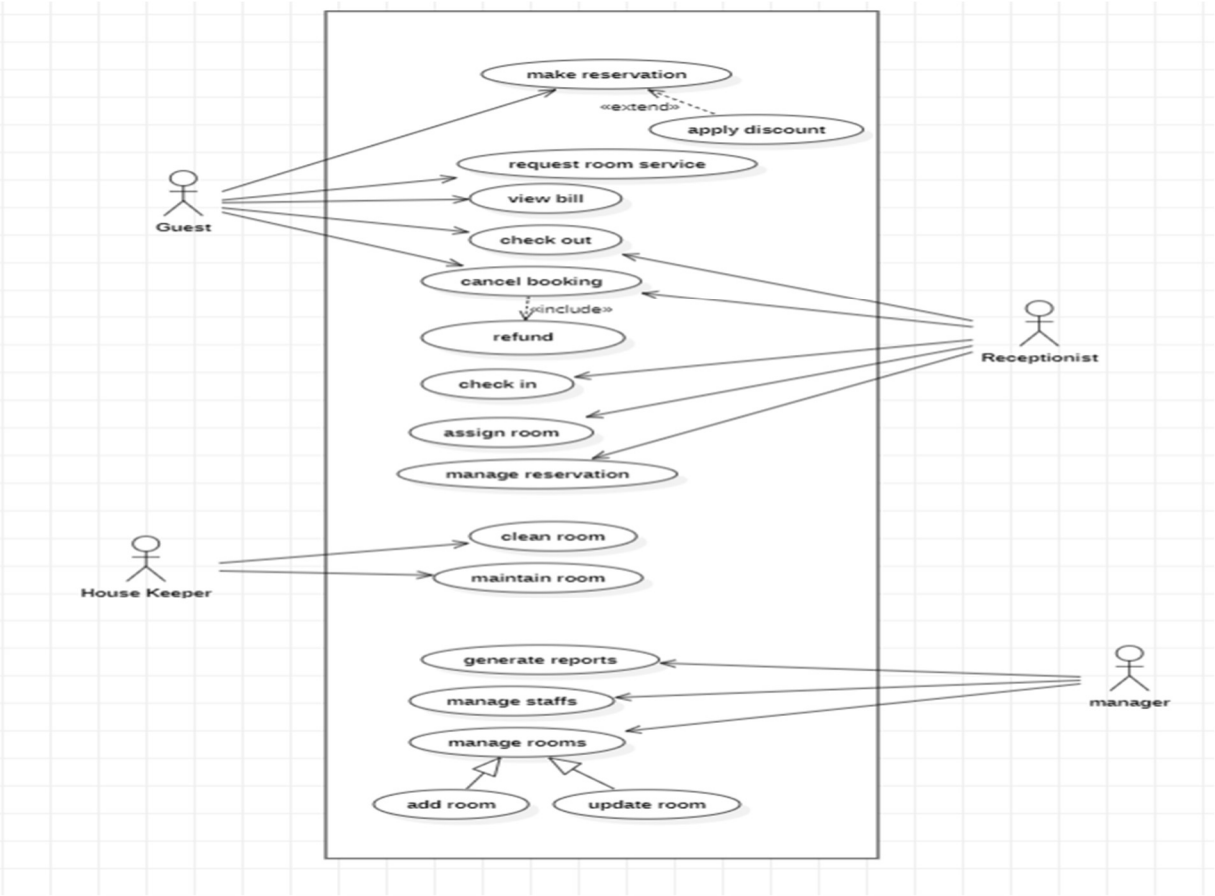
Advanced State Diagram



Description

Start: The process begins.
Verify ID: The system checks the customer's identity (could be login or guest verification).
Book Room: The customer selects a room and initiates the booking process.
Validate Details: The system validates the provided booking details (e.g., guest info, payment).
Assign Room: Once validated, the system assigns the selected room to the customer.
Update Booking: The system updates the booking status with relevant details (e.g., dates, room number).
Booking Successful: The booking is completed successfully, and confirmation is sent

Use-Case Diagram



Description

Customer:

- **Request Room Service**: The guest can request services like food, cleaning, or maintenance.
- **View Bill**: The guest can view their current charges and total bill.
- **Check Out**: The guest checks out when their stay ends.

Hotel Receptionist:

- **Assign Room**: The receptionist assigns a room to the guest based on availability.
- **Manage Reservation**: The receptionist handles booking, modifications, and cancellations of reservations.

Housekeeper:

- **Maintain Room**: The housekeeper ensures rooms are cleaned, maintained, and ready for new guests.

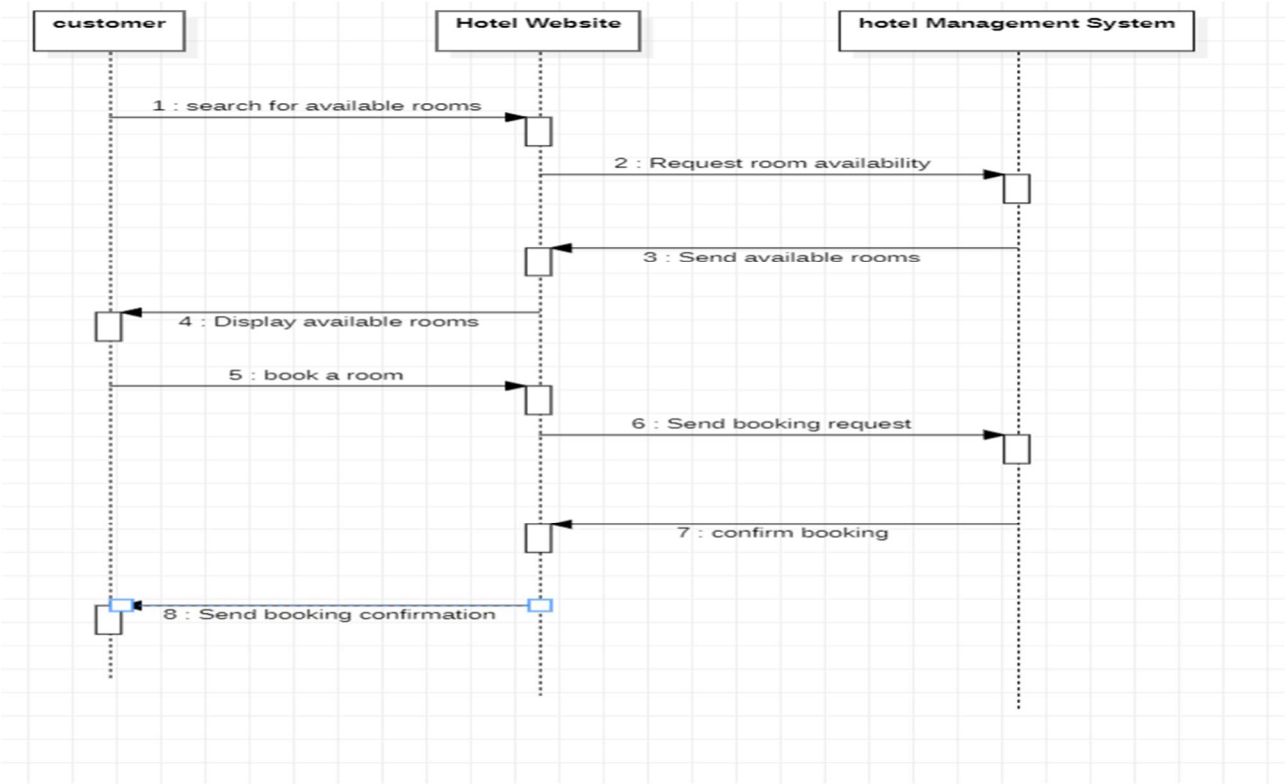
Hotel Manager:

- **Generate Reports**: The manager generates reports on occupancy, revenue, and guest feedback.
- **Manage Staff**: The manager oversees staff scheduling, roles, and performance.

Admin/Management:

- **Manage Hotel**: Admin manages hotel details like amenities, pricing, and policies.
- **Add New Staff**: Admin can add new staff members to the system.

Sequence Diagram

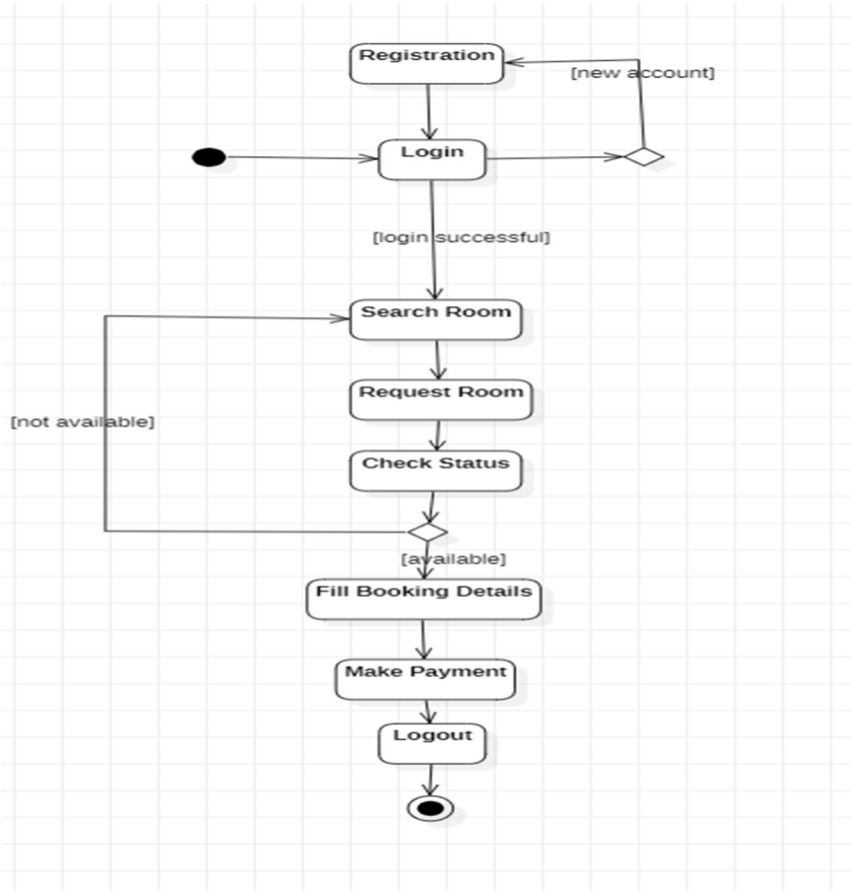


Description

Customer searches for available rooms on the **Hotel Website**.
Hotel Website sends a request to the **Hotel Management System** to check room availability.
Hotel Management System sends back a list of available rooms.
Hotel Website displays the available rooms to the customer.

Customer selects and books a room.
Hotel Website sends the booking request to the **Hotel Management System**.
Hotel Management System confirms the booking
Hotel Website sends the booking confirmation to the customer.

Activity Diagram



Description

Registration: The user registers on the platform to create an account (if not already registered).
Login: The user logs into the system using their credentials.
Search Room: The user searches for available rooms based on their preferences (dates, type, etc.).
Request Room: The system checks room availability and displays the options.
Room Available: If rooms are available, the user can proceed; otherwise, they are notified of unavailability.
Fill Booking Details: The user enters booking details (guest info, check-in/check-out dates).
Make Payment: The user proceeds to make payment for the booking.
Booking Confirmation: After successful payment, the system confirms the booking and updates the status

Advanced Activity Diagram

Description

Customer Lane: Includes actions like **Registration**, **Login**, **Search Room**, **Fill Booking Details**, and **Make Payment**.
Hotel System Lane: Includes actions like **Check Room Availability**, **Display Available Rooms**, **Process Payment**, and **Confirm Booking**.
Payment Gateway Lane: Includes the action of **Process Payment**.
Admin/Manager Lane: Can include actions like **Manage Reservations** and **Generate Reports**.

2. Credit Card Processing System

Problem Statement

The **Credit Card Processing System** is designed to securely handle and process payments made via credit cards. It enables customers to make payments, while ensuring their sensitive information (like card details) is encrypted and securely transmitted. The system validates card information, checks for sufficient funds or credit, processes the transaction, and updates both the merchant's and customer's accounts accordingly. The goal is to provide a fast, reliable, and secure method for online or point-of-sale transactions, reducing the risk of fraud and ensuring compliance with industry standards.

Software Requirement Specification(SRS)

1. Introduction

- **Purpose**: To securely process credit card transactions, ensuring reliable payments for customers and merchants while preventing fraud.
- **Scope**: Handles card validation, transaction authorization, fraud detection, and integrates with payment gateways (Visa, MasterCard).
- **Overview**: Manages end-to-end payment processing, from card entry to transaction completion.

2. General Description

- **User Characteristics**:
  - **Customer**: Makes payments via credit card.
  - **Merchant**: Accepts and verifies payments.
  - **Admin**: Manages system settings, transactions, and security.
- **Features**:
  - Secure card validation, transaction authorization, fraud detection.
  - Transaction history and reporting.
  - Integration with external gateways like Visa, MasterCard.
- **Benefits**:
  - **For Customers**: Fast, secure payments.
  - **For Merchants**: Reliable processing, fraud protection.
  - **For Both**: Real-time monitoring and transaction tracking.

3. Functional Requirements

- **Transaction Initiation**: Customers input card details for processing.
- **Card Validation**: The system validates card information.
- **Payment Authorization**: Communicates with payment gateways for transaction approval.
- **Transaction Completion**: Updates customer and merchant accounts after approval.

4. Interface Requirements

- User-friendly interface for customers to input card details and view transaction history.
- Merchant dashboard to manage payments and generate reports.

5. Performance Requirements

- The system should handle a minimum of 100 transactions per second.
- Response time for transaction authorization should be under 2 seconds.

6. Design Constraints

- Must comply with **PCI DSS** security standards for data protection.
- Integration with third-party gateways like Visa, MasterCard, etc.

