**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



**LAB REPORT**

# on

OBJECT ORIENTED MODELING

***Submitted by***

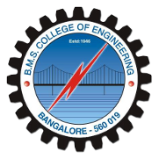
# Sanvi Nadiga (1BM22CS245)

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

# September-2024 to January-2025

**B. M. S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “**OBJECT ORIENTED MODELING**” was carried out by **Sanvi Nadiga (1BM22CS245),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-2025. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Modeling- (23CS5PCOOM)** work prescribed for the said degree.

**Lakshmi Neelima Dr. Kavitha Sooda**

Assistant Professor Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

**Table of Content**

| **Sl.**  **No.** | **Experiment Title** | **Page No.** |
| --- | --- | --- |
| **1** | **Library Management System** | **4** |
| **2** | **Hotel Management System** | **11** |
| **3** | **Stock Management System** | **17** |
| **4** | **Credit Card Management System** | **24** |
| **5** | **Passport Management System** | **30** |

# 1. Library Management System

1. **Problem Statement**

Design and implement a **Library Management System** to streamline the operations of a library. The system should manage a catalog of items, including articles, magazines, and books, all organized within an online database. Users and librarians are central to the system, with librarians responsible for managing the catalog, issuing library cards, and overseeing borrowing and returning operations. Users can search the catalog, borrow items, and view their borrowing history using a library card that tracks their unique ID, issue date, expiry date, and borrowing limit. The catalog items should be classified into specific types—articles, magazines, and books—with attributes such as title, author, publication date, and genre, depending on the type. The system should enable librarians to add, update, or remove catalog items and maintain a record of borrowing activity and overdue penalties. This comprehensive system aims to simplify library operations, enhance user experience, and ensure efficient management of library resources.

1. **SRS - Software Requirements Specification**

## Introduction

* + 1. Purpose

The Library Management System (LMS) aims to automate library operations such as book management, user registration, borrowing, and returning of books. It will provide an intuitive interface for librarians and users to interact with the system efficiently.

* + 1. Scope

The LMS will be a centralized system that supports:

* + - * Maintaining records of books, users, and transactions.
      * Facilitating book search, issue, and return processes.
      * Generating reports for inventory and overdue books.
    1. Definitions, Acronyms, and Abbreviations
       - LMS: Library Management System.
       - User: A library member who can borrow/return books.
       - Admin/Librarian: The person managing library operations.
    2. References
       - IEEE Standard 830-1998 for SRS documentation.
       - Library operation manuals and workflows.



## System Overview

The system will include:

* User Management Module
* Book Inventory Management
* Borrow and Return Module
* Reports and Analytics
* Notifications for overdue books.



## Functional Requirements

* + 1. User Management
       - Users can register and log in to their accounts.
       - Admin can add, update, or delete user records.
       - Role-based access: Admin vs. User.
    2. Book Management
       - Add, update, delete, and categorize books.
       - Search for books by title, author, genre, or ISBN.
    3. Borrow and Return
       - Users can borrow books (limited to 3 books at a time).
       - Borrowing duration: 14 days; fines for overdue returns.
       - Return process updates the book inventory.
    4. Notifications
       - Email or SMS reminders for due dates and overdue books.
    5. Reports and Analytics
       - Generate reports on:
         * Total books issued.
         * Overdue books.
         * Popular books.



## Non-Functional Requirements

* + 1. Performance
       - The system should handle up to 500 concurrent users.
       - Search functionality must return results within 3 seconds.
    2. Scalability
       - The system should support expansion to accommodate more libraries.
    3. Usability
       - User-friendly UI for both users and admins.
    4. Security
       - Data encryption for sensitive information.
       - Role-based access control.
    5. Availability
       - The system should be operational 24/7 with 99.9% uptime.



## System Models

* + 1. Use Case Diagram

1. Actors: User, Admin.
2. Key Use Cases:
   * Search books
   * Borrow/return books
   * Manage inventory (Admin)
   * View notifications (User)
     1. Data Flow Diagram
3. Level 0: User inputs -> LMS -> Output (Search results, Borrow status, etc.)
4. Level 1: Breaks down into User Management, Book Management, and Borrow/Return.
   * 1. ER Diagram

* Tables: Users, Books, Transactions, Notifications.
* Relationships:
  + Users ↔ Transactions (One-to-Many)
  + Books ↔ Transactions (One-to-Many)

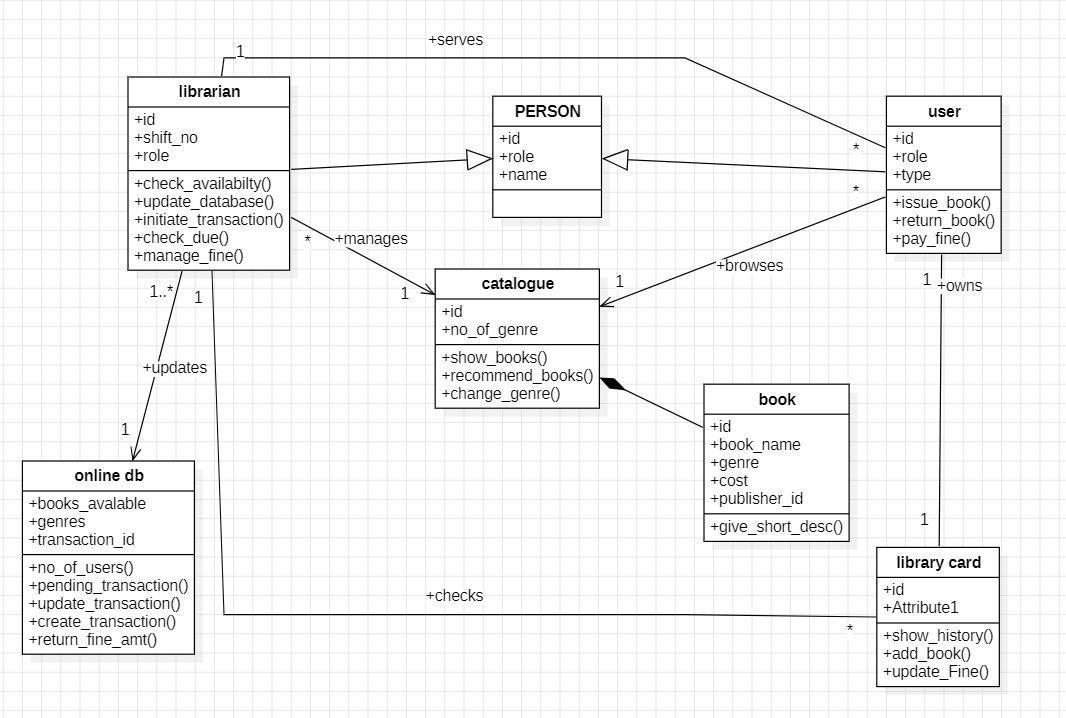


## System Requirements

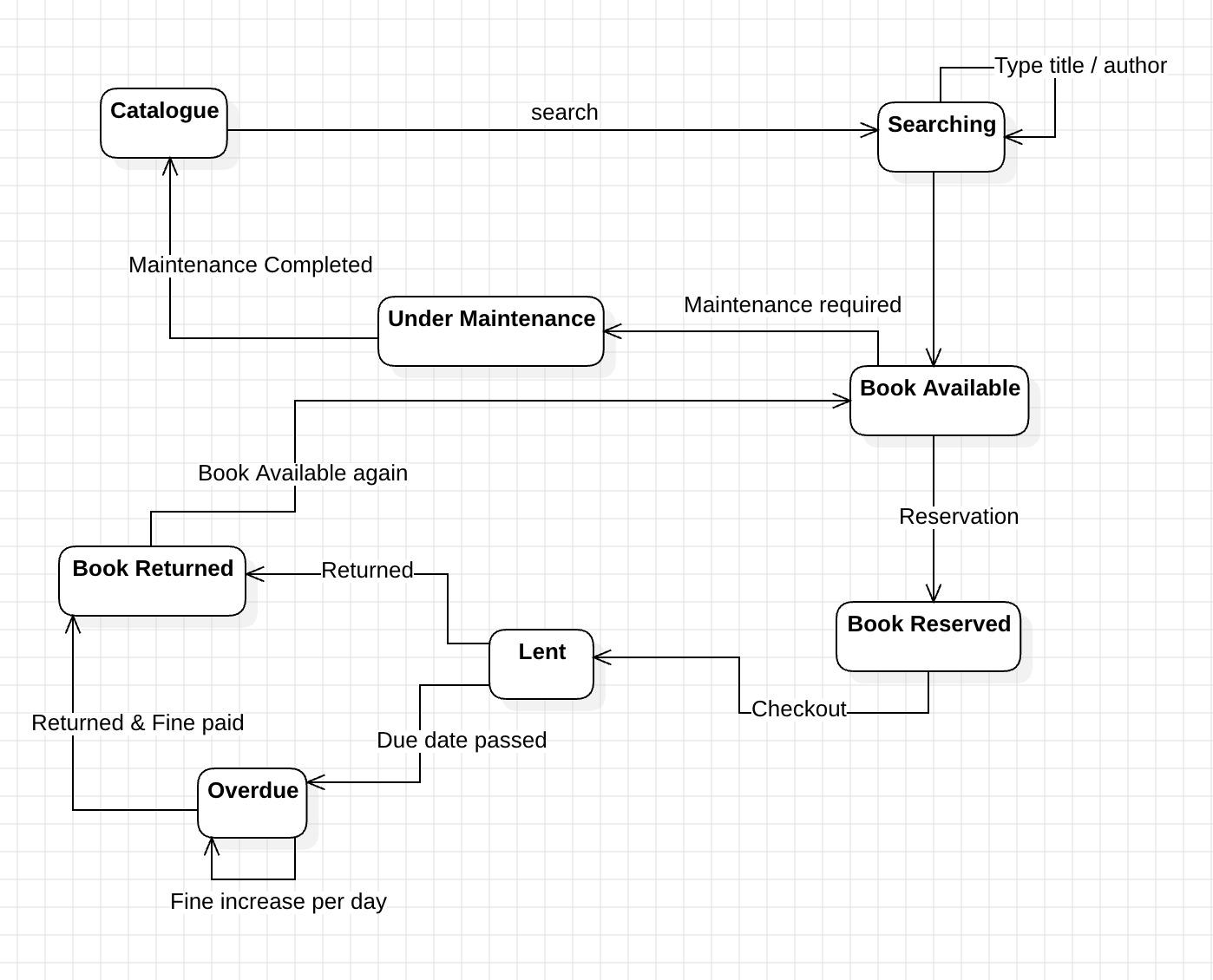
* + 1. Hardware Requirements
       - Server with at least 8GB RAM, 500GB SSD.
       - Client devices: Desktop or mobile with browsers.
    2. Software Requirements
       - Backend: Node.js / Django / Spring Boot.
       - Frontend: React.js / Angular / Flutter.
       - Database: MySQL / PostgreSQL.
       - OS: Windows/Linux/Mac.



