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

Lab Record

Object-Oriented Modeling – 23CS5PCOOM

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

PRAGATHI Y RAJ

(1BM24CS415)

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

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Modeling(23CS5PCOOM) laboratory has been carried out by **Pragathi Y Raj (1BM24CS415)** during the 5th Semester August 2025-December 2025

Signature of the Faculty Incharge:

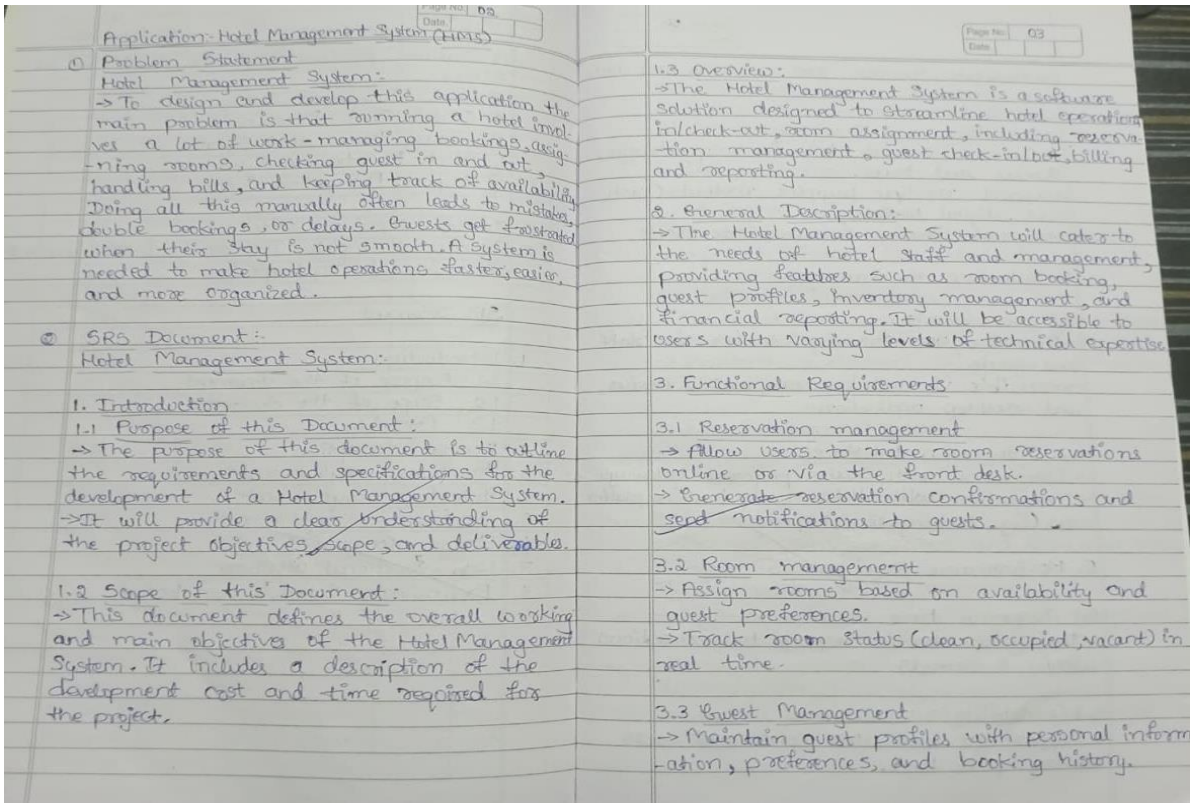
Dr. Adarsha Sagar HV
Assistant Professor

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

Table of Contents

1. Hotel Management System

Problem Statement & SRS-Software Requirements Specification:



Application- Hotel Management System (HMS)

① Problem Statement

Hotel Management System:

→ To design and develop this application the main problem is that running a hotel involves a lot of work - managing bookings, assigning rooms, checking guest in and out, handling bills, and keeping track of availability. Doing all this manually often leads to mistakes, double bookings, or delays. Guests get frustrated when their stay is not smooth. A system is needed to make hotel operations faster, easier, and more organized.

② SRS Document:

Hotel Management System:

1. Introduction

1.1 Purpose of this Document:

→ The purpose of this document is to outline the requirements and specifications for the development of a Hotel Management System.

→ It will provide a clear understanding of the project objectives, scope, and deliverables.

1.2 Scope of this Document:

→ This document defines the overall working and main objectives of the Hotel Management System. It includes a description of the development cost and time required for the project.

1.3 Overview:

→ The Hotel Management System is a software solution designed to streamline hotel operations in/check-out, room assignment, including reservation management, guest check-in/out, billing and reporting.

2. General Description:

→ The Hotel Management System will cater to the needs of hotel staff and management, providing features such as room booking, guest profiles, inventory management, and financial reporting. It will be accessible to users with varying levels of technical expertise.

3. Functional Requirements

3.1 Reservation management

→ Allow users to make room reservations online or via the front desk.