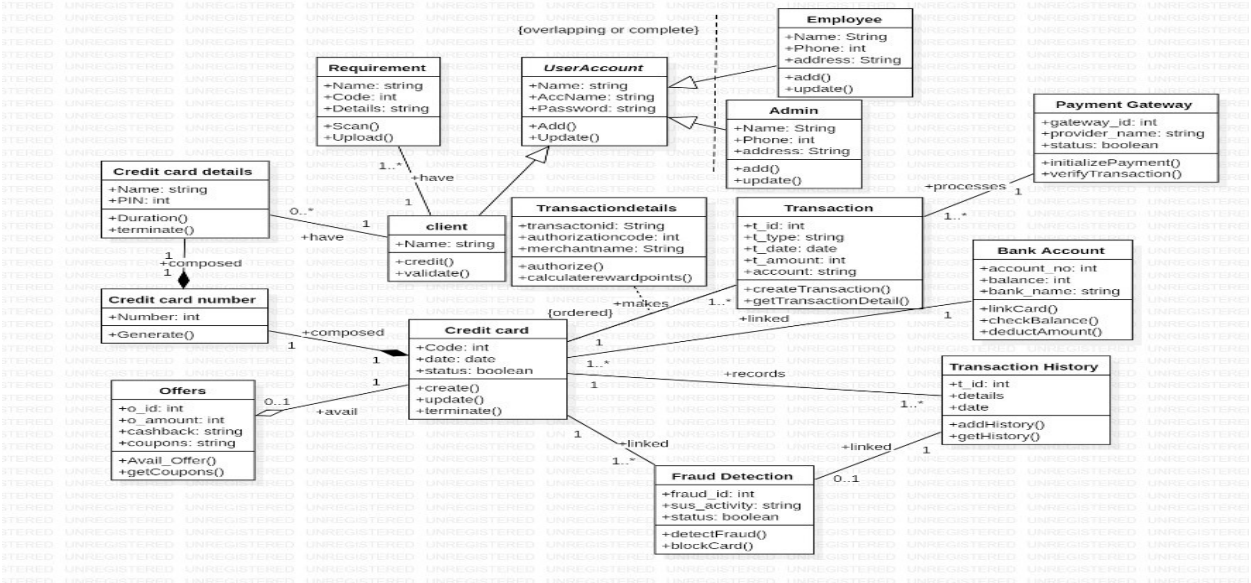
7. Non-Functional Requirements

- **Security**: Encrypt sensitive data during transactions.
- **Scalability**: The system must scale to handle increasing transaction volumes.
- **Availability**: 99.9% uptime for continuous payment processing.

8. Preliminary Budget and Time

- **Budget**: Estimated development cost: $200,000.
- **Timeline**: System design, development, and deployment within 6 months.

Class Diagram



Description

CreditCardDetails:

- Attributes: Name, PIN, Number, Duration, Cashback, Coupons.
- Methods: Generate(), Terminate().
- Represents the user's credit card information.

Client:

- Attributes: Name.
- Methods: validate().
- Represents a customer who uses the card.

UserAccount:

- Attributes: Name, AccName, Password, Address.
- Methods: Update().
- Manages user login and personal details.

TransactionDetails:

- Attributes: TransactionID, MerchantName.
- Methods: authorize(), CalculateRewardPoints().
- Handles transaction info and reward calculations.

Employee:

- Attributes: Name, Phone, Address.
- Methods: Update().
- Manages system operations.

Admin:

- Attributes: Name, Phone, Address.
- Methods: add().
- Manages users and employees.

Transaction:

- Attributes: T_ID, T_Type, T_Date, T_Amount.
- Methods: update().
- Tracks payment transactions.

PaymentGateway:

- Attributes: ProviderName, Status.
- Represents third-party payment processors.

BankAccount:

- Attributes: Balance, BankName.
- Methods: linkCard(), checkBalance().
- Represents a linked bank account for payments.

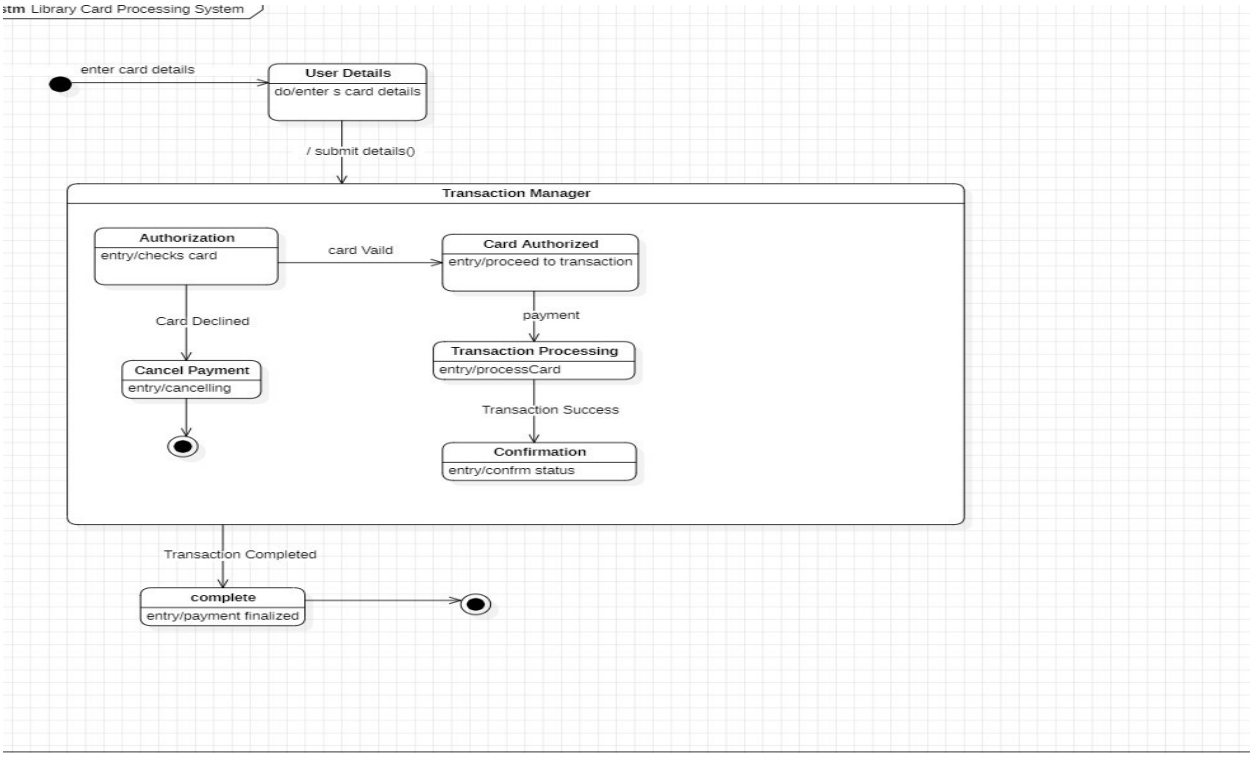
TransactionHistory:

- Attributes: ID.
- Stores transaction records.

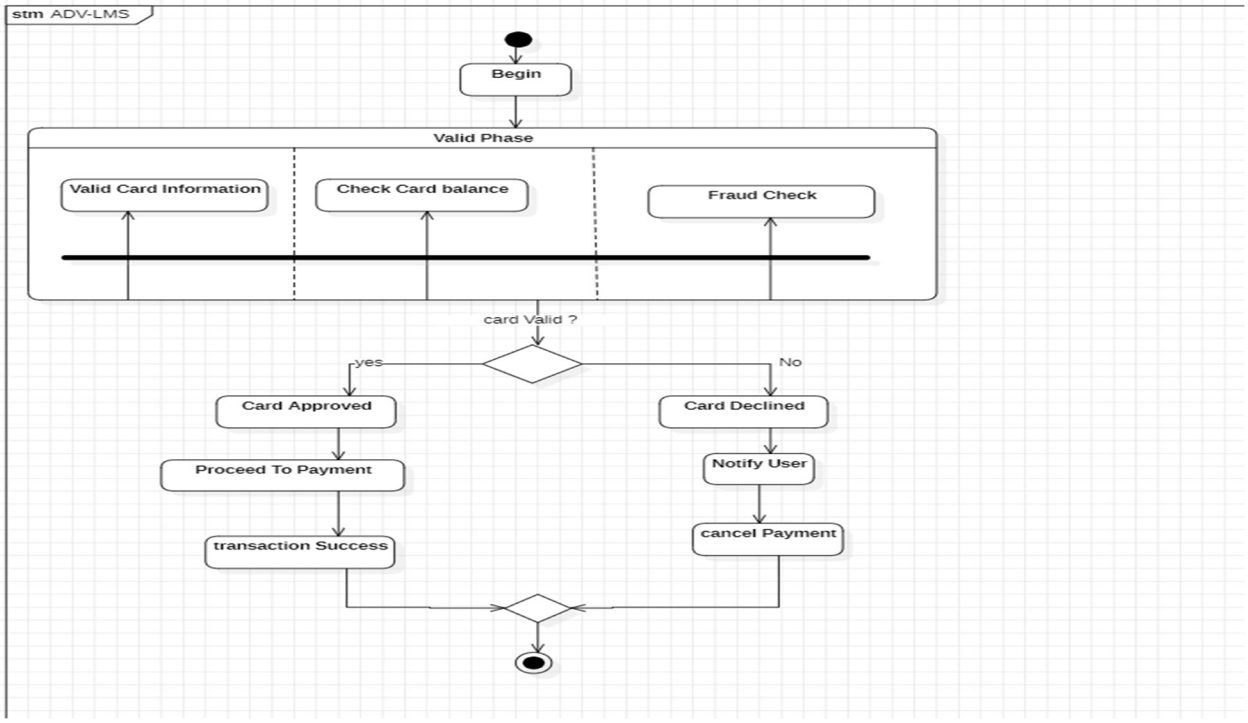
FraudDetection:

- Attributes: FraudID, Status.
- Methods: detectFraud(), blockCard().
- Detects fraud and blocks suspicious cards.

State Diagram



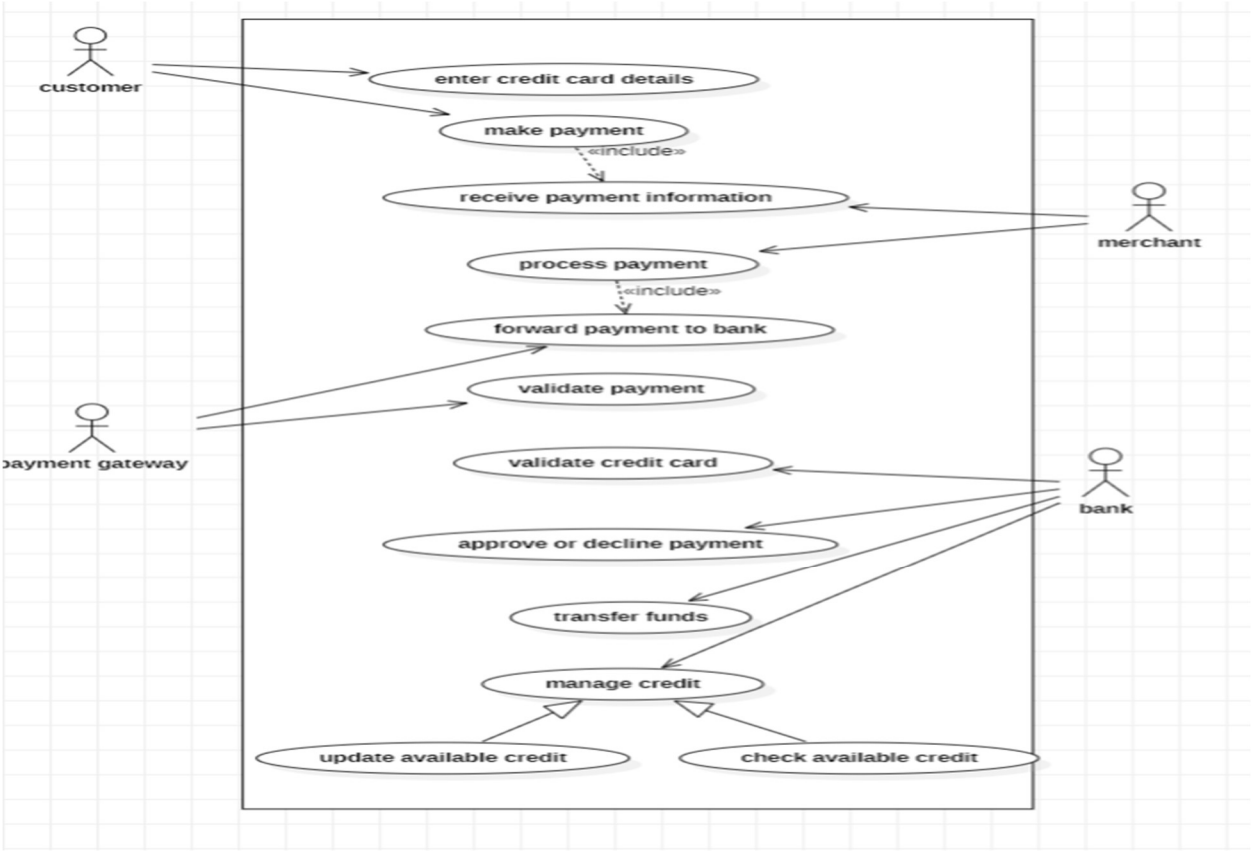
Advanced State Diagram



Description

Begin: The payment process is initiated by the user.
Valid Card Information: The system checks if the card details are valid.
Valid Phase: If valid, the system moves to the next phase to check balance and fraud.
Check Card Balance: Verifies if the card has sufficient funds.
Card Approved: If the balance is sufficient, the card is approved for payment.
Fraud Check: The system checks for fraudulent activity.
Transaction Success: If no fraud is detected, the transaction is successfully completed.
Notify User: The user is notified of the payment status.
Cancel Payment: If issues arise (e.g., insufficient funds or fraud), the payment is canceled .

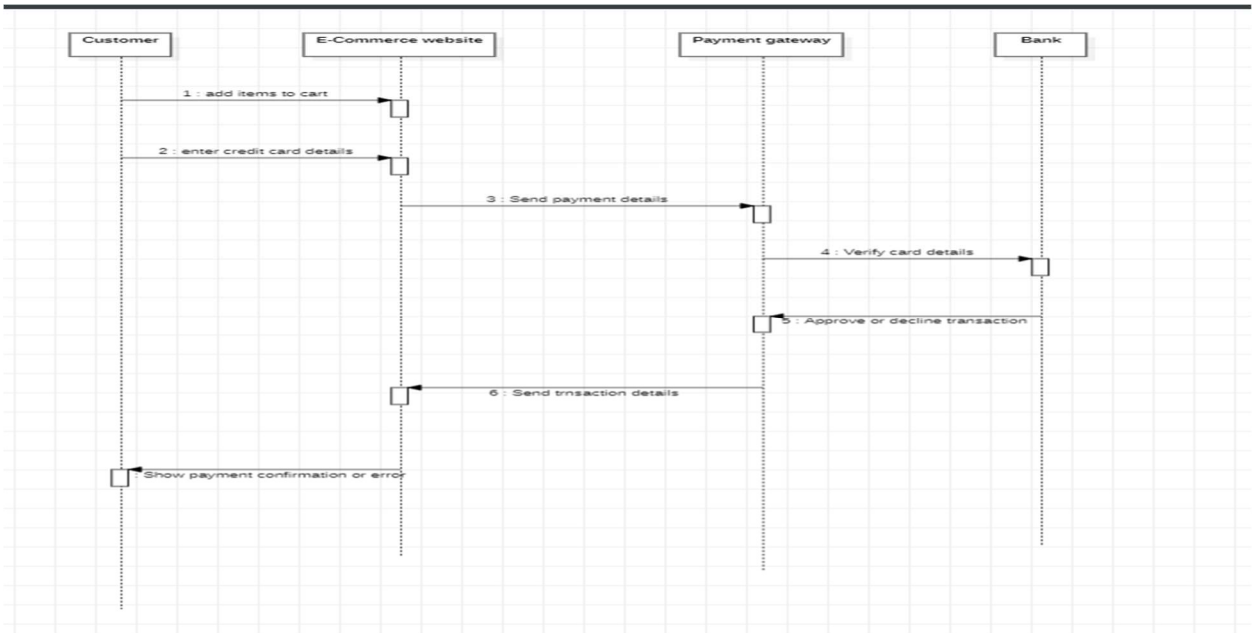
Use-Case Diagram



Description

Enter Credit Card Details: The **Customer** enters their credit card information for payment.
Make Payment: The **Customer** initiates the payment process.
Receive Payment Information: The **System** receives the payment details from the customer.
Forward Payment to Bank: The **System** sends payment information to the **Bank** for validation.
Validate Payment: The **Bank** validates the payment information.
Validate Credit Card: The **Bank** checks if the credit card is valid.
Approve or Decline: The **Bank** either approves or declines the transaction based on the validation.
Transfer Funds: If approved, the **Bank** transfers the funds to the merchant's account.
Manage Available Credit: The **Bank** updates the available credit on the customer's card.
Check Available Credit: The **System** checks the customer's available credit during the validation process.