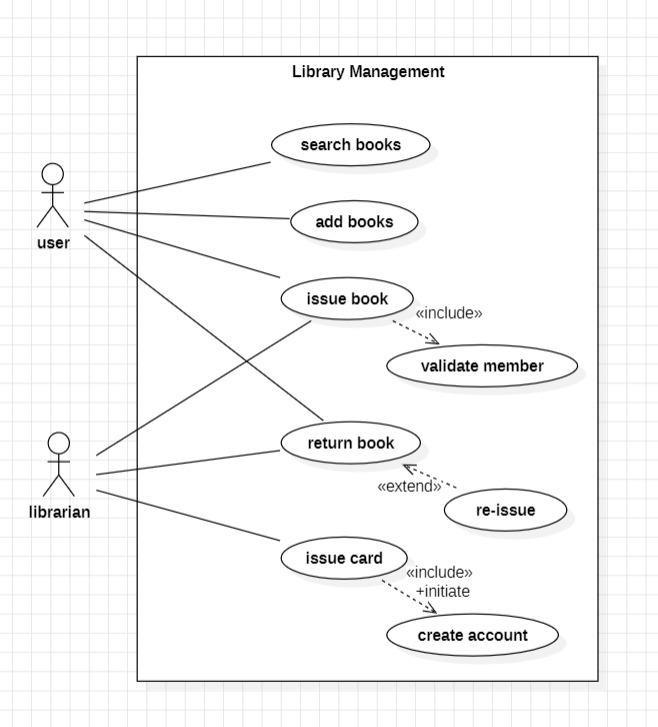
1. **Class Diagram**



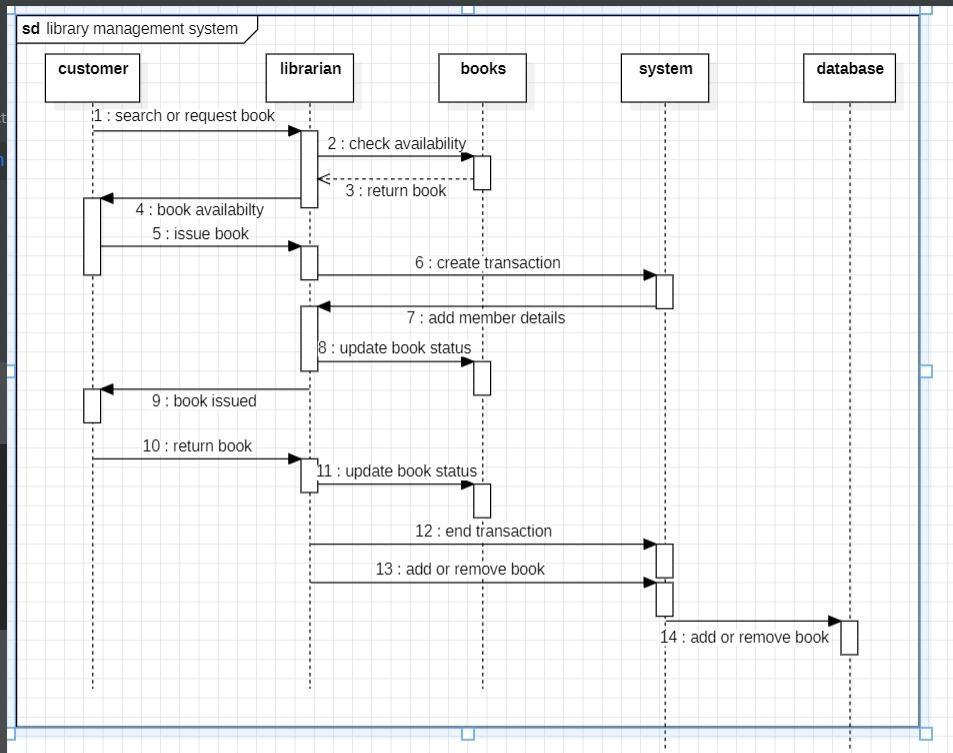
1. **State Diagram**



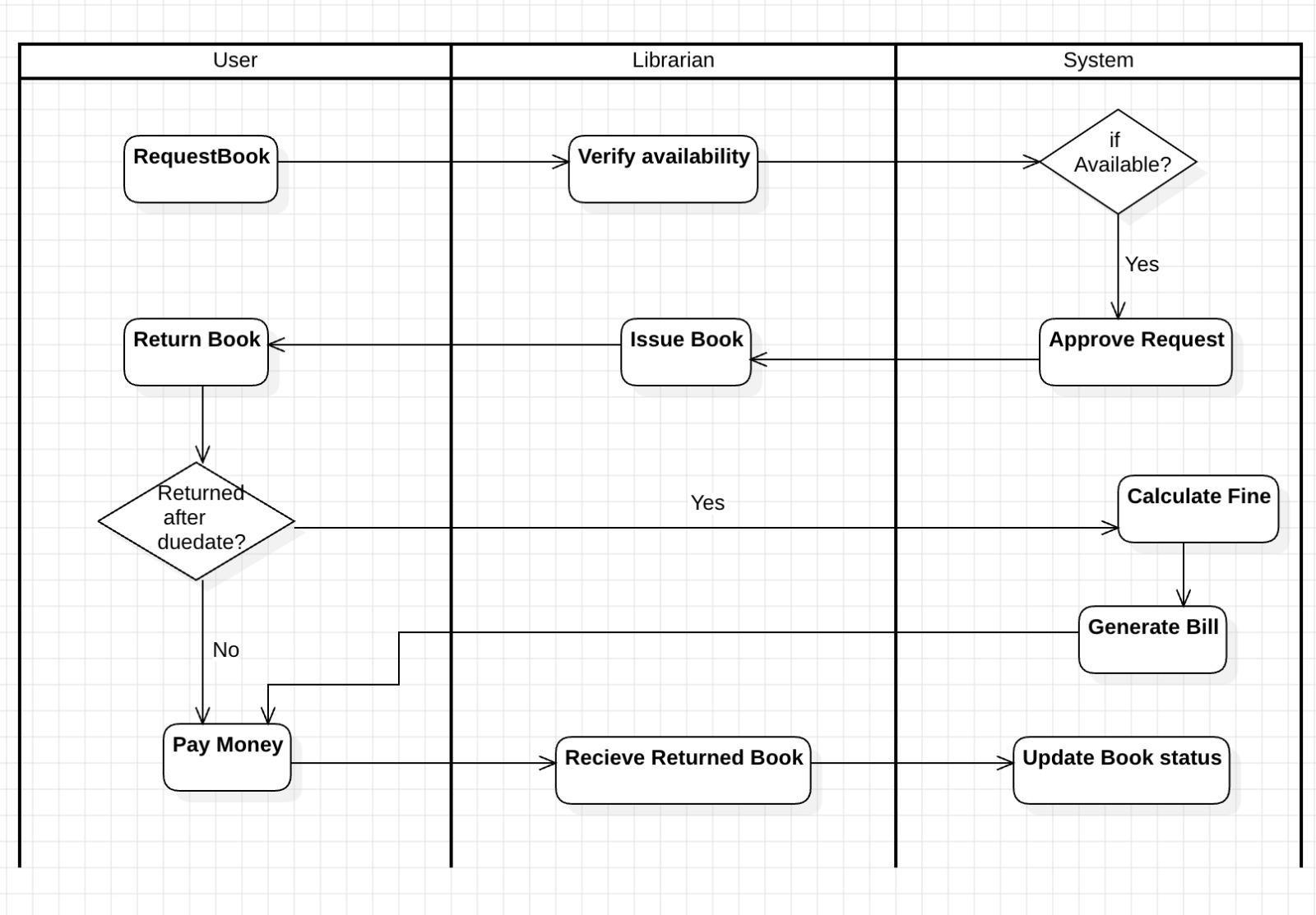
1. **Use Case Diagram**



1. **Sequence Diagram**



1. **Activity Diagram**



1. **Problem Statement**

# Hotel Management System

Develop a **Hotel Management System** to efficiently manage the operations and services of a hotel. The system should handle various aspects such as customer bookings, order management, staff responsibilities, and billing. The hotel consists of multiple roles, including customers, receptionists, hotel staff, chefs, and managers, each with distinct responsibilities. Customers can book rooms, place orders, and receive bills for their stay and services.

Receptionists manage room bookings, check-ins, and check-outs, ensuring seamless customer experiences. Chefs handle food orders from customers, while hotel staff maintain cleanliness and fulfill customer service requests.

Managers oversee the overall operations, including staff performance and customer satisfaction. The system should include functionalities for managing hotel rooms, such as availability, types, and pricing, and generating detailed bills for services provided. By integrating these functionalities, the system aims to enhance efficiency, improve customer satisfaction, and streamline hotel operations.

1. **SRS - Software Requirements Specification**
   1. **Introduction**
      1. **Purpose**

The Hotel Management System (HMS) is designed to streamline and automate hotel operations such as room bookings, customer management, billing, and reporting. It provides a centralized platform for hotel administrators, staff, and customers to interact efficiently.