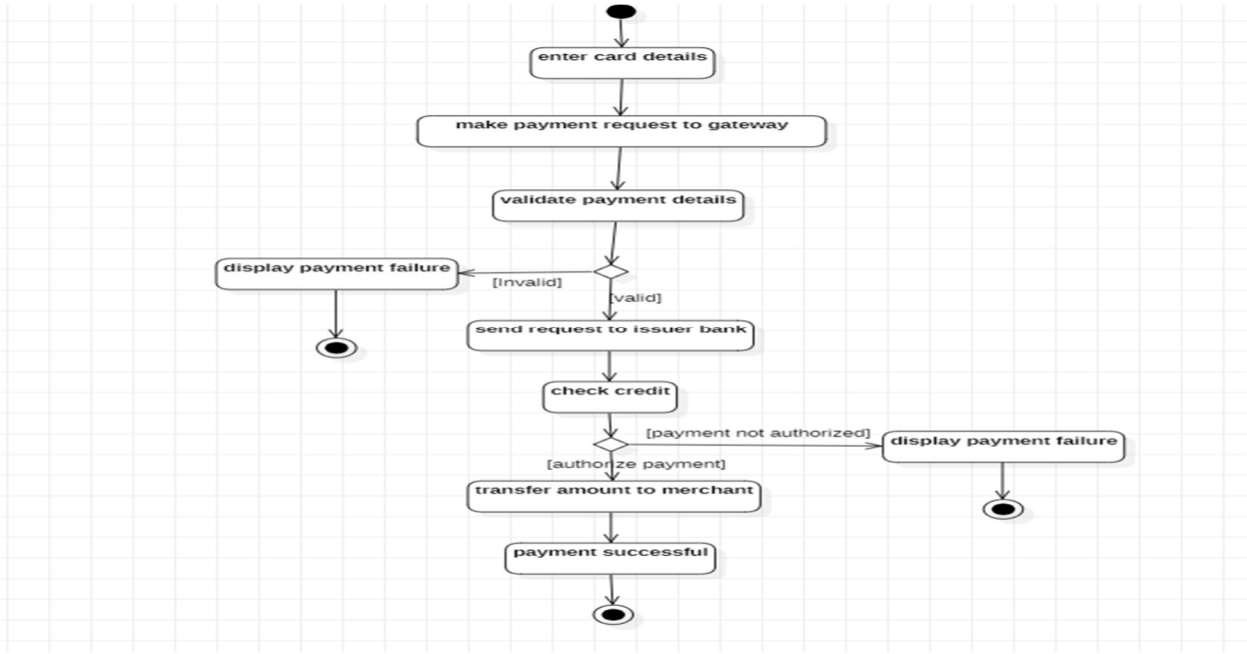
Sequence Diagram



Description

Add to Cart: The **Customer** adds items to their cart for purchase.
Enter Credit Card Details: The **Customer** enters their credit card information for payment.
Show or Send Payment Details: The **System** displays or sends the entered payment details to the **Payment Gateway**.
Verify Details: The **Payment Gateway** verifies the credit card details and checks for sufficient funds.
Approve or Decline Transaction: The **Payment Gateway** either approves or declines the transaction based on the verification.
Send Details: The **System** sends the final transaction details (approval or decline) back to the **Customer**.

Activity Diagram



Description

Enter Card Details: The **Customer** inputs their credit card information.
Make Payment Request to Gateway: The **System** sends the payment details to the **Payment Gateway** for processing.
Validate Payment Details: The **Gateway** validates the card information (card number, expiration date, CVV).

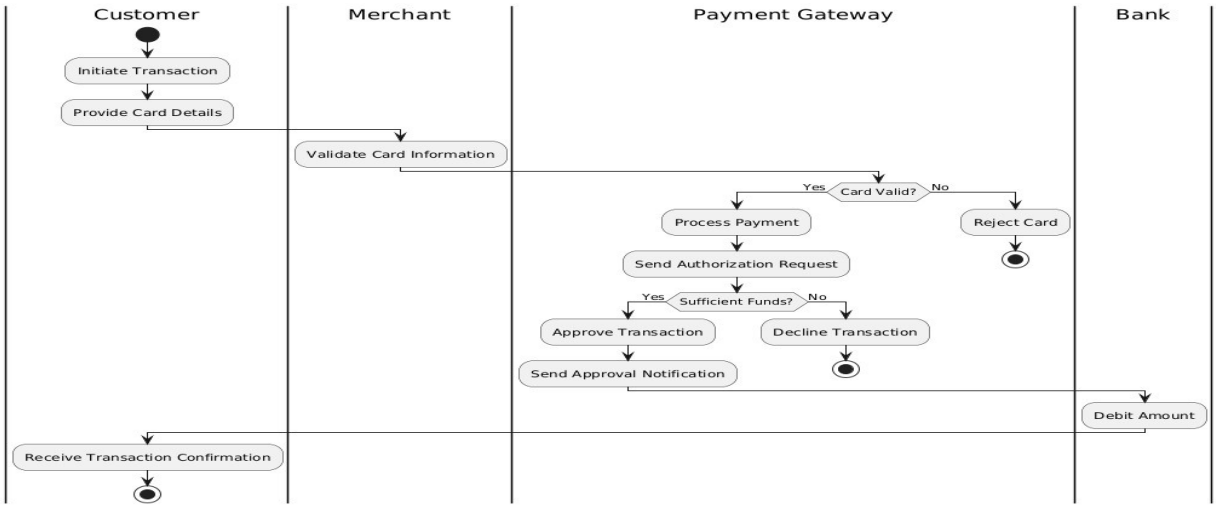
- **If Invalid**: Display **Payment Failure** with an "Invalid Card" message.

Send Request to Issuer Bank: If the card details are valid, the **Payment Gateway** sends the payment request to the **Issuer Bank**.
Check Credit: The **Issuer Bank** checks if the customer has sufficient funds or credit.

- **If Payment Not Authorized**: Display **Payment Failure** (e.g., insufficient funds or fraud detected).
- **If Authorized**: Proceed to payment transfer.

Transfer Amount to Merchant: Once authorized, the **Issuer Bank** transfers the payment to the **Merchant**.
Payment Successful: The **System** notifies the **Customer** of successful payment completion.

Advanced Activity Diagram



Description

Customer Lane:

- **Enter Card Details**: The customer inputs their credit card information.
- **Request Payment**: The customer submits the payment request.

Payment Gateway Lane:

- **Validate Card Information**: The payment gateway checks if the card details are correct.
- **Send Request to Issuer**: If the card is valid, the gateway forwards the payment request to the issuer bank.

Issuer Bank Lane:

- **Check Credit**: The bank verifies the availability of funds or credit for the transaction.
- **Authorize Payment**: If the funds are available, the bank approves the payment.
- **Decline Payment**: If there are issues (e.g., insufficient funds, fraud), the bank declines the transaction.

Merchant Lane:

- **Receive Payment**: The merchant receives the payment confirmation (if successful).
- **Process Transaction**: The merchant processes the transaction and completes the sale.

System Lane:

- **Display Success or Failure**: The system notifies the customer whether the payment was successful or failed.

3. Library management System

Problem Statement

The current manual system of managing library operations is inefficient and prone to errors. It involves manual tracking of books, members, transactions, and overdue items, leading to delays and mismanagement. There is no centralized system to automate processes such as book search, issuing, return, reservation, and fine calculation. The lack of real-time data and reporting hampers effective decision-making and overall library management. A **Library Management System (LMS)** is needed to streamline these operations, automate tasks, improve accuracy, and enhance user experience for both library staff and members.

Software Requirement Specification(SRS)

1.Introduction

- **Purpose**: The purpose of this document is to define the software requirements for a Library Management System (LMS) that will automate book management, user interactions, and administrative tasks in a library environment.
- **Scope**: The LMS will handle user registration, book search, issue/return management, fine calculation, and reporting. It will be used by library staff and members for efficient management and access to library resources.
- **Overview**: This system will help reduce manual work, streamline library operations, and provide real-time information to both staff and users, ensuring accurate tracking and better service.

2. General Description

- **User Characteristics**:
  - **Library Staff**: Manages books, tracks issues and returns, and handles user management.
  - **Library Members**: Browse books, borrow/return items, and pay fines.
  - **Admin**: Manages system settings, reports, and configurations.
- **Features**:
  - Book Search and Categorization
  - User Registration and Management
  - Book Issue and Return Management
  - Fine Calculation
  - Reservation and Book Hold
  - Transaction History and Reporting
- **Benefits**:
  - Automates manual tasks, saving time.
  - Provides accurate and up-to-date information.
  - Reduces errors in book management and fine calculations.
  - Improves user satisfaction with faster access to library resources.

3. Functional Requirements

- **User Registration**: Members can register with the system by providing their details.
- **Book Search**: Users and staff can search for books by title, author, or category.
- **Book Issue/Return**: Staff can issue or return books to/from members.
- **Fine Management**: The system calculates overdue fines automatically.
- **Reservation**: Members can reserve books that are currently unavailable.
- **Transaction History**: Both staff and members can view borrowing history and fine status.

4. Interface Requirements

- **User Interface**:
  - Simple, easy-to-navigate interface for both staff and members.
  - Book search functionality, issue/return forms, and fine details.
- **Admin Interface**:
  - Dashboard for managing users, books, and reports.
  - Access to system settings, book inventory, and overdue reports.

5. Performance Requirements

- The system must handle at least 100 concurrent users.
- Response time for searching books should not exceed 3 seconds.
- System should handle up to 1000 transactions per day without performance degradation.

6. Design Constraints

- **Technology**: The system will be built using web-based technologies (e.g., HTML, CSS, JavaScript) and a relational database (e.g., MySQL).
- **Integration**: The system must be capable of integrating with external tools for library cataloging if required.
- **Security**: User data, especially personal information and transaction history, must be encrypted.

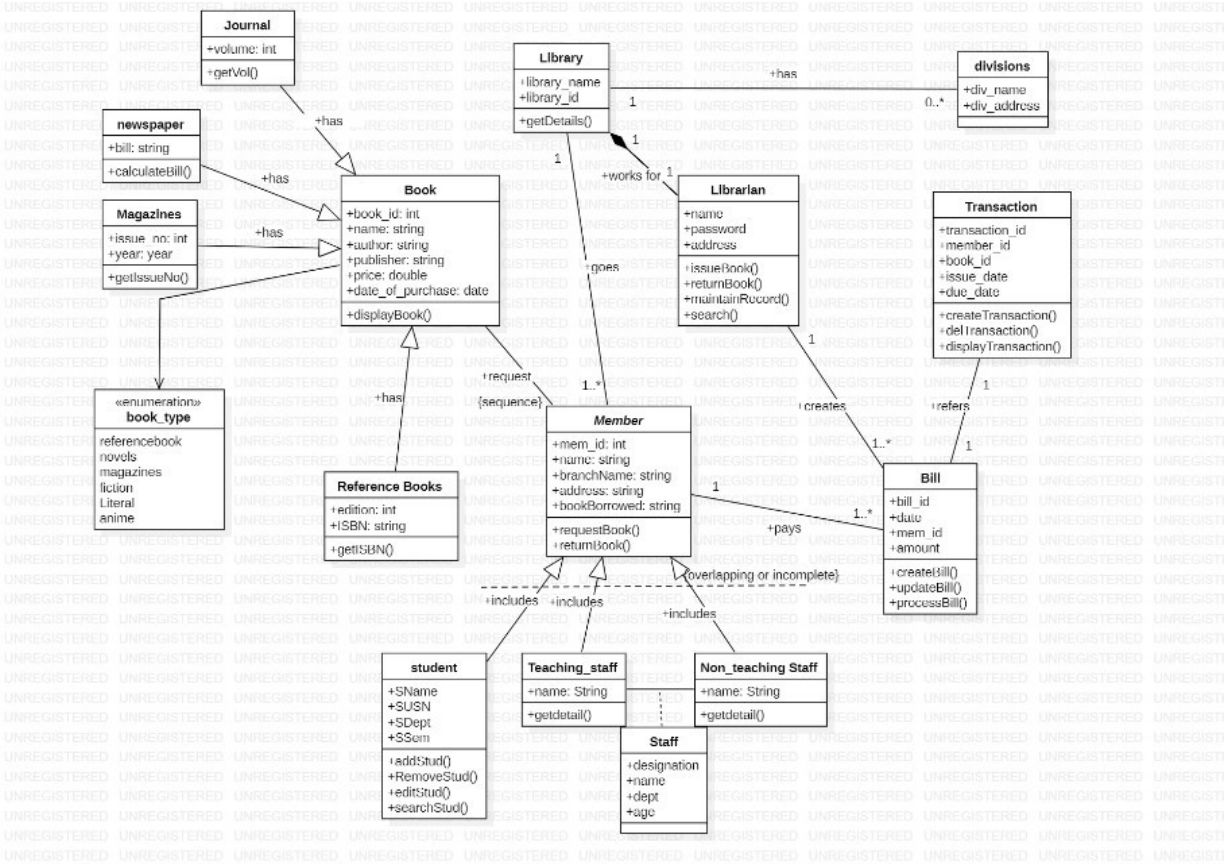
7. Non-Functional Requirements

- **Reliability**: The system should be available 99.9% of the time.
- **Scalability**: It should support future growth, such as more books, users, and transactions.
- **Usability**: The interface should be user-friendly and intuitive for all types of users (staff, members, admins).
- **Security**: The system must ensure secure access with user authentication and authorization.

8. Preliminary Budget and Time

- **Budget**: Estimated development cost is around $50,000.
- **Timeline**: The development and deployment of the system should be completed within 6 months.

Class Diagram



Description

Library

- **Attributes**: library_id
- **Methods**: calculateBill()
- Represents the library and manages bill calculations.

Book

- **Attributes**: ISBN, title, publisher, price, year, book_type
- **Methods**: displayBook()
- Represents a book with details like ISBN, title, and price.

Member

- **Attributes**: id, name, student_number, department
- **Methods**: searchBook()
- Represents a library member who can search for and borrow books.

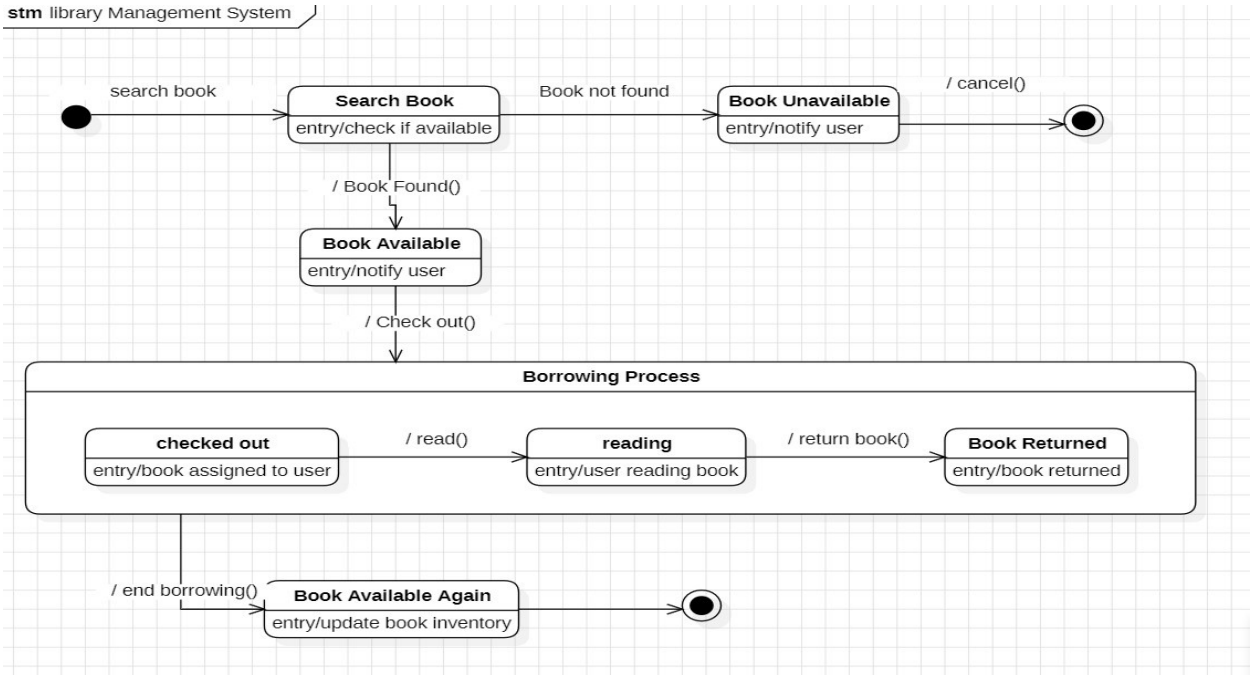
Staff

- **Attributes**: name, department, branchName
- **Methods**: returnBook(), search()
- Represents library staff managing book issues and returns.

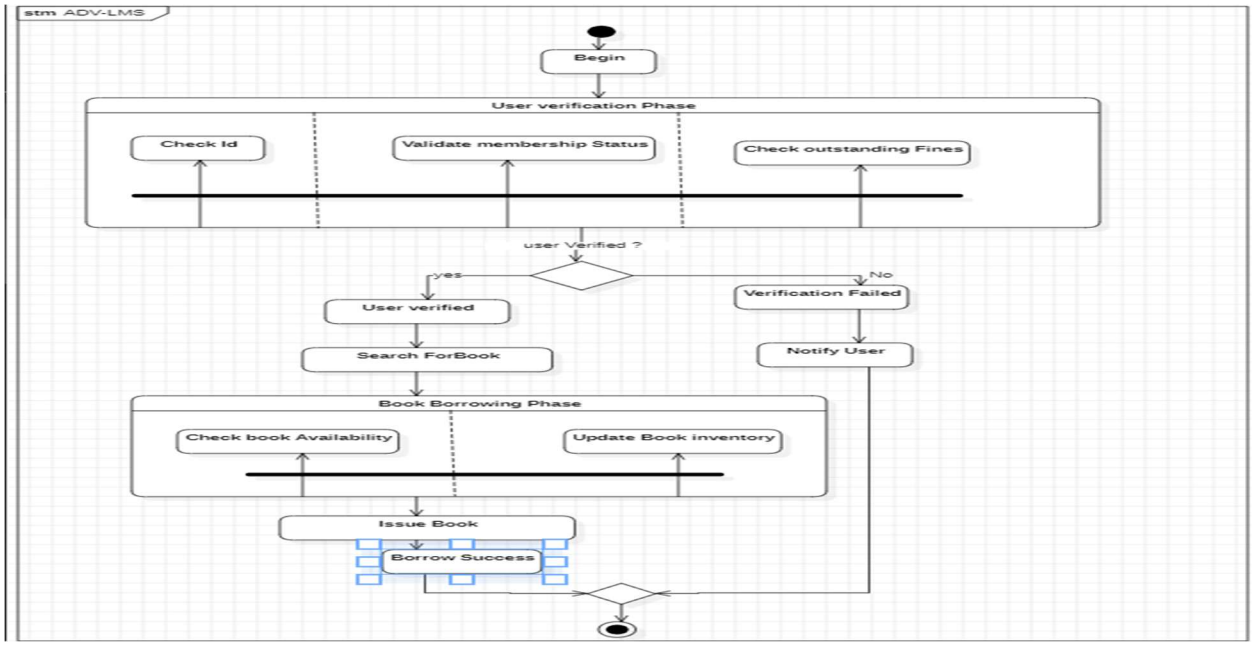
Transaction

- **Attributes**: id, book_id, date
- **Methods**: updateBill()
- Represents a borrowing transaction and updates billing.

State Diagram



Advanced State Diagram



Description

Start: The process begins when a user interacts with the system (e.g., attempting to borrow or return a book).

Validate Card/Member Info: The system validates the user's membership details (e.g., ID or card) to ensure they are eligible to borrow books.

Check Book Availability: The system checks if the requested book is available for borrowing.

- **If Book Available**: Move to the "Book Issued" state.
- **If Book Not Available**: Notify the user that the book is unavailable.

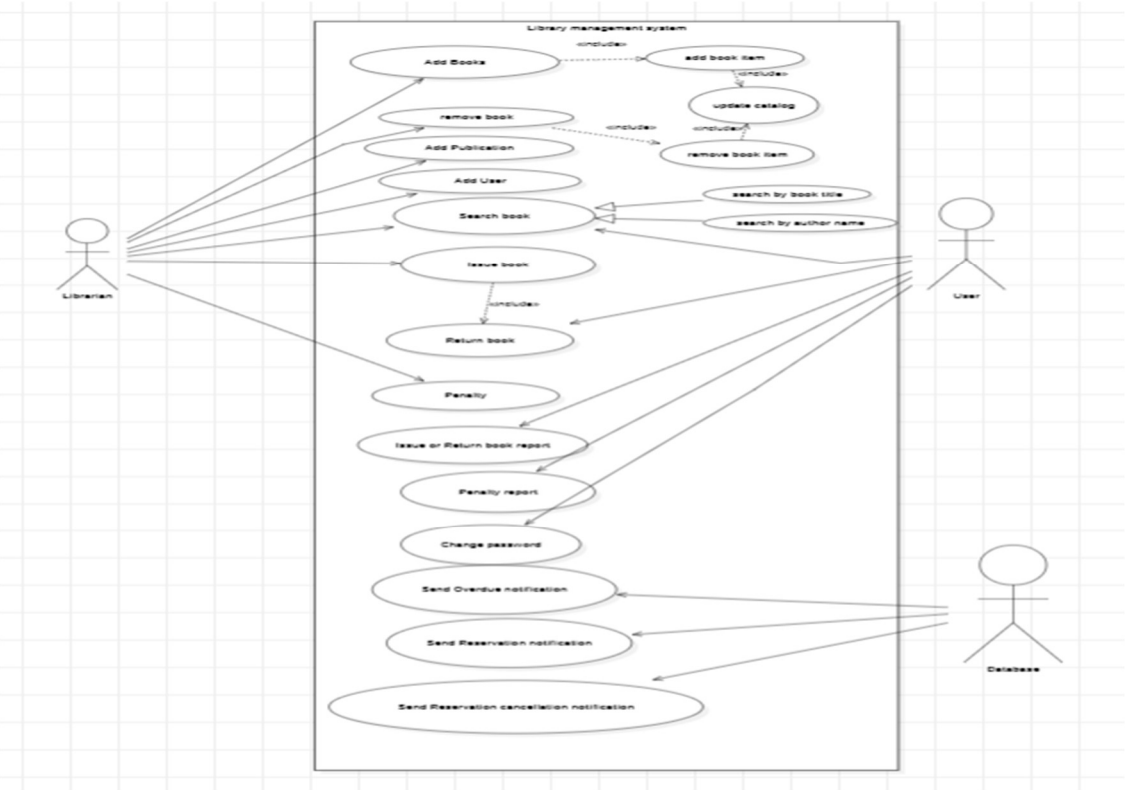
Borrowing: If the book is available, the system processes the borrowing request and issues the book to the user.

Return Book: After borrowing, the user can return the book within the specified time.

Update Book Status: Once the book is returned, the system updates the status of the book (available for others).

End: The process ends after the book is returned, and the transaction is complete.

Use-Case Diagram



Description

Library Member:

- **Search for Books**: The member searches for books based on title, author, or genre.
- **Borrow Books**: The member can borrow available books and view their due dates.
- **Return Books**: The member returns borrowed books to the library.
- **Pay Fines**: The member pays fines for overdue books.

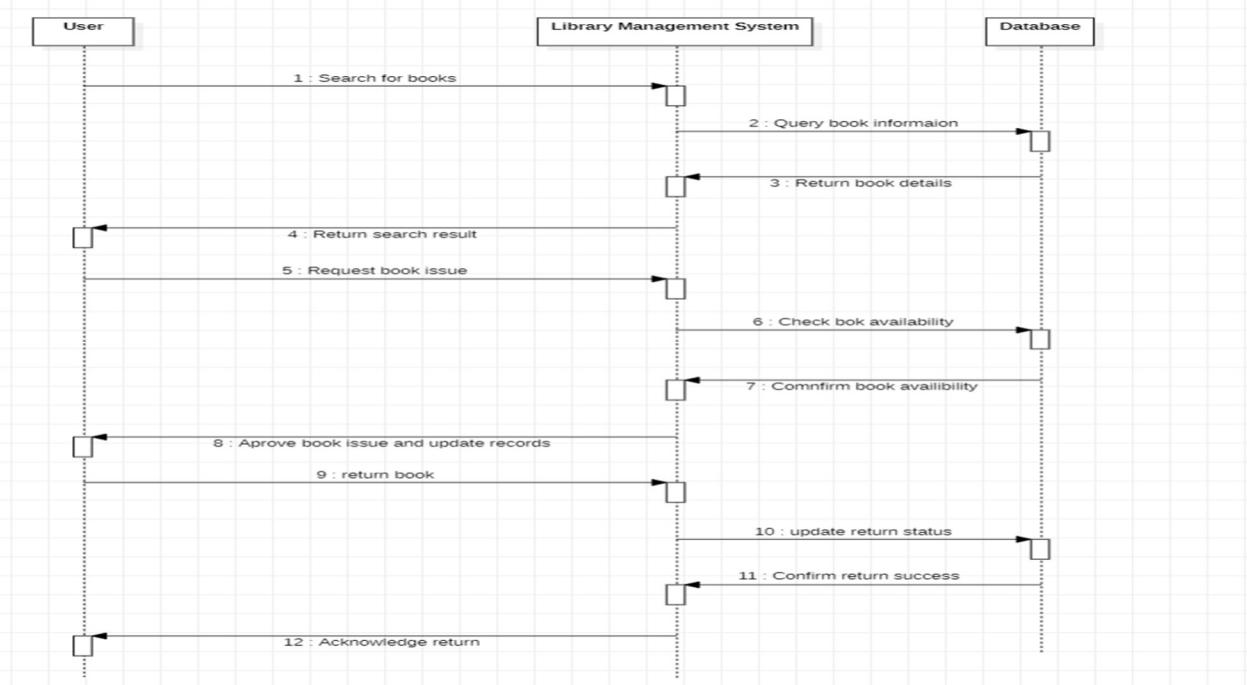
Library Staff:

- **Manage Books**: Staff adds, updates, or removes books from the library catalog.
- **Issue Books**: Staff issues books to members, recording borrow transactions.
- **Return Books**: Staff processes returned books and updates their status.
- **Generate Reports**: Staff generates reports on issued books, fines, and overdue items.

Admin:

- **Manage Users**: Admin registers, updates, or removes library members and staff.
- **View System Logs**: Admin monitors system activity and user interactions.