* + 1. **Scope**

The HMS will cater to hotels of all sizes, offering modules for:

* + - * Room and reservation management.
      * Customer and staff management.
      * Billing and payment processing.
      * Generating reports for occupancy, revenue, and customer feedback.
    1. **Definitions, Acronyms, and Abbreviations**
       - HMS: Hotel Management System.
       - Customer: A guest staying at or reserving the hotel.
       - Admin: Hotel staff managing operations.
    2. **References**
       - IEEE Standard 830-1998 for SRS documentation.
       - Industry-standard hotel operation workflows.
  1. **System Overview**

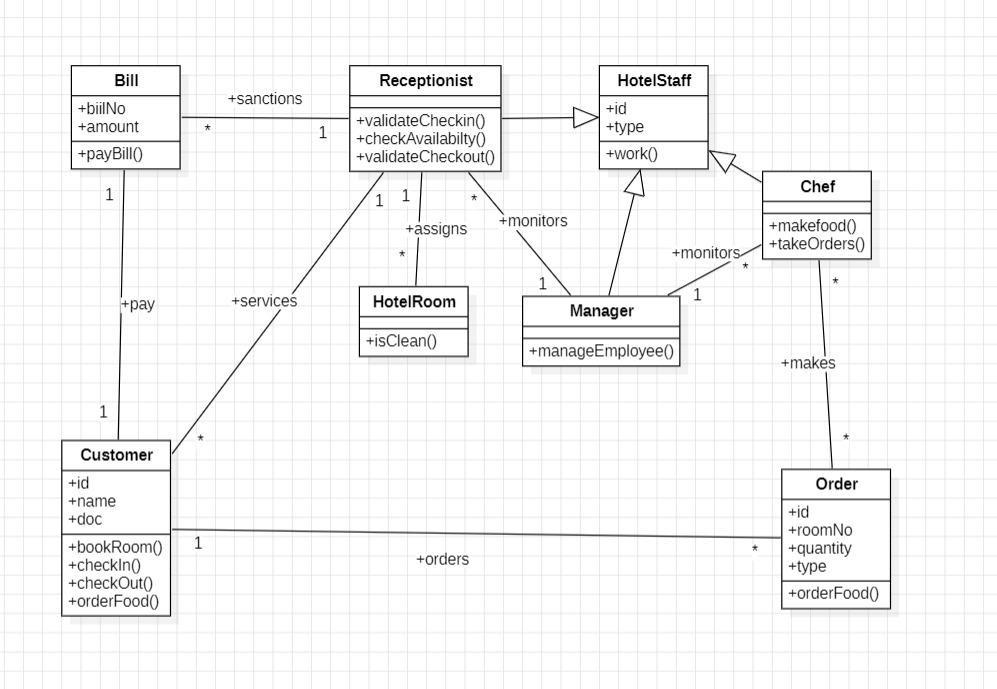
The HMS will include:

* Reservation Module: Room booking and availability management.
* Customer Management Module: Maintaining guest details.
* Billing Module: Automated invoice generation and payment tracking.
* Reporting Module: Generating operational and financial reports.
* Feedback Module: Gathering customer reviews.
  1. **Functional Requirements**
     1. **User Management**
        + Users can register and log in (Admin and Customers).
        + Admin can manage customer and staff records.
     2. **Room Management**
        + Add, update, delete room details.
        + Manage room categories (e.g., deluxe, suite, standard).
        + Display room availability in real-time.
     3. **Reservation Management**
        + Customers can search, book, modify, or cancel reservations.
        + Admin can view, confirm, or cancel bookings.
        + Integration with payment gateways for online booking payments.
     4. **Billing and Payments**
        + Generate invoices for customer stays.
        + Accept multiple payment modes (cash, card, online).
        + Track payment history.
     5. **Reporting**
        + Occupancy reports.
        + Revenue reports by room category, period, or payment type.
        + Feedback and review analysis.
     6. **Notifications**
        + Send booking confirmation, reminders, and promotions via email/SMS.
  2. **Non-Functional Requirements**
     1. **Performance**
        + Handle up to 1,000 concurrent users.
        + Ensure response time for booking confirmation is under 2 seconds.
     2. **Scalability**
        + Support multi-hotel setups for chains.
     3. **Usability**
        + Responsive design for use on desktops, tablets, and smartphones.
     4. **Security**
        + Encrypt sensitive customer and payment data.
        + Role-based access for Admin and Customers.
     5. **Availability**
        + Ensure 99.9% system uptime.
  3. **System Models**
     1. **Use Case Diagram**

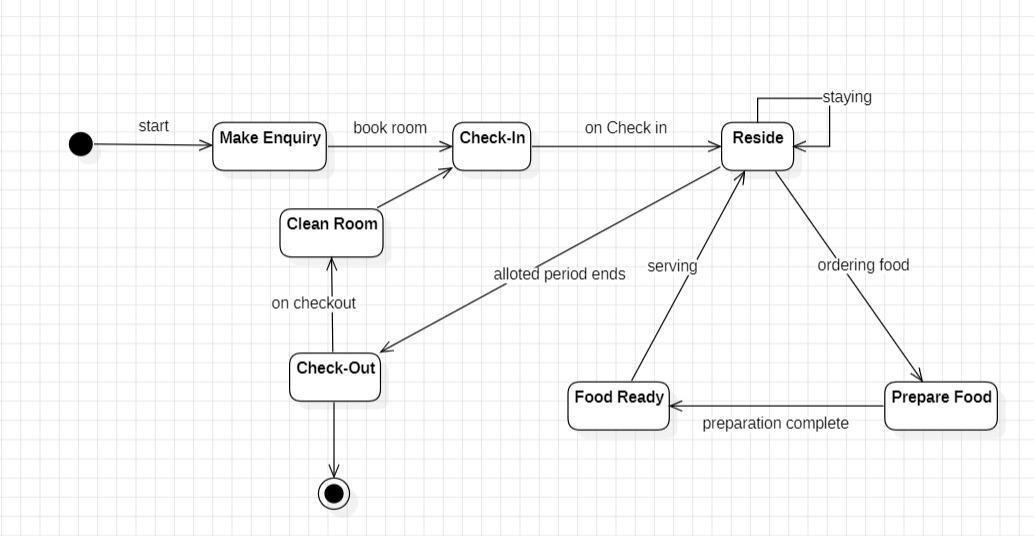
Actors:

* + - * Customer: Search rooms, book rooms, cancel bookings, make payments.
      * Admin: Manage rooms, confirm reservations, generate invoices, view reports.
    1. **Data Flow Diagram**

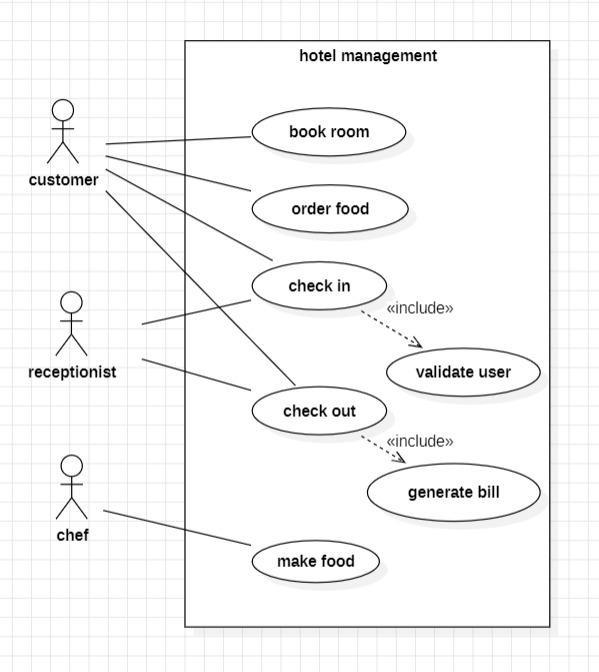
1. Level 0: Inputs (room search, booking) → HMS → Outputs (availability, booking confirmation).
2. Level 1: Divided into Reservation, Billing, and Reporting processes.
   1. **System Requirements**
      1. **Hardware Requirements**
         * Server: Minimum 16GB RAM, 1TB SSD, quad-core processor.
         * Client Devices: Desktop/laptop with a modern browser or mobile device.
      2. **Software Requirements**
3. Backend: Node.js / Django / Spring Boot.
4. Frontend: React.js / Angular / Flutter.
5. Database: MySQL / PostgreSQL.
6. Payment Gateway: Stripe / Razorpay / PayPal.
7. **Class Diagram**