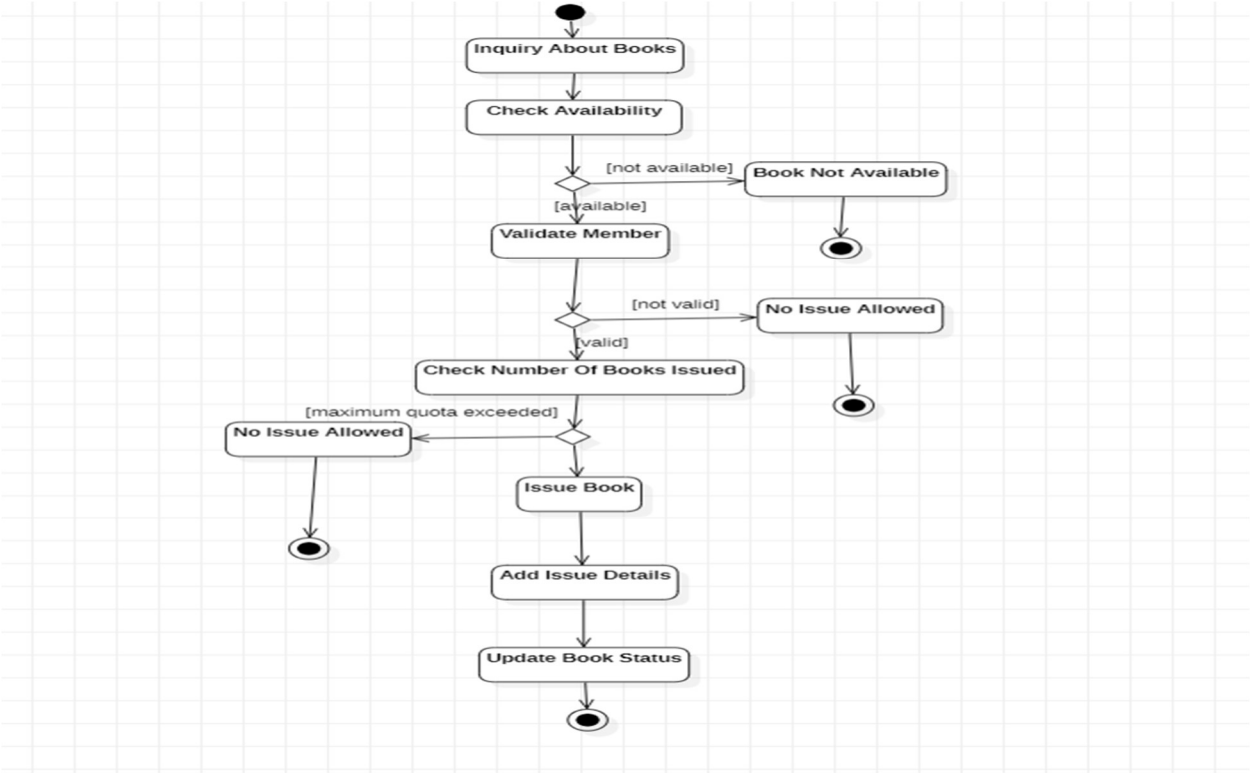
Sequence Diagram



Description

Search for Books: **Member** searches for books.
Query Book Info: **System** retrieves book details.
Return Search Results: **System** shows matching books.
Request Issue: **Member** requests to borrow a book.
Check Availability: **System** checks if the book is available.
Confirm Availability: **System** confirms book availability.
Approve and Update Records: **Staff** approves issue, updates records.
Return Book: **Member** returns borrowed book.
Acknowledge Return: **System** acknowledges book return.
Update Status: **System** updates book status to "available".
Confirm Return Success: **System** confirms successful return.

Activity Diagram



Description

Inquiry About Book: The **Member** inquires about the availability of a book.
Check Availability: The **System** checks if the book is available.

- **If Not Available**: The process ends with a "Book Not Available" message.

Validate Member: The **System** verifies if the **Member**'s account is valid.

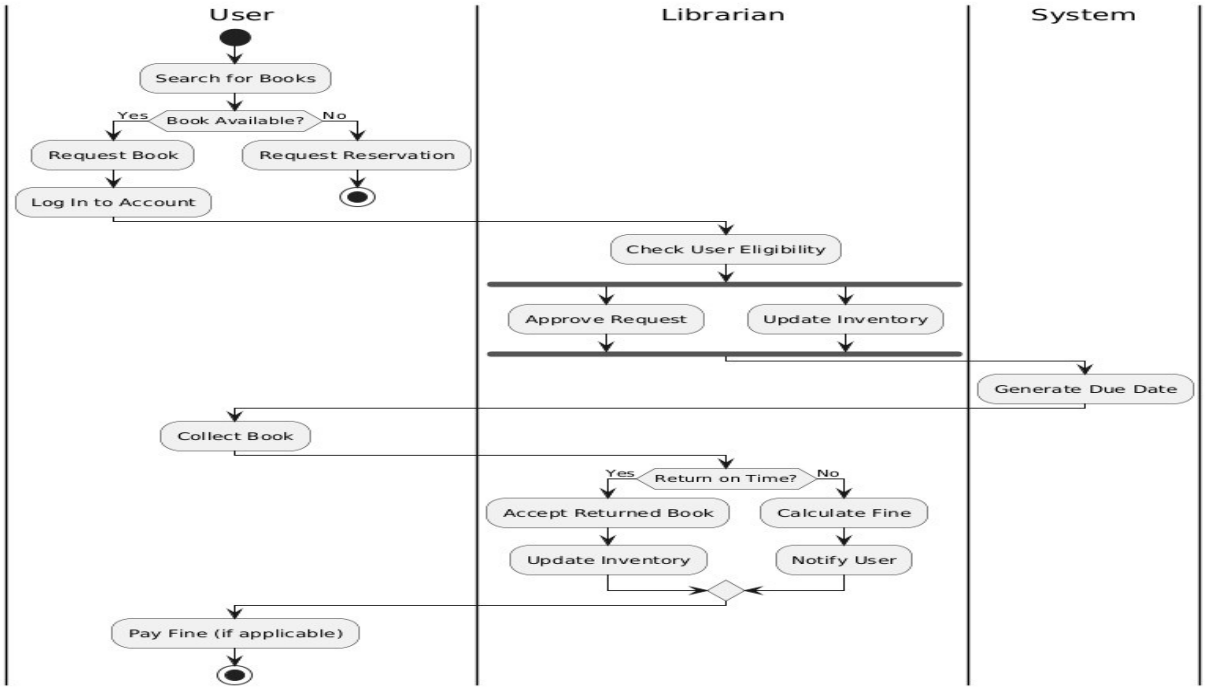
- **If Invalid**: Display "No Issue Allowed" and end the process.

Check Number of Issued Books: The **System** checks the number of books the **Member** has issued.

- **If Maximum Quota Exceeded**: Display "Quota Exceeded" and end the process.

Issue Book: If the **Member** is eligible, the **Staff** issues the book.
Add Issue Details: The **System** records the book details and member transaction.
Update Status: The **System** updates the book's status to "Issued" and completes the process.

Advanced Activity Diagram



Description

Member Lane

- **Inquiry About Book**: The member initiates the process by checking if the book is available.

System Lane

- **Check Availability**: The system checks if the book is available.
  - If **Not Available**, the system notifies the member and ends the process.
- **Validate Member**: The system validates if the member's account is active.
  - If **Invalid**, the system shows "No Issue Allowed" and ends the process.
- **Check Number of Issued Books**: The system checks if the member has exceeded their quota of issued books.
  - If **Quota Exceeded**, the system displays "Quota Exceeded" and ends the process.

Staff Lane

- **Issue Book**: If the member is valid and within their book limit, the staff proceeds with issuing the book.
- **Add Issue Details**: The staff adds transaction details, such as the book ID and issue date.

4. Stock Management System

Problem Statement

The **Stock Management System** aims to address the challenges of tracking and managing inventory in businesses. The problem lies in inefficient manual tracking, leading to errors in stock levels, missed orders, and slow decision-making. This system will automate the process of managing stock, tracking inventory levels, and ensuring timely reordering of products to prevent shortages or overstocking. It will improve accuracy, reduce operational costs, and provide real-time data to enhance inventory control and optimize supply chain management.

Software Requirement Specification(SRS)

1. Introduction

- **Purpose**: Automates stock tracking, updates, and management to prevent stockouts/overstocking.
- **Scope**: Manages inventory, orders, and generates reports for businesses across multiple locations.
- **Overview**: Tracks stock, processes orders, generates reports, and integrates with external systems (e.g., accounting).

2. General Description

- **Users**:
  - **Warehouse Managers**: Update stock levels.
  - **Sales/Purchasing Staff**: Place orders.
  - **Admins**: Manage system settings.
- **Features**: Real-time stock tracking, order processing, low-stock alerts, and reports.
- **Benefits**: Accurate inventory, reduced errors, streamlined processes, better decision-making.

3. Functional Requirements

- Track and update stock levels.
- Generate stock and sales reports.
- Send low-stock alerts and reorder suggestions.

4. Interface Requirements

- Simple UI for managing stock and viewing reports.
- Integrate with external systems (ERP/accounting).

5. Performance Requirements

- Handle up to 1,000 SKUs and 100 users simultaneously.
- Real-time stock updates.

6. Design Constraints

- Scalable and compatible with web browsers and Windows.

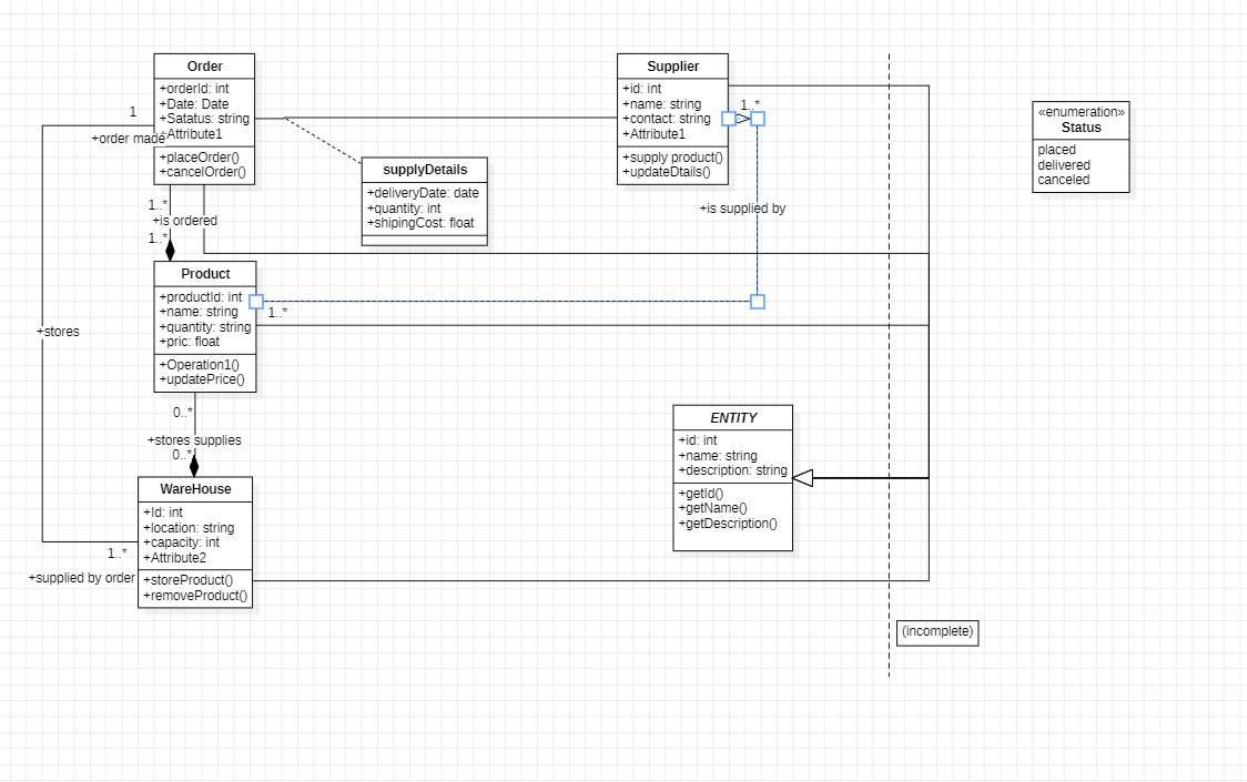
7. Non-Functional Requirements

- **Usability**: Easy for non-technical users.
- **Reliability**: 99% uptime.
- **Security**: Role-based access control.

8. Preliminary Budget and Time

- **Budget**: $20,000.
- **Timeline**: 6 months.

Class Diagram



Description

Order: Represents an order placed by a customer. It includes attributes such as orderId, status, and methods like placeOrder() and cancelOrder() to manage order statuses.

- **Attributes**: orderId, status, orderDate
- **Methods**: placeOrder(), cancelOrder(), updateStatus()

Product: Represents individual items in the order, detailing product information such as productId, name, quantity, and price.

- **Attributes**: productId, name, quantity, price
- **Methods**: updatePrice()

Warehouse: Stores the product inventory with details about its location and capacity. The warehouse manages product storage and retrieval.

- **Attributes**: id, location, capacity
- **Methods**: storeProduct(), removeProduct()

Supplier: Represents suppliers who provide products to the system. It contains details like supplierId, name, and contact.

- **Attributes**: id, name, contact
- **Methods**: supplyProduct(), updateDetails()

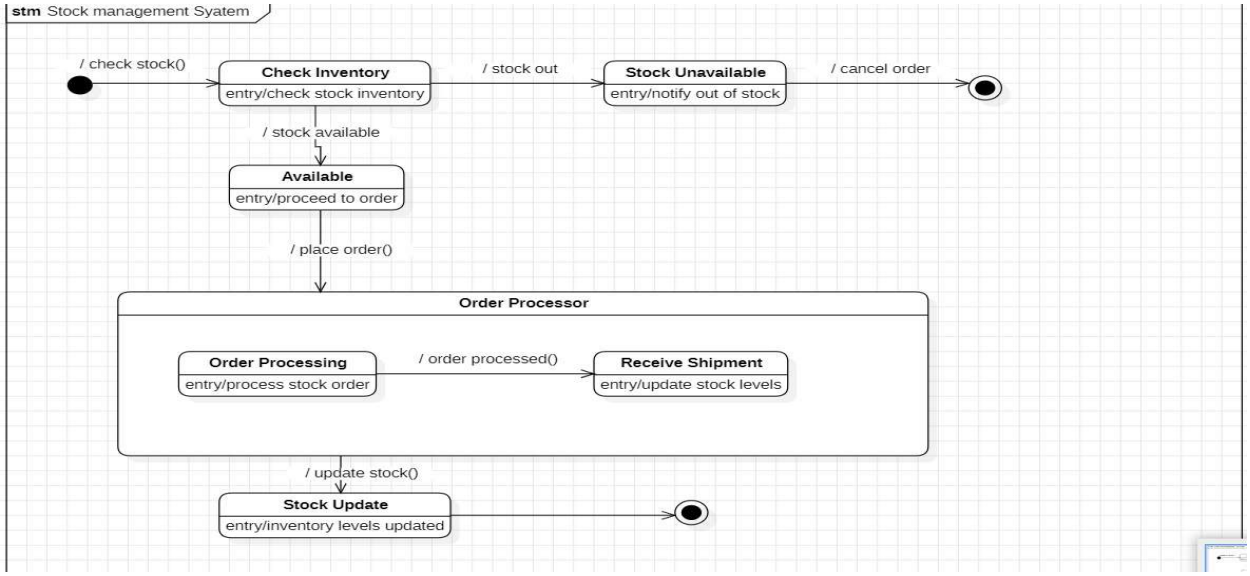
SupplyDetails: Represents the supply of products from a supplier, including delivery details and cost.