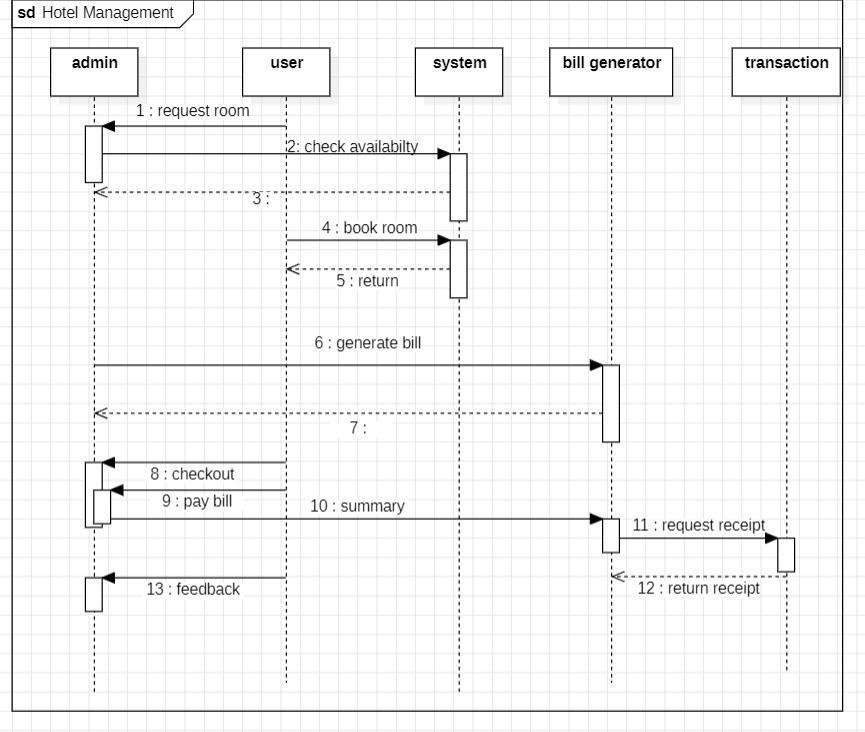
1. **State Diagram**



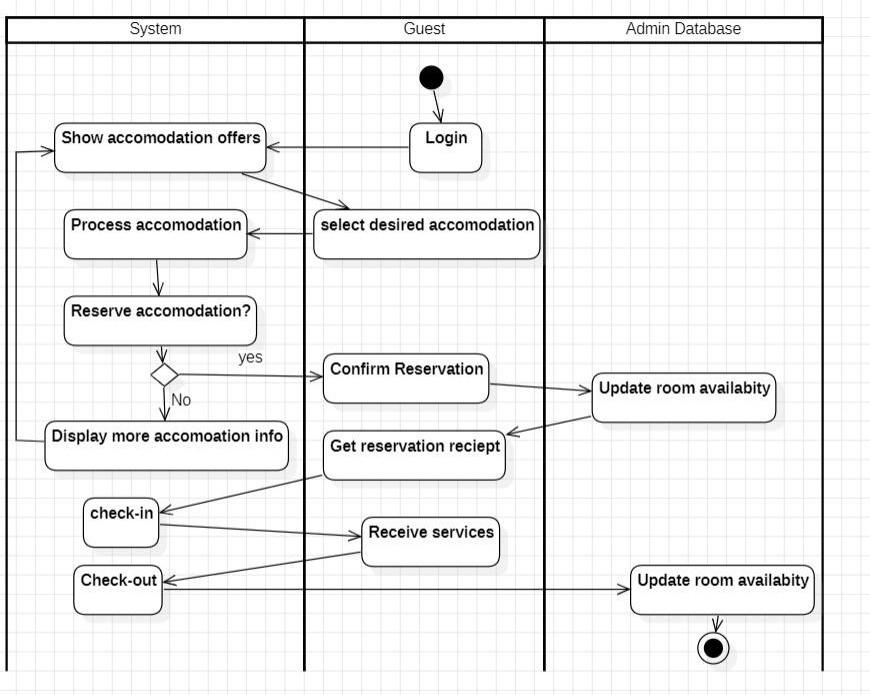
1. **Use Case Diagram**



1. **Sequence Diagram**



1. **Activity Diagram**



1. **Problem Statement**

# 3. Stock Management System

Design and implement a robust **Stock Management System** for a retail business. The system should effectively track and manage inventory levels, process orders from dealers, facilitate communication with product suppliers, and provide valuable insights to the StockManager for informed decision-making. The system should utilize

object-oriented principles with well-defined classes such as StockManager, Dealer, OrderDetails, ProductSupplier, Inventory, Product, and Stock to ensure efficient data management and maintain a clear and organized structure.

1. **SRS - Software Requirements Specification**

## Introduction

* + 1. Purpose

The Stock Management System (SMS) is designed to efficiently manage inventory, track stock levels, and handle product orders for businesses. It aims to automate stock management processes, reduce manual errors, and ensure that stock levels are optimized to meet demand.

* + 1. Scope

The SMS will include the following functionalities:

* + - * Stock level tracking and updates in real time.
      * Automatic alerts for low stock levels.
      * Order management for restocking and sales.
      * Reporting and analytics for stock trends.
      * User roles for admins, staff, and suppliers.
    1. Definitions, Acronyms, and Abbreviations
       - SMS: Stock Management System.
       - Admin: The primary user responsible for managing the system.
       - Staff: Users handling daily operations.
       - Supplier: External entity providing stock.
    2. References
       - IEEE Standard 830-1998 for SRS documentation.
       - Inventory control best practices.



## System Overview

The SMS is a web-based and mobile-enabled application that manages inventory in real time. It supports:

* Stock Management: Add, update, and delete stock records.
* Order Tracking: Manage incoming (purchase) and outgoing (sales) orders.
* Notifications: Alerts for low stock and upcoming restocking.
* Reports: Generate inventory, sales, and order reports.



## Functional Requirements

* + 1. User Management
       - Admin: Add/remove users, assign roles, and oversee the system.
       - Staff: Perform daily operations like updating stock levels and managing orders.
    2. Stock Management
       - Add new products with details such as name, SKU, category, price, and quantity.
       - Update stock levels after sales or restocking.
       - Automatically adjust stock levels based on sales or returns.
    3. Order Management
       - Create and track purchase orders for suppliers.
       - Manage sales orders for customers.
       - Provide an order history for tracking and auditing.
    4. Notifications
       - Alert admins/staff when stock levels drop below the minimum threshold.
       - Notify about overdue purchase orders or pending customer orders.
    5. Reporting and Analytics
       - Generate reports on stock levels, sales trends, and supplier performance.
       - Provide analytics for frequently purchased products.
    6. Supplier Management
       - Store and manage supplier information.
       - Track supplier performance (e.g., delivery time, product quality).
    7. Audit Logs
       - Maintain logs of all stock-related actions (e.g., updates, deletions).
       - Allow admins to review changes for accountability.



## Non-Functional Requirements

* + 1. Performance
       - Process inventory updates within 2 seconds.
       - Support up to 1,000 concurrent users.
    2. Scalability
       - Handle multiple warehouses or locations.
       - Expand to support e-commerce integration.
    3. Usability
       - Provide a simple, user-friendly interface for all users.
       - Include training materials and a help section.
    4. Security
       - Use role-based access control to restrict sensitive features to admins.
       - Encrypt sensitive data such as user credentials and supplier details.
    5. Availability
       - Ensure 99.9% uptime for critical operations.



## System Models

* + 1. Use Case Diagram Actors:
       - Admin: Manage users, oversee stock levels, and generate reports.
       - Staff: Handle stock updates, orders, and notifications.
       - Supplier: Fulfill purchase orders.



## System Requirements

* + 1. Hardware Requirements
       - Server: Minimum 8GB RAM, 500GB SSD, dual-core processor.
       - Client Devices: Desktop or mobile device with modern browsers.
    2. Software Requirements
       - Backend: Django / Spring Boot / Node.js.
       - Frontend: React.js / Angular / Vue.js.
       - Database: PostgreSQL / MySQL / MongoDB.
       - Hosting: AWS / Azure / Google Cloud.

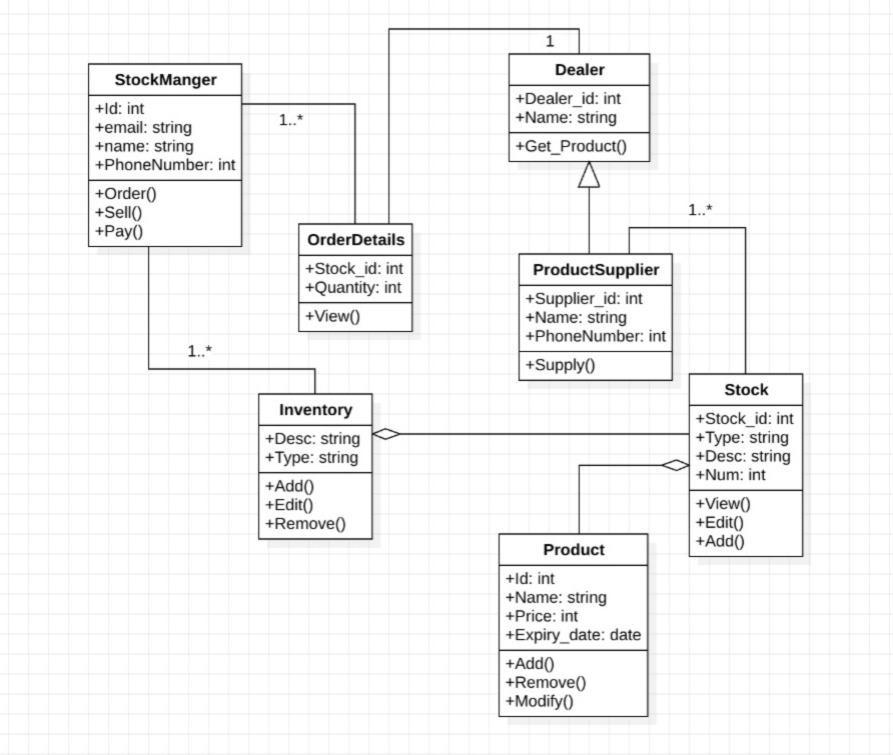


## Constraints

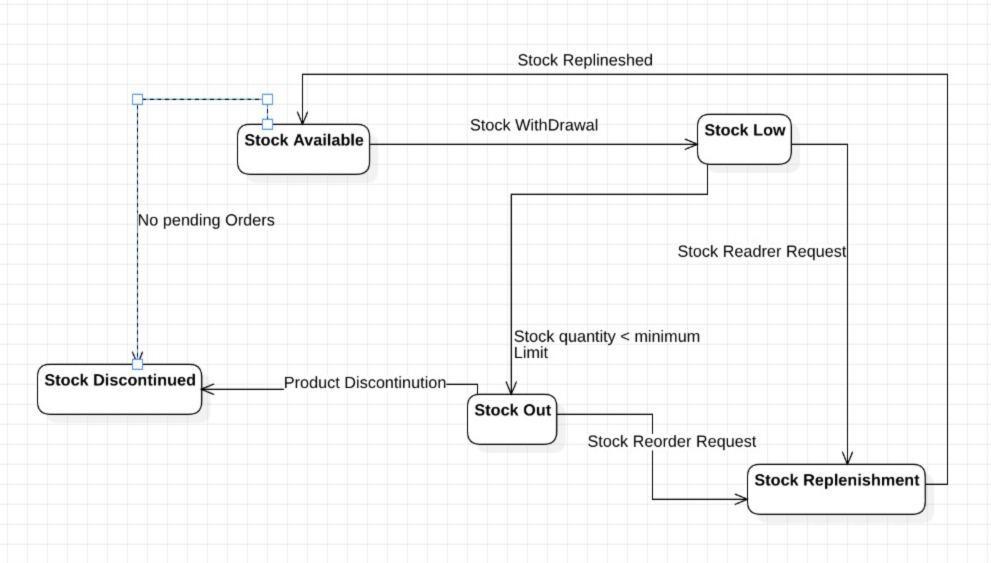
* Must integrate with barcode scanners for stock entry.
* Deployment deadline: 6 months.



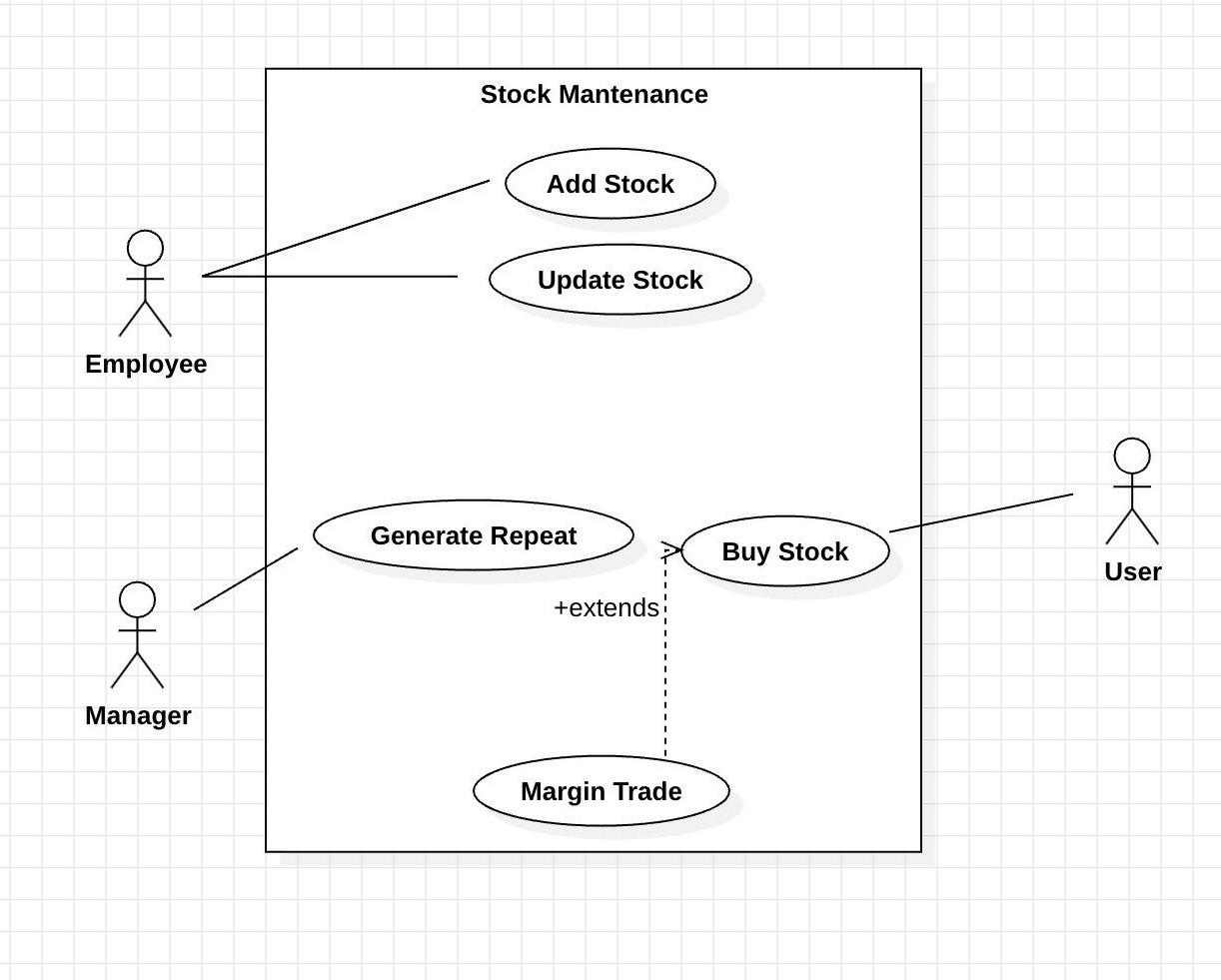
1. **Class Diagram**



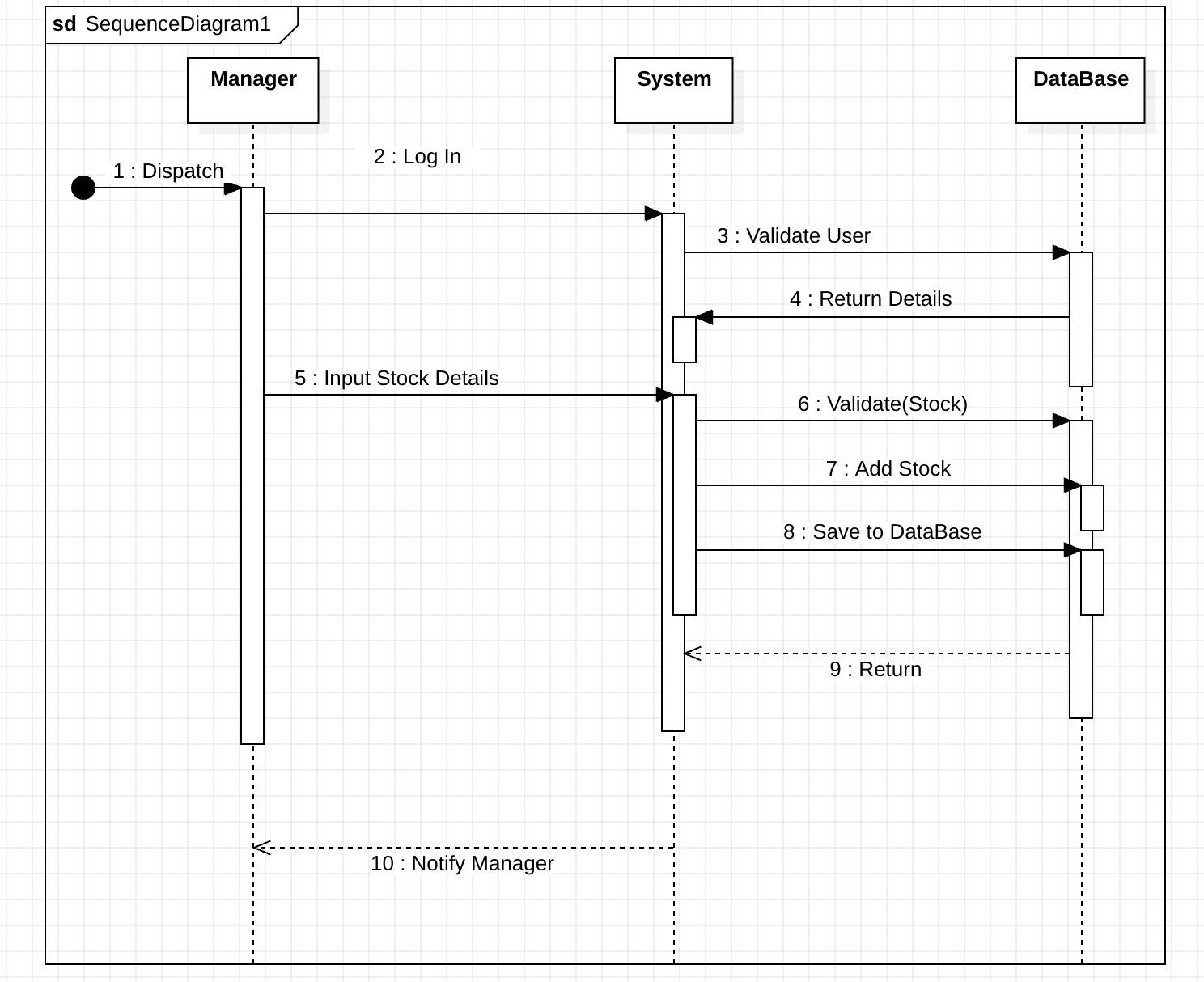
1. **State Diagram**



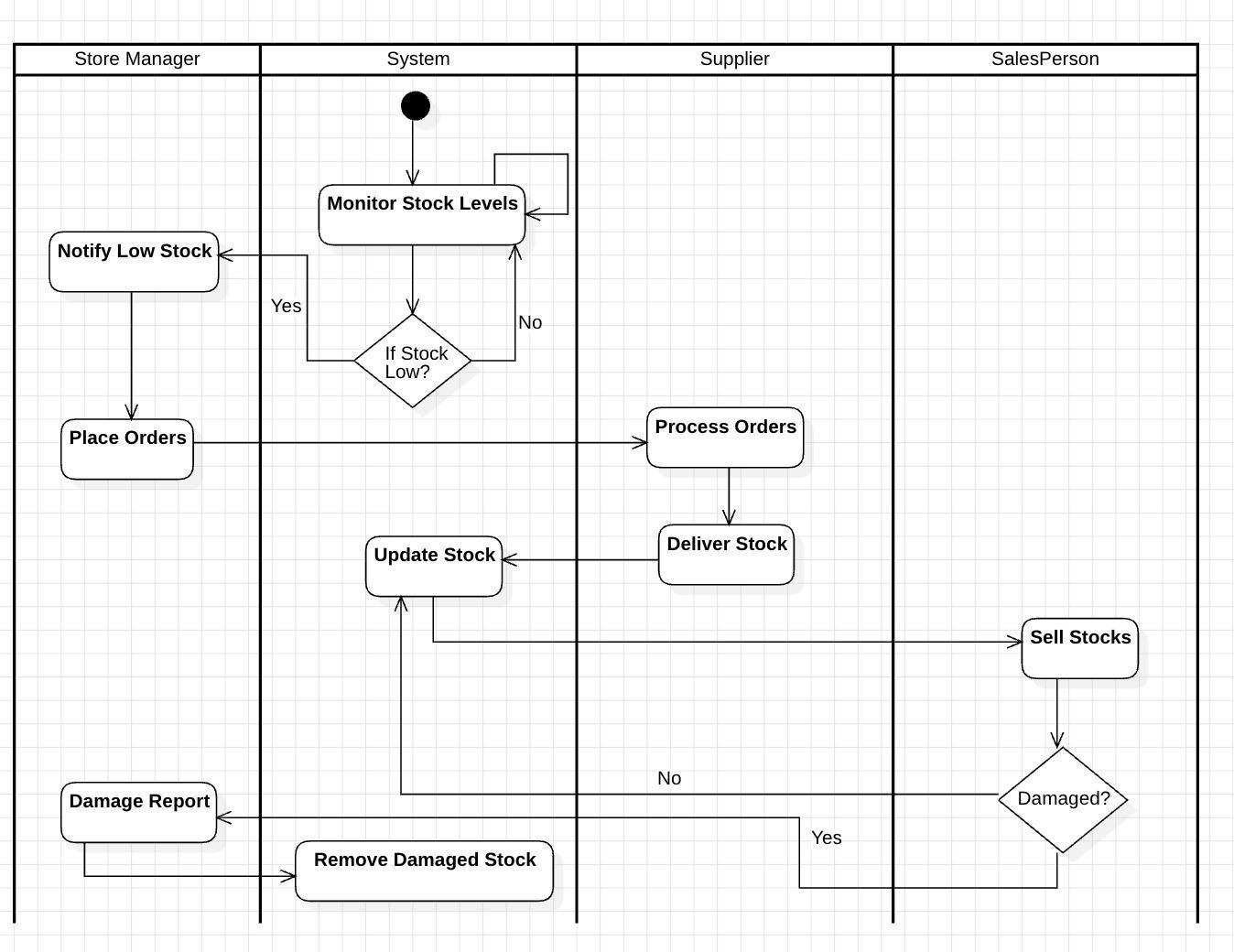
1. **Use Case Diagram**



1. **Sequence Diagram**



1. **Activity Diagram**



1. **Problem Statement**

# 4. Credit Card Management System

Design and implement a comprehensive **Credit Card Management System** for a financial institution. The system should effectively manage customer accounts, process transactions, handle credit limits, assess creditworthiness, and provide secure and efficient services. The system should utilize object-oriented principles with well-defined classes such as UserAcc, Bank, Tstatus, Client, Credit Card, Requirements, and Employee to ensure data security, maintain a robust and scalable architecture, and facilitate seamless interactions between various system components.

1. **SRS - Software Requirements Specification**

## Introduction

* + 1. Purpose

The Credit Card Management System (CCMS) is designed to simplify and automate the management of credit card-related operations such as card issuance, billing, payment processing, and fraud detection. It aims to provide a secure and efficient platform for customers, banks, and administrators to interact.

* + 1. Scope

The CCMS will support the following functionalities:

* + - * Credit card application, activation, and deactivation.
      * Transaction tracking and monthly billing statements.
      * Payment processing and overdue alerts.
      * Fraud monitoring and secure transactions.
    1. Definitions, Acronyms, and Abbreviations
       - CCMS: Credit Card Management System.
       - User: Credit cardholder or applicant.
       - Admin: Bank employee managing credit card operations.
    2. References
       - PCI DSS Compliance Standards.
       - IEEE Standard 830-1998 for SRS documentation.



## System Overview

The CCMS will be a web-based and mobile-accessible platform that includes:

* Card Management: Issuance, cancellation, and replacement.
* Transaction Module: Real-time tracking of transactions.
* Billing and Payments: Generating statements and processing payments.
* Fraud Detection: Identifying and flagging suspicious activities.
* Customer Support: Handling queries and complaints.



## Functional Requirements

* + 1. User Management
       - Customers can register, log in, and manage their profiles.