- **Attributes**: deliveryDate, quantity, shippingCost
- **Methods**: None listed

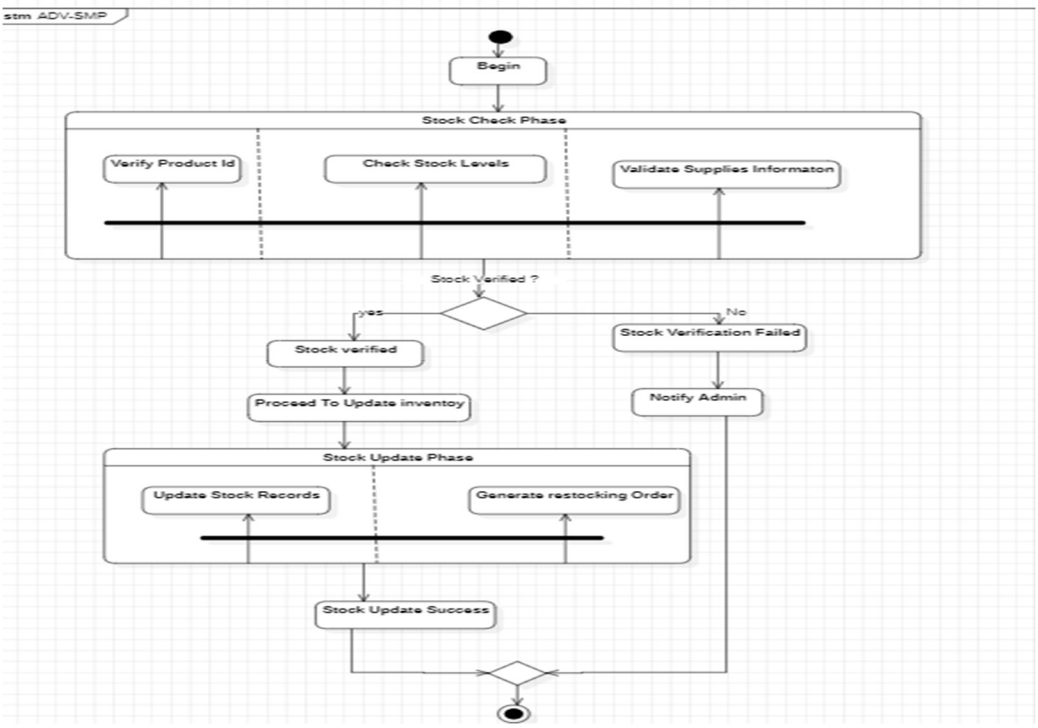
Status (Enumeration): Defines the possible states of an order, such as Placed, Delivered, or Canceled.

- **Values**: Placed, Delivered, Canceled

State Diagram



Advanced State Diagram



Description

Check Stock: The process starts by checking the current stock inventory. If stock is available, it moves to the "Proceed to Order" state.

- **Stock Available**: If stock is available, the system proceeds to the order placement.
- **Stock Unavailable**: If stock is unavailable, the system notifies that the item is out of stock.

Place Order: If stock is available, the **Order Processor** places the order.

- **Notify Out of Stock**: If the item is unavailable, the system notifies the user that the product is out of stock and cancels the order.

Receive Shipment: When new stock arrives, the system updates the stock levels.

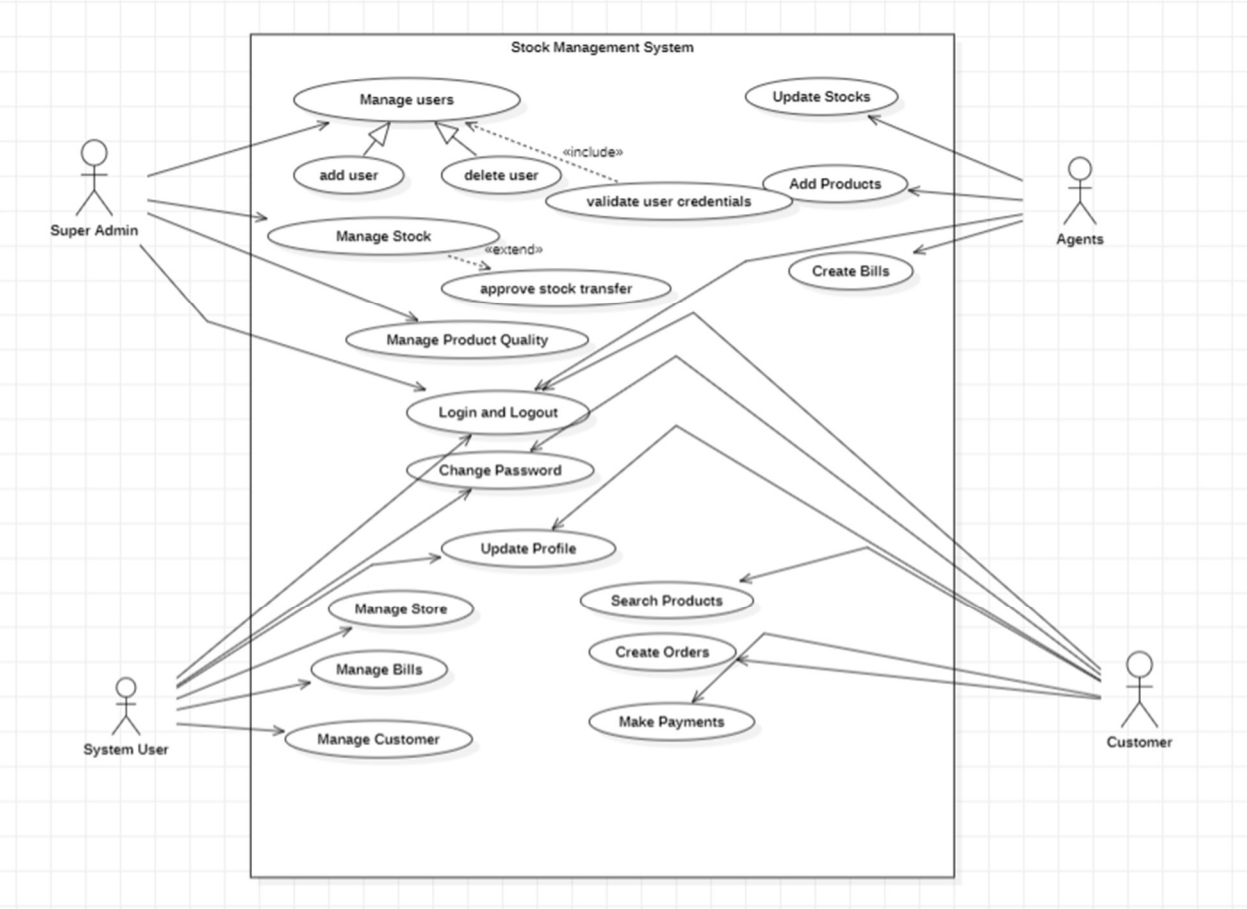
- **Update Stock Levels**: The system updates the inventory with the new stock, ensuring accurate levels.

Order Processing: Once the stock is confirmed, the system processes the order.

- **Order Processed**: The order is successfully processed and recorded in the system.

Stock Update: The system updates inventory levels after the order is placed or a shipment is received.

Use-Case Diagram



Description

Super Admin:

- **Create User**: The Super Admin can create new user accounts for other system users.
- **Manage Stock**: Oversee stock levels, updates, and product availability.
- **Approve Stock Transfer**: Approve requests for transferring stock between locations or warehouses.
- **Manage Quality**: Monitor and enforce product quality control measures.
- **Change Password**: Super Admin can update their password for system security.
- **Update Profile**: Edit personal or organizational details.

System:

- **Validate User Credentials**: Ensures users are authenticated before accessing the system.

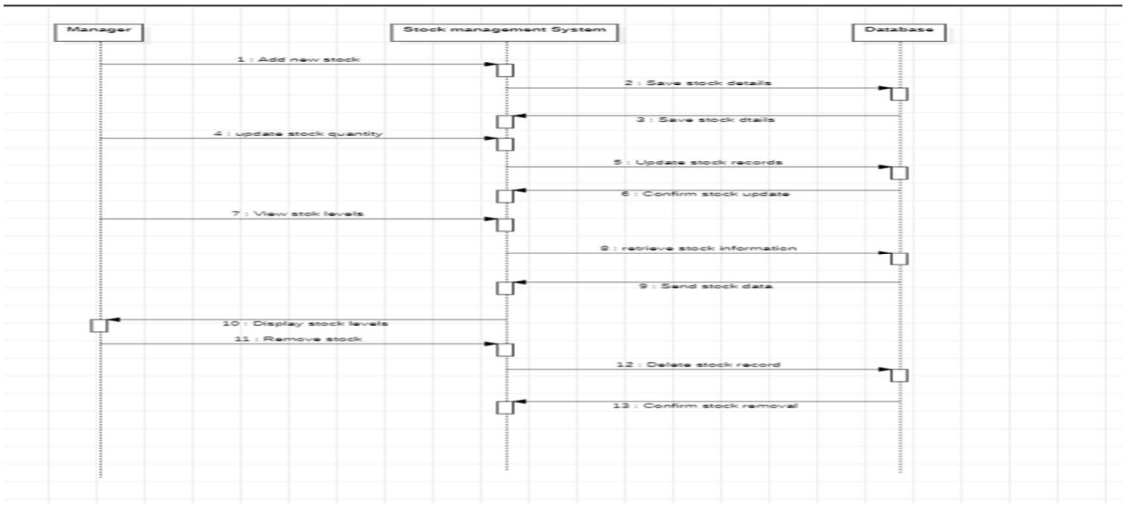
Store Manager/Staff:

- **Add Product**: Add new products to the system's inventory.
- **Search Products**: Search for specific products in the system.
- **Manage Store**: Oversee the operations and configuration of the store (e.g., stock levels, product details).
- **Create Bills**: Generate invoices for customer purchases.
- **Make Payments**: Process payments for sales transactions.

Customer:

- **Manage Customer**: Customer details, feedback, and transactions can be managed by the system.

Sequence Diagram



Description

User/Store Manager:

- **Initiates Stock Check**: The process begins when a user or store manager checks the stock availability for a product.

System (Stock Management):

- **Check Stock**: The system verifies the available quantity of the product in inventory.
- **Stock Available**: If the product is available, the system proceeds with the next steps, like placing an order or updating stock.
- **Stock Unavailable**: If the stock is insufficient, the system notifies the user of the "Out of Stock" status.

Order Processor:

- **Place Order**: If the stock is available, the order processor places the order, updating the inventory.
- **Update Stock**: The stock levels are adjusted after the order is processed.

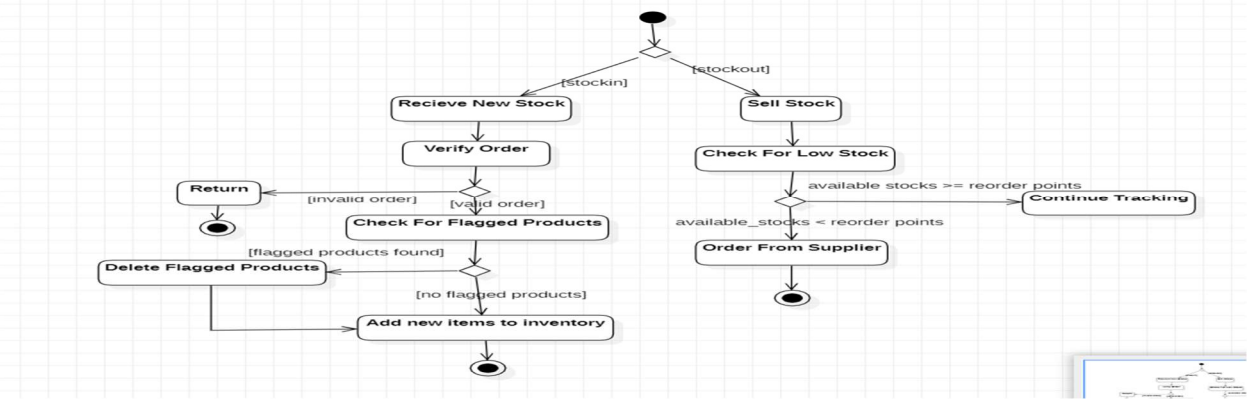
Supplier:

- **Send Shipment**: If stock is insufficient, the system sends a request to the supplier to ship more products.
- **Receive Shipment**: Upon receiving the stock, the system updates the inventory accordingly.

Customer:

- **Receive Confirmation**: After order processing, the customer is notified of successful order placement or out-of-stock status.
-

Activity Diagram



Description

Receive New Stock: The process begins with receiving new stock into the system.
Verify Order: The system checks if the incoming order is valid.

- **Invalid Order**: If the order is invalid, the system stops the process.
- **Valid Order**: If valid, the system proceeds to check for flagged products.

Check for Flagged Products: The system verifies if any products in the order are flagged for issues.

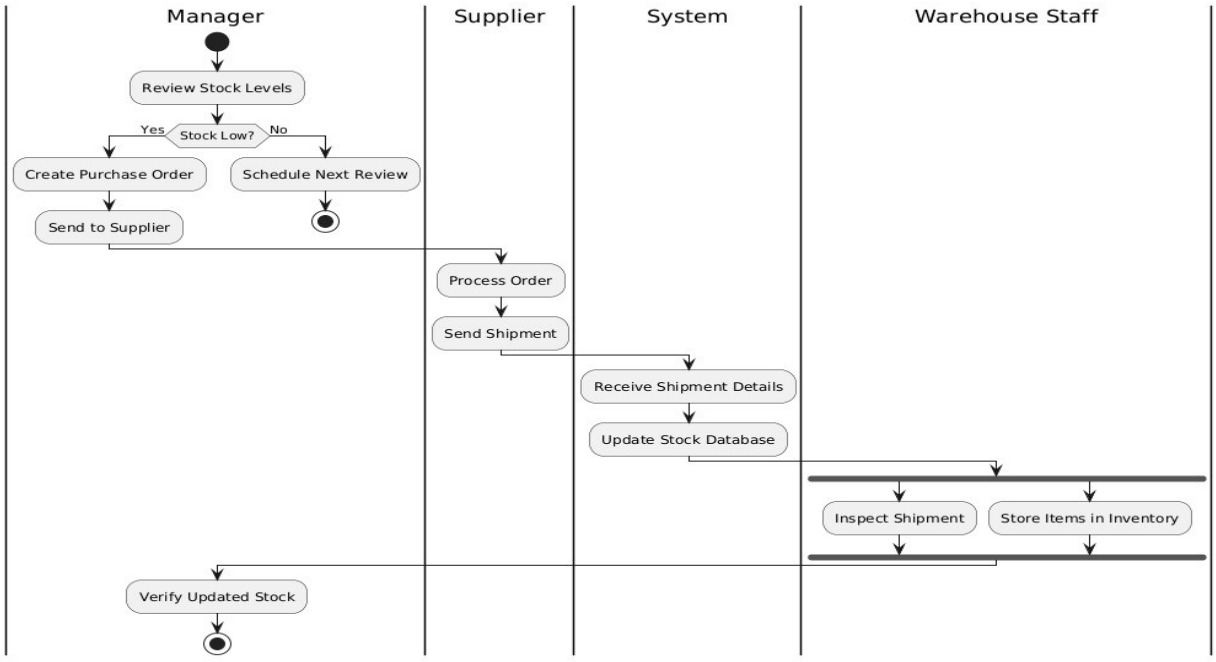
- **Flagged Products Found**: If flagged products are found, they are deleted from the order.
- **No Flagged Products**: If no flagged products, the system moves on to inventory updates.

Add New Items to Inventory: The system adds the ordered or newly received items to the inventory.
Sell Stock: As stock is sold, the system reduces the available stock levels.
Check for Low Stock: The system monitors stock levels.

- **Low Stock**: If stock levels fall below the reorder point, the system triggers an order to the supplier.

Reorder Points: The system checks the reorder points for products and places an order with the supplier.

Advanced Activity Diagram



Description

Swimlane 1: Store Manager

- **Receive New Stock**: The store manager receives and verifies incoming stock from suppliers.
- **Verify Order**: The store manager verifies customer orders, ensuring the requested stock is available.
- **Add New Items to Inventory**: New items are added to the inventory upon successful receipt and verification.
- **Check for Low Stock**: The manager monitors inventory levels and flags products approaching reorder levels.

Swimlane 2: System

- **Validate Stock**: The system automatically checks the stock levels for each item in the order.
- **Flagged Products Check**: The system checks for any flagged products (e.g., expired or defective) and alerts the store manager.
- **Update Inventory**: Once the order is processed or stock is received, the system updates the inventory levels accordingly.
- **Order Replenishment**: The system triggers reorder requests when stock levels hit the predefined reorder point.

Swimlane 3: Supplier

- **Ship Products**: When stock levels are low, the supplier ships the requested products.
- **Deliver Shipment**: The supplier delivers the goods, and the store manager verifies the shipment.

Swimlane 4: Customer

- **Place Order**: Customers place orders via the website or in-store, which triggers the order verification process.
- **Receive Confirmation**: The customer receives order confirmation and tracking details.

5. Passport Automation System

Problem Statement

The current passport application and processing system is manual, resulting in delays, errors, and inefficiencies in handling large volumes of passport requests. The process involves significant paperwork, long wait times for applicants, and difficulties in tracking application statuses. There is a need for an automated **Passport Automation System** to streamline application submissions, improve processing times, reduce human error, and enhance the overall user experience.

Software Requirement Specification(SRS)

Introduction

- **Purpose**: Automate passport application, verification, and issuance to improve efficiency.
- **Scope**: Handle online applications, document verification, payments, and passport issuance.
- **Overview**: A web platform for applicants to apply, track, and receive updates; staff to process applications.

2. General Description

- **Users**: Applicants (apply and track), Staff (verify and approve), Admin (manage users and settings).
- **Features**: Online application, document upload, payment integration, status tracking, notifications.
- **Benefits**: Faster processing, better user experience, reduced errors, streamlined management.

3. Functional Requirements

- **Applicant**: Submit application, make payments, track status.
- **Staff**: Verify documents, approve/reject applications, issue passports.
- **Admin**: Manage users, monitor system performance.

4. Interface Requirements

- **UI**: Intuitive web interface for all users.
- **System**: Integrate with databases for verification, payment systems, and notifications.

5. Performance Requirements

- **Availability**: 99% uptime.
- **Response Time**: <2 seconds for status updates.
- **Scalability**: Handle 10,000 applications/day.

6. Design Constraints

- **Platform**: Web-based, compatible with modern browsers.
- **Security**: Data encryption, secure payments, and user authentication.

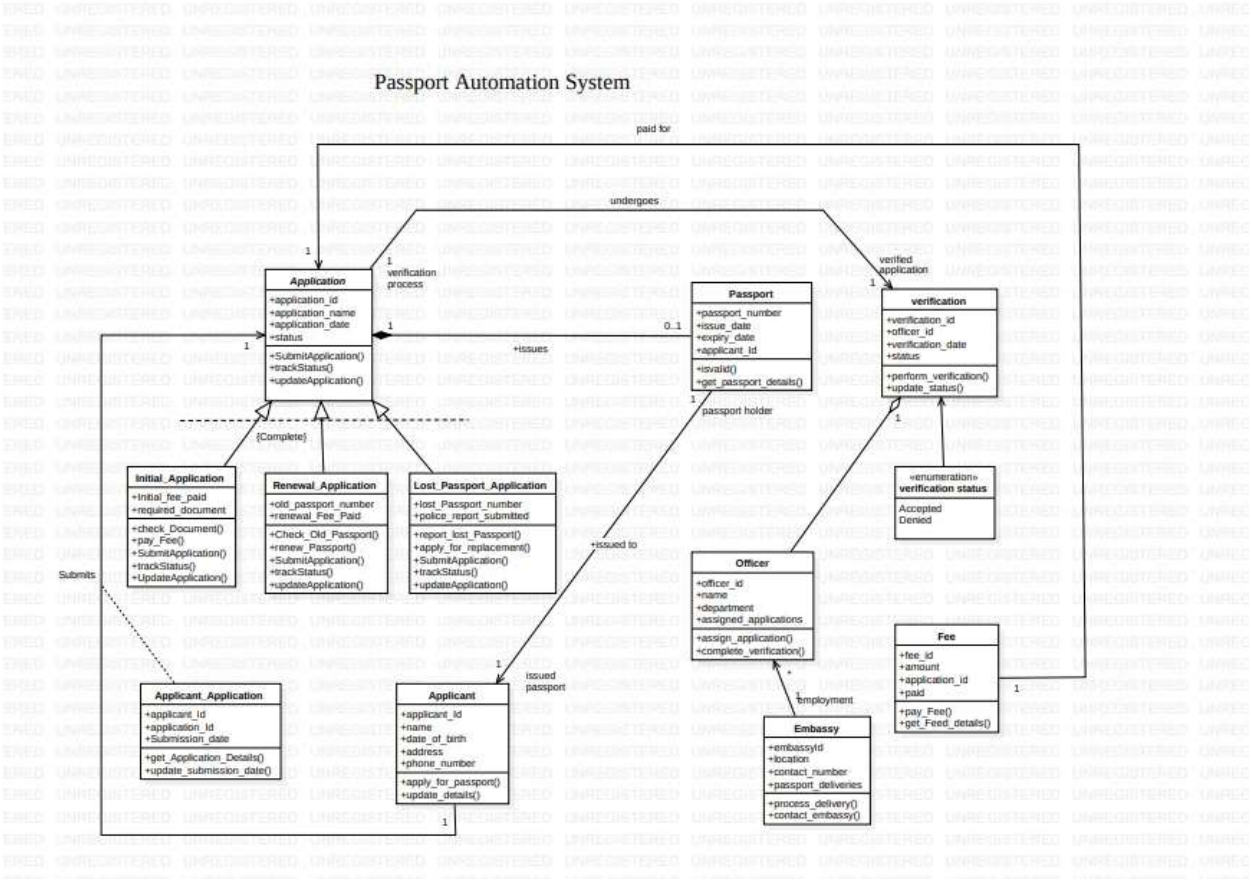
7. Non-Functional Requirements

- **Usability**: Easy to navigate for all users.
- **Reliability**: Stable performance, minimal downtime.
- **Security**: Strong encryption and compliance with data privacy laws.

8. Preliminary Budget and Time

- **Budget**: $150,000.
- **Timeline**: 6 months for development.

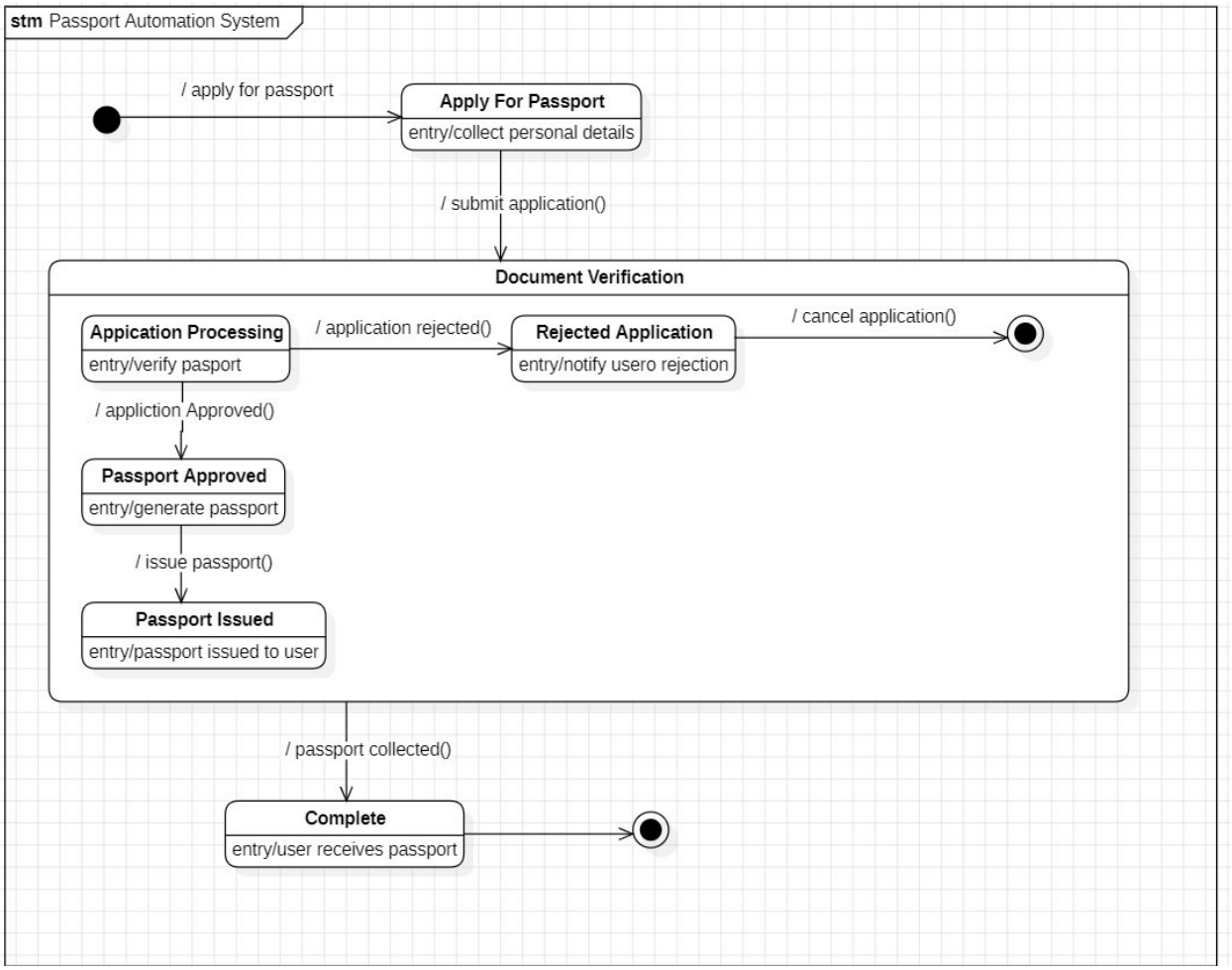
Class Diagram



Description

Applicant Class: Holds attributes like name, address, and passport application details. Methods include submitting applications and tracking status.
Staff Class: Responsible for verifying documents and approving/rejecting applications. Includes methods for validating documents and processing applications.
Payment Class: Handles payment details, including transaction ID, amount, and payment status. Methods include processing payments and verifying payment status.
Passport Class: Represents the issued passport with attributes like passport number, issue date, and expiry date. Methods include generating a new passport and updating details.
Notification Class: Manages sending notifications (email/SMS) to applicants, including methods for sending status updates and alerts.
Admin Class: Manages user roles, system settings, and oversees application processes.

State Diagram



Description

Apply for Passport: The process begins with the applicant submitting personal details and the application.
Document Verification: The application enters a verification stage where documents are checked.

- If valid, it moves to **Application Processing**.
- If invalid, the applicant can **cancel the application**.

Application Processing: In this state, the application is reviewed for approval.

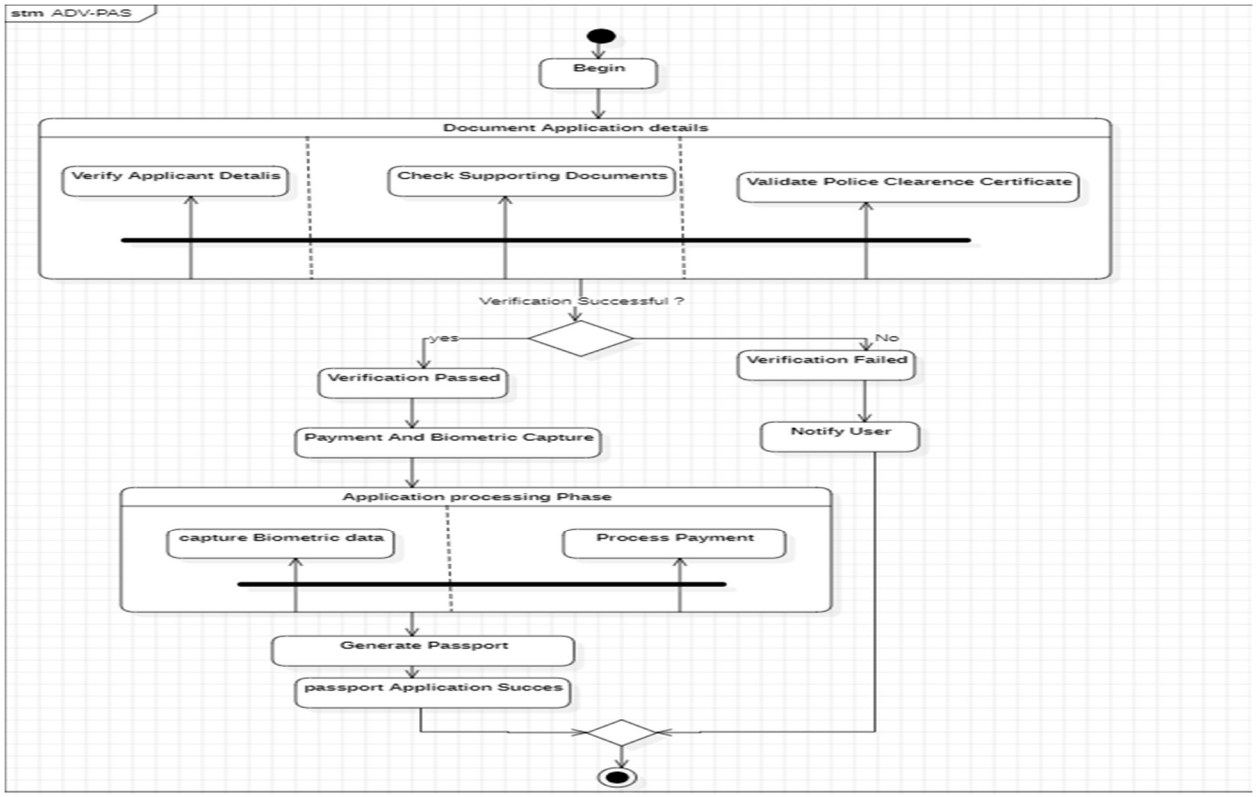
- If approved, it transitions to **Passport Approved**.
- If rejected, the system notifies the applicant of the rejection.