       - Admins can add, update, and remove customer accounts.
    2. Card Management
       - Apply for a new card, activate, or deactivate existing cards.
       - Set credit limits and track card usage.
       - Report lost/stolen cards and request replacements.
    3. Transaction Management
       - View real-time transaction history.
       - Categorize transactions (e.g., groceries, utilities, travel).
       - Flag suspicious transactions for fraud detection.
    4. Billing and Payments
       - Generate and email monthly statements.
       - Calculate interest and late fees for overdue payments.
       - Accept multiple payment modes (bank transfers, UPI, credit/debit cards).
    5. Fraud Detection
       - Monitor and flag unusual transaction patterns.
       - Notify users of potentially fraudulent activity.
    6. Notifications and Alerts
       - Notify users about due dates, overdue payments, and fraud alerts via SMS/email.
    7. Reporting and Analytics
       - Generate reports for card usage trends, payment histories, and fraud cases.
       - Provide insights into spending patterns for customers.



## Non-Functional Requirements

* + 1. Performance
       - Support up to 10,000 concurrent users.
       - Real-time transaction processing within 2 seconds.
    2. Scalability
       - Support multiple card types (e.g., basic, gold, platinum).
    3. Usability
       - User-friendly interface for web and mobile platforms.
    4. Security
       - Comply with PCI DSS standards for payment data security.
       - Encrypt sensitive customer and transaction data.
    5. Availability
       - Ensure 24/7 availability with 99.9% uptime.



## System Models

* + 1. Use Case Diagram Actors:
       - Customer: Apply for cards, view transactions, make payments, report fraud.
       - Admin: Manage customer accounts, generate reports, monitor fraud.

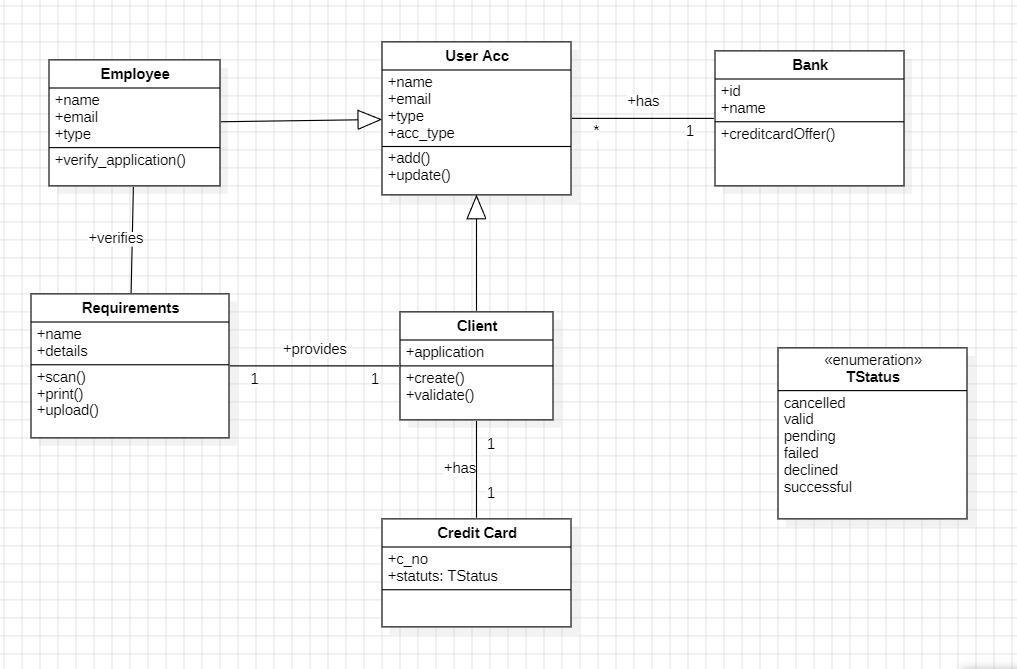


## System Requirements

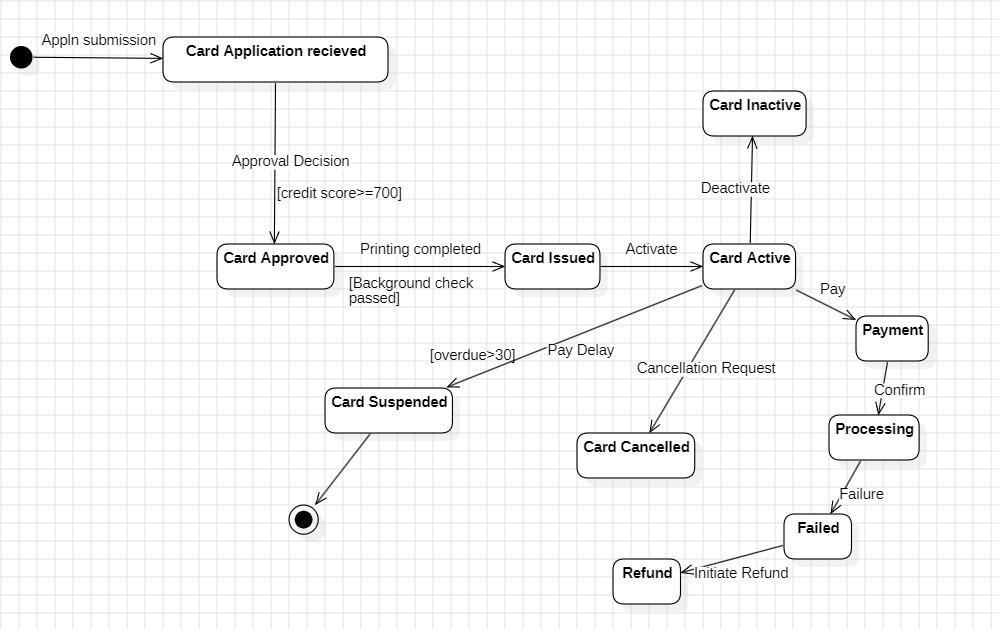
* + 1. Hardware Requirements
       - Server: Minimum 16GB RAM, 1TB SSD, quad-core processor.
       - Client Devices: Desktop/laptop with a modern browser or mobile device.
    2. Software Requirements
       - Backend: Node.js / Spring Boot / Django.
       - Frontend: React.js / Angular / Flutter.
       - Database: PostgreSQL / MySQL.
       - Payment Gateway: Razorpay / PayPal / Stripe.



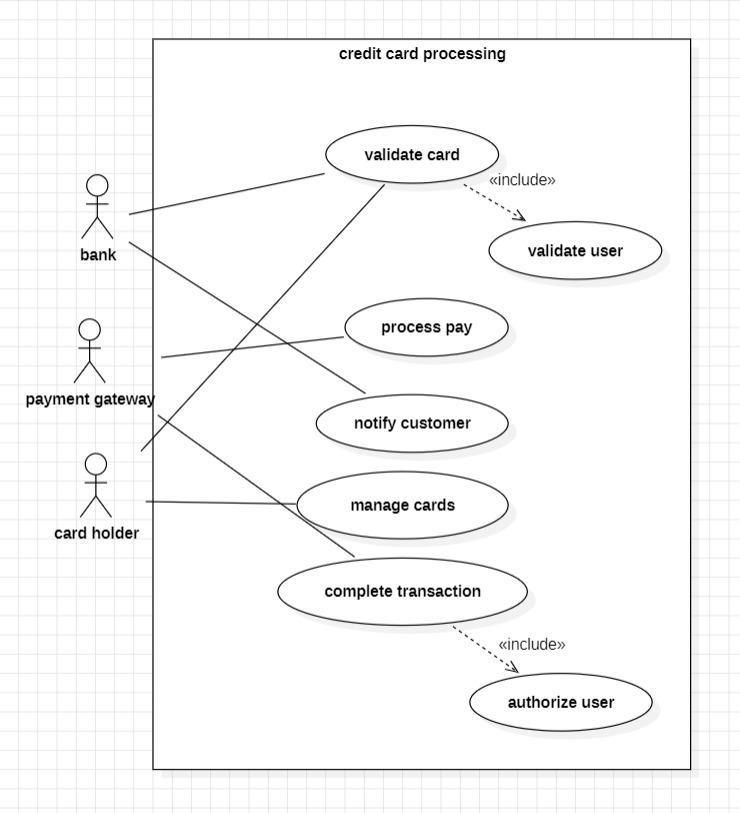
1. **Class Diagram**



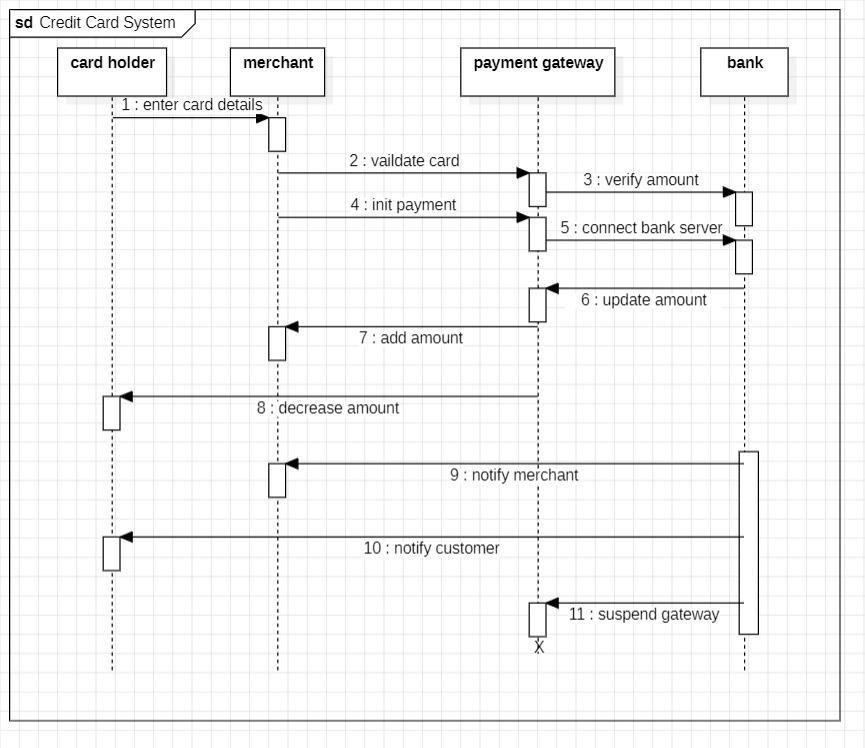
1. **State Diagram**



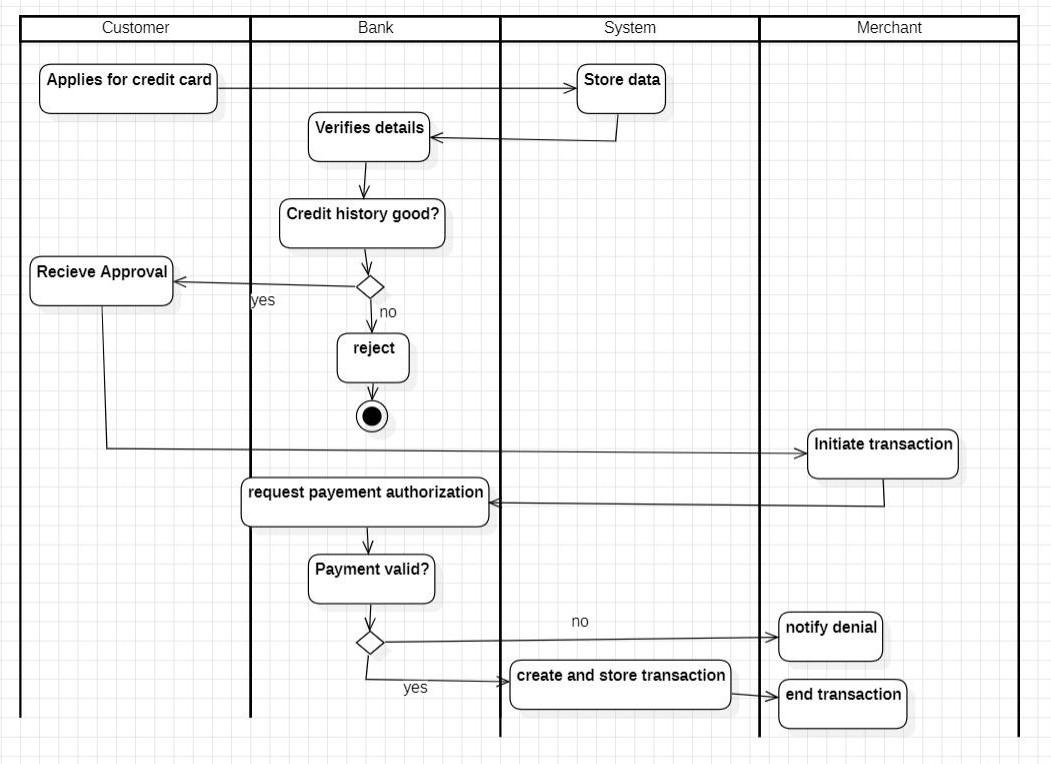
1. **Use Case Diagram**



1. **Sequence Diagram**



1. **Activity Diagram**



1. **Problem Statement**

# 5. Passport Management System

Design and implement a robust and efficient **Passport Management system** for a government agency. The system should streamline the passport application process for Applicants, facilitate efficient document verification by Admins, and ensure secure and timely issuance of Passports. The system should utilize object-oriented principles with well-defined classes such as Applicant, Admin (subclass of User), Application, AddressProof, IdProof (subclasses of Document), Passport, and Biometrics to ensure data integrity, improve operational efficiency, and enhance customer satisfaction.

1. **SRS - Software Requirements Specification**

## Introduction

* + 1. Purpose

The Passport Automation System (PAS) aims to streamline the process of applying, verifying, issuing, and renewing passports. It automates manual processes, minimizes delays, and provides a secure, transparent, and efficient way for citizens and passport authorities to interact.

* + 1. Scope

The PAS will include:

* + - * Online application and document submission.
      * Real-time application status tracking.
      * Integration with police verification and biometric systems.
      * Automated scheduling for appointments and interviews.
      * Secure storage and retrieval of applicant records.
    1. Definitions, Acronyms, and Abbreviations
       - PAS: Passport Automation System.
       - Applicant: Individual applying for a passport.
       - Authority/Admin: Passport office personnel managing the application process.
    2. References
       - Government guidelines for passport application and issuance.
       - IEEE Standard 830-1998 for SRS documentation.



## System Overview

The PAS will provide a centralized platform accessible to both applicants and authorities for managing passport-related processes. Modules include:

* Application Management: Submission and tracking.
* Document Verification: Integration with police and document verification authorities.
* Interview Scheduling: Automated appointment system.
* Passport Issuance: Generation and dispatch.



## Functional Requirements

* + 1. User Management
       - Applicants can register, log in, and manage their profiles.
       - Admins can add, update, and delete applicant records.
    2. Application Management
       - Applicants can fill out online passport application forms.
       - Upload required documents (e.g., identity proof, address proof).
       - Track the real-time status of the application.
    3. Verification and Approval
       - Integration with police for background verification.
       - Document validation by passport authorities.
       - Status updates for verification completion and approval.
    4. Biometric Data Collection
       - Capture and store biometric data (fingerprints, photographs).
       - Secure integration with biometric devices.
    5. Interview Scheduling
       - Automated slot allocation for interviews.
       - Allow applicants to reschedule within defined limits.
    6. Passport Issuance and Renewal
       - Generate unique passport numbers for new applications.
       - Issue e-passports with embedded biometric details.
       - Provide options for passport renewal.
    7. Notifications
       - Send SMS/email updates for application status, appointment reminders, and renewal alerts.
    8. Reports and Analytics
       - Generate reports on issued passports, pending applications, and police verification statistics.
       - Analyze application trends by region or time period.