Passport Approved: Once approved, the system generates the passport.

- The applicant then receives the passport in the **Passport Issued** state.

Rejected Application: If the application is rejected, the applicant is notified of the rejection and the process ends.
Complete: The final state, where the passport is either issued or the application is rejected.

Advanced State Diagram



Description

Submit Application: The applicant begins the process by entering personal details and submitting the passport application.

Validate Credentials: The system validates the applicant's details (e.g., identity check, address verification).

- If valid, proceed to biometric data collection.
- If invalid, the process is halted, and the applicant is notified.

Biometric Data Collection: The applicant provides biometric data (fingerprints, photo, etc.), which is verified.

- If biometric data matches, the process moves to **Application Processing**.
- If invalid or failed, the application is flagged for rejection.

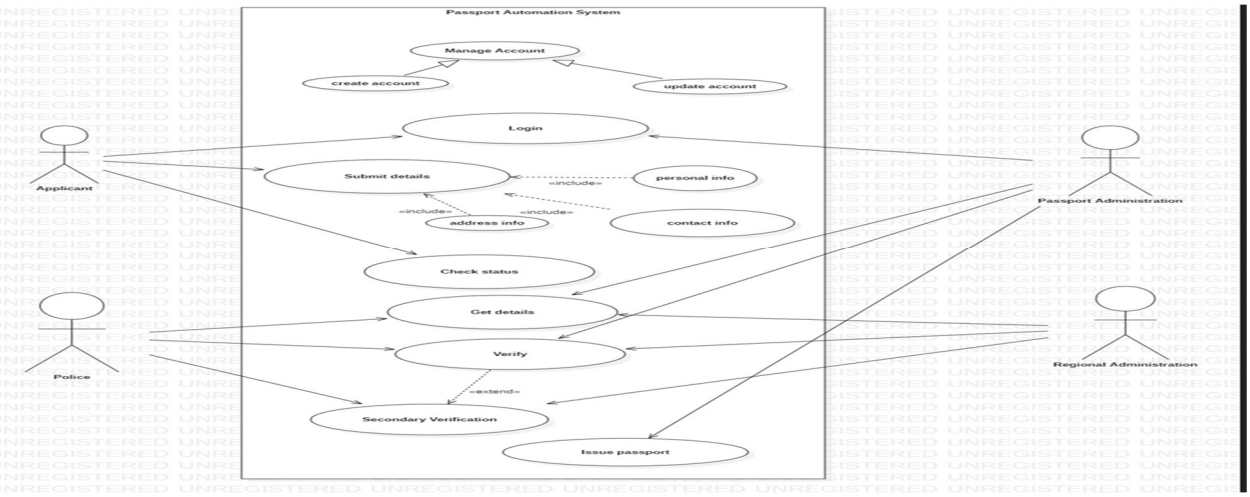
Application Processing: The system processes the application, including verification of submitted documents and biometric data.

- If everything is valid, the application moves to the **Passport Issuance** state.
- If any issues arise, the application may be sent for **Manual Review**.

SMS Notification: Once the application is processed, the applicant receives an SMS notification about their application status (approved or rejected).

Complete: If the application is approved, the passport is issued, and the process is complete. If rejected, the applicant is notified via SMS, and the process is concluded.

Use-Case Diagram



Description

Applicant:

- **Manage Account**: Applicants can view and update their personal information.
- **Submit Passport Application**: Applicants can fill out and submit their application.
- **Track Application Status**: Applicants can track the progress of their passport application.
- **Make Payment**: Applicants can pay for the passport processing fee.
- **Receive Passport**: Upon approval, the applicant receives their passport.

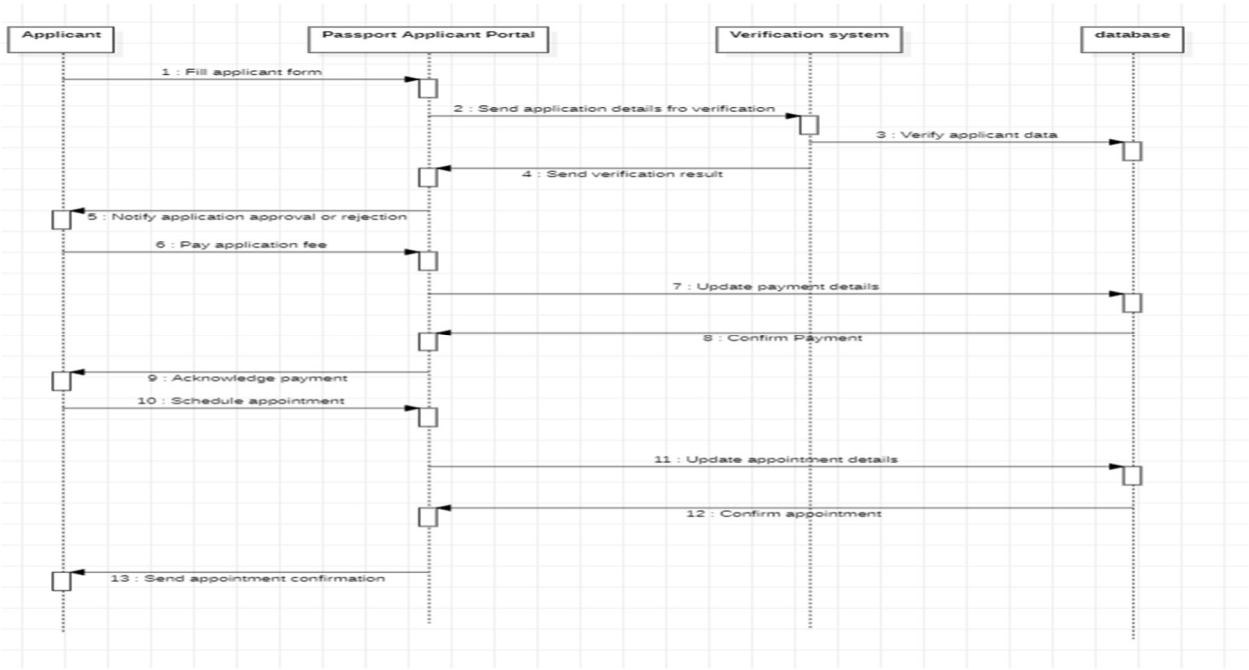
Admin:

- **Manage Admin Info**: Admin can manage system settings, including user roles and permissions.
- **Verify Application**: Admin validates submitted passport applications, including documents and biometric data.
- **Approve/Reject Application**: Admin reviews applications and approves or rejects based on verification.

Secondary Verification:

- **Verify Identity**: Secondary verification process for applicants' identity, which may involve additional checks or manual review.

Sequence Diagram



Description

Applicant Submits Details: The applicant provides personal details and submits the application.

Send Application for Verification: The system forwards the applicant's details to the verification service (e.g., admin or biometric system) for validation.

Verification Result: The system receives the result of the verification (approved or rejected) and notifies the applicant of the approval.

Applicant Makes Payment: Upon approval, the applicant proceeds with the payment for the application.

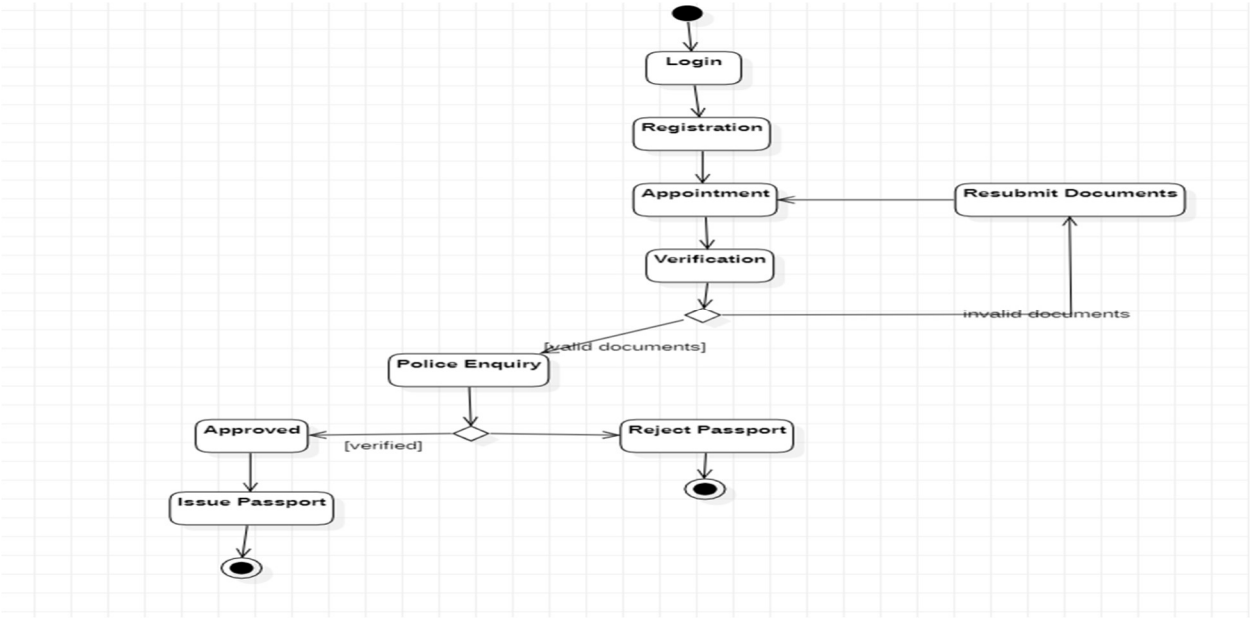
Payment Confirmation: Once the payment is processed, the system confirms the payment and acknowledges the applicant's transaction.

Appointment Scheduling: The applicant is notified of the appointment for biometric data collection or interview.

Appointment Details: The system sends appointment details (date, time, location) to the applicant.

Verify Data: The system verifies the data during the appointment, which can include biometric scanning or document verification.

Activity Diagram



Description

Login: The applicant logs into the system, either with existing credentials or by registering a new account.

Registration: If not already registered, the applicant creates a new account by submitting personal information.

Appointment: Once registered or logged in, the applicant schedules an appointment for biometric data collection or document verification.

Document Verification: The applicant submits required documents, and the system verifies them.

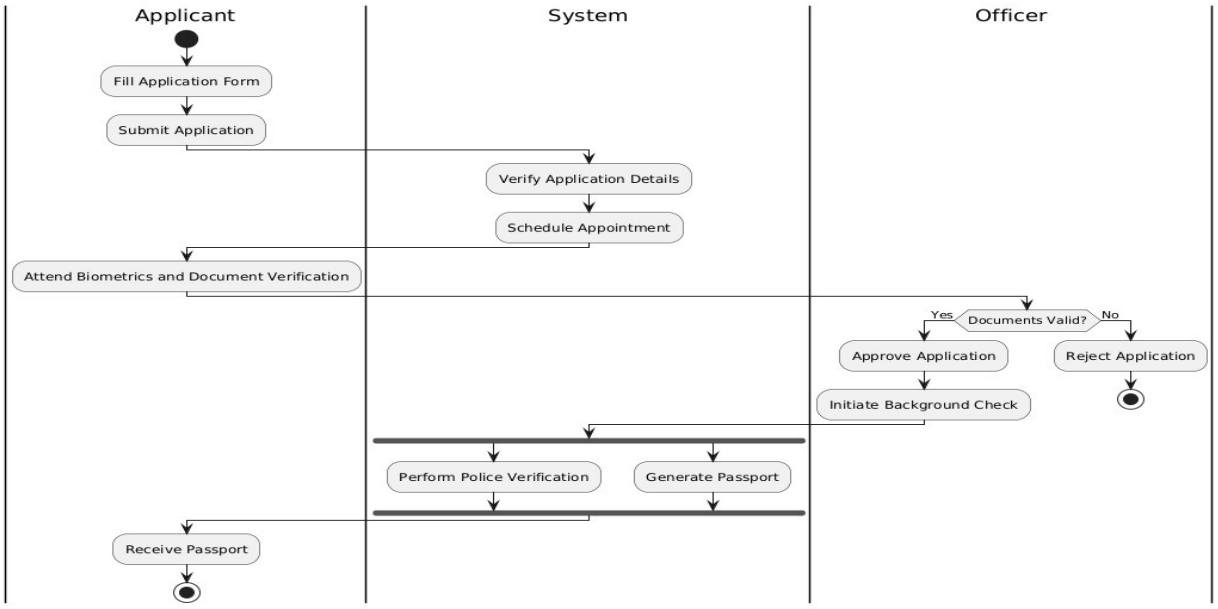
- If the documents are **valid**, the process moves to the next stage.
- If **rejected**, the applicant is asked to **resubmit documents**.

Police Enquiry: A police enquiry is initiated for security checks. If the enquiry clears, the application proceeds; if **flagged**, the process is halted or further action is needed.

Approval: If all checks are successful, the application is **approved**. The system generates and issues the passport.

Issue Passport: The final step, where the passport is issued to the applicant, completing the process.

Advanced Activity Diagram



Description

Applicant Lane:

- **Login/Registration**: The applicant logs into the system or registers a new account.
- **Schedule Appointment**: The applicant schedules a biometric or document verification appointment.
- **Submit Documents**: The applicant uploads necessary documents for verification.
- **Resubmit Documents**: If documents are rejected, the applicant resubmits them.

System Lane:

- **Verify Documents**: The system verifies the submitted documents.
- **Trigger Police Enquiry**: The system triggers a police enquiry for background checks.
- **Notify Applicant**: The system notifies the applicant about the status of document verification, appointment scheduling, or rejections.
- **Process Payment**: After approval, the system processes the application payment.

Admin Lane:

- **Review Documents**: The admin manually reviews submitted documents and performs additional verification as needed.
- **Approve/Reject Application**: The admin approves or rejects the passport application based on document verification and police enquiry results.

Police Lane:

- **Conduct Police Enquiry**: The police department performs background checks or security screening.
- **Flag Issues**: If any issues are flagged, the process can be delayed or halted.

Final Stage:

- **Issue Passport**: Once all steps are complete and the application is approved, the system issues the passport to the applicant.