## Non-Functional Requirements

* + 1. Performance
       - Handle up to 10,000 concurrent users.
       - Process application status updates in real-time (response time under 2 seconds).
    2. Scalability
       - Support additional modules for visa processing and international integration.
    3. Usability
       - Provide a user-friendly interface for all age groups.
       - Support multilingual capabilities for broader accessibility.
    4. Security
       - Encrypt sensitive data (personal details, biometric data).
       - Ensure role-based access control for different system modules.
       - Comply with GDPR and government data protection policies.



## System Models

* + 1. Use Case Diagram Actors:
       - Applicant: Apply for passports, upload documents, schedule interviews, track status.
       - Admin: Verify documents, update statuses, issue passports.
       - Police: Conduct background verification.

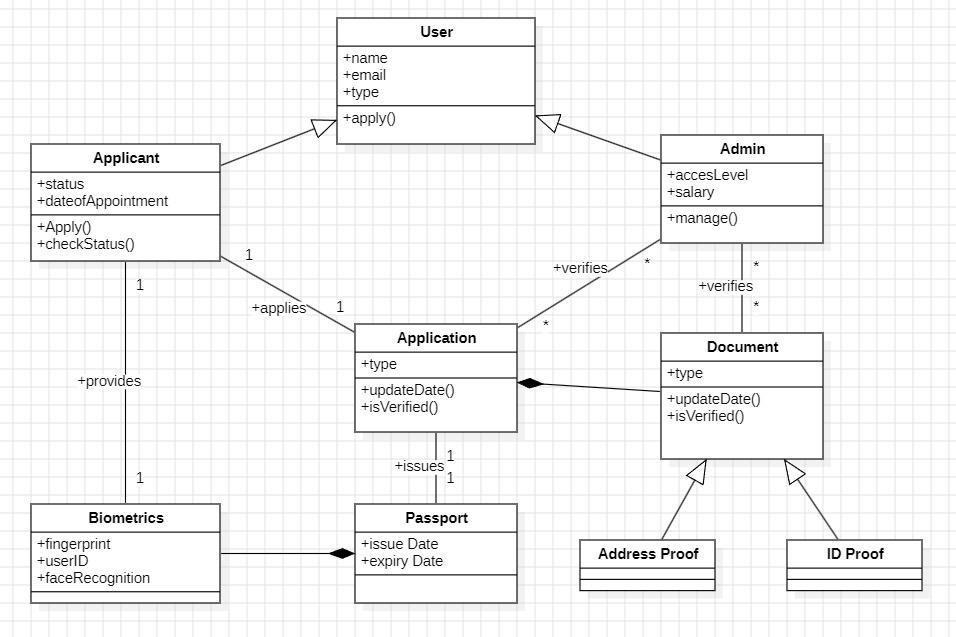


## System Requirements

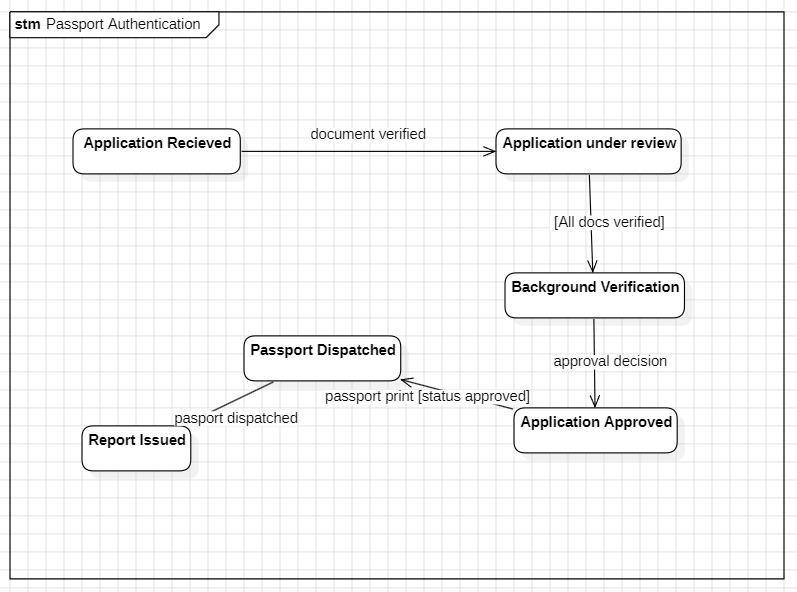
* + 1. Hardware Requirements
       - Server: Minimum 16GB RAM, 1TB SSD, quad-core processor.
       - Client Devices: Desktop/laptop with a modern browser or mobile device.
    2. Software Requirements
       - Backend: Spring Boot / Django / Node.js.
       - Frontend: Angular / React.js / Flutter.
       - Database: PostgreSQL / MySQL.
       - Biometric Integration: APIs for fingerprint and facial recognition devices.



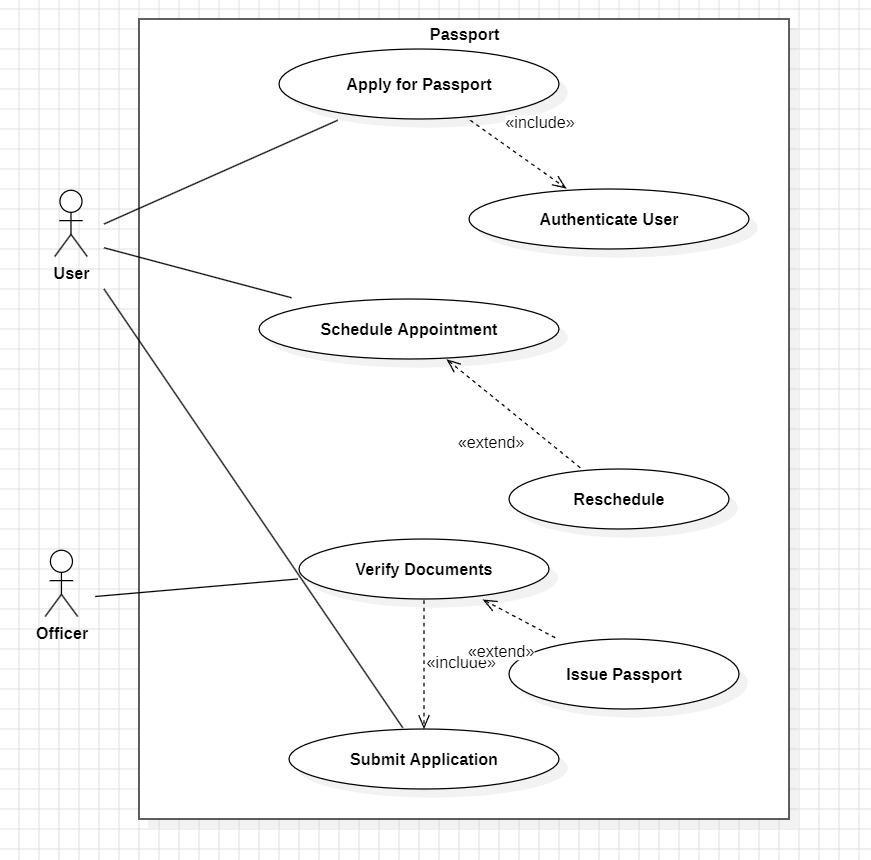
1. **Class Diagram**



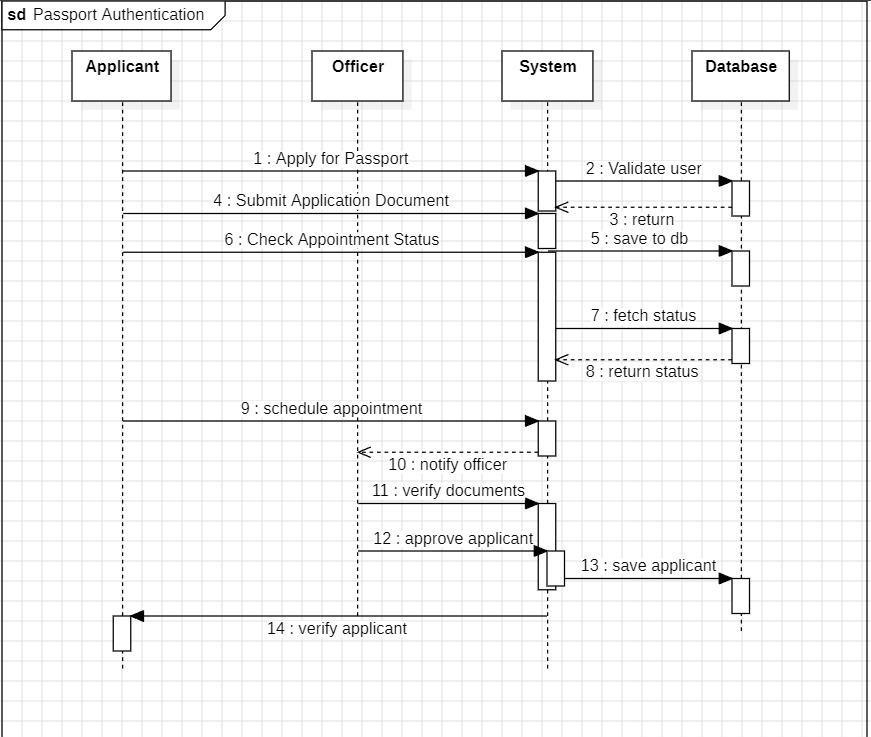
1. **State Diagram**



1. **Use Case Diagram**



1. **Sequence Diagram**



1. **Activity Diagram**