→ Generate reservation confirmations and send notifications to guests.

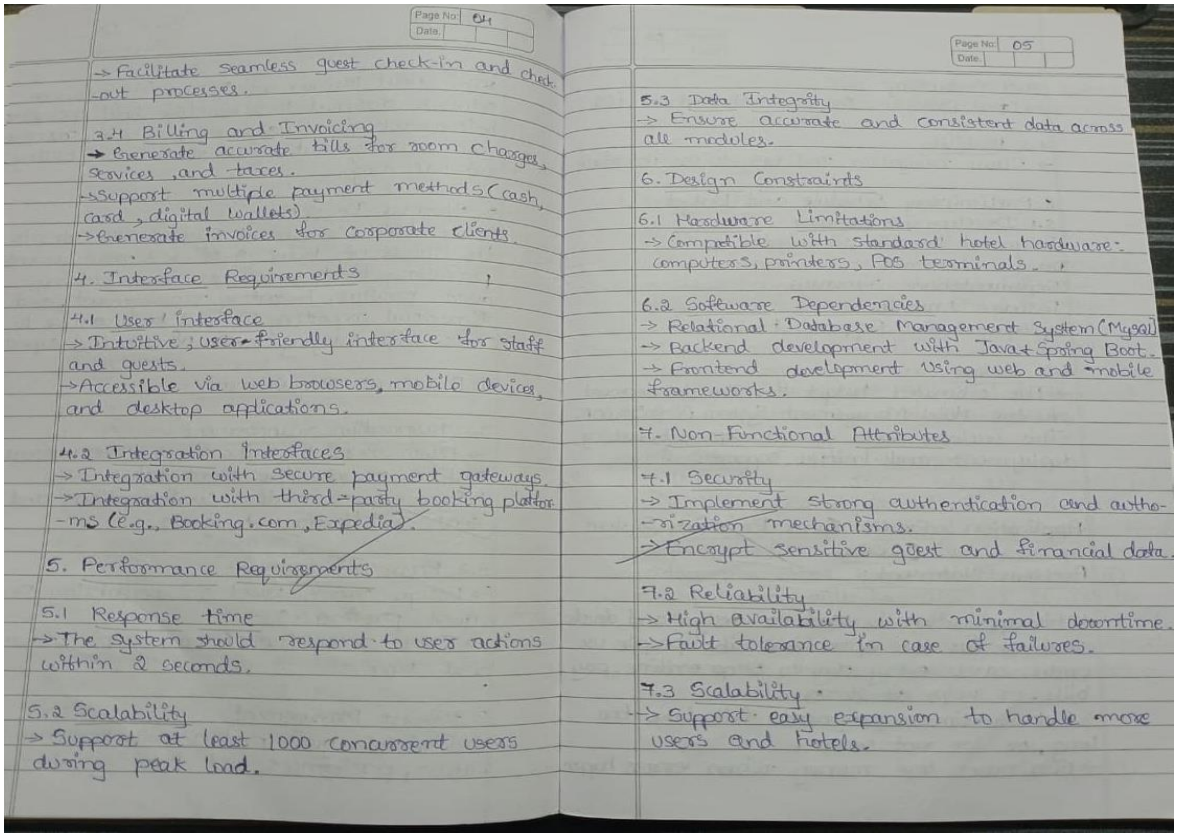
3.2 Room management

→ Assign rooms based on availability and guest preferences.

→ Track room status (clean, occupied, vacant) in real time.

3.3 Guest Management

→ Maintain guest profiles with personal information, preferences, and booking history.



→ Facilitate seamless guest check-in and check-out processes.

3.4 Billing and Invoicing

→ Generate accurate bills for room charges, services, and taxes.

→ Support multiple payment methods (cash, card, digital wallets).

→ Generate invoices for corporate clients.

4. Interface Requirements

4.1 User Interface

→ Intuitive, user-friendly interface for staff and guests.

→ Accessible via web browsers, mobile devices, and desktop applications.

4.2 Integration Interfaces

→ Integration with secure payment gateways.

→ Integration with third-party booking platforms (e.g., Booking.com, Expedia).

5. Performance Requirements

5.1 Response time

→ The system should respond to user actions within 2 seconds.

5.2 Scalability

→ Support at least 1000 concurrent users during peak load.

5.3 Data Integrity

→ Ensure accurate and consistent data across all modules.

6. Design Constraints

6.1 Hardware Limitations

→ Compatible with standard hotel hardware: computers, printers, POS terminals.

6.2 Software Dependencies

→ Relational Database Management System (MySQL)

→ Backend development with Java + Spring Boot.

→ Frontend development using web and mobile frameworks.

7. Non-Functional Attributes

7.1 Security

→ Implement strong authentication and authorization mechanisms.

→ Encrypt sensitive guest and financial data.

7.2 Reliability

→ High availability with minimal downtime.

→ Fault tolerance in case of failures.

7.3 Scalability

→ Support easy expansion to handle more users and hotels.

Page No. 06
Date.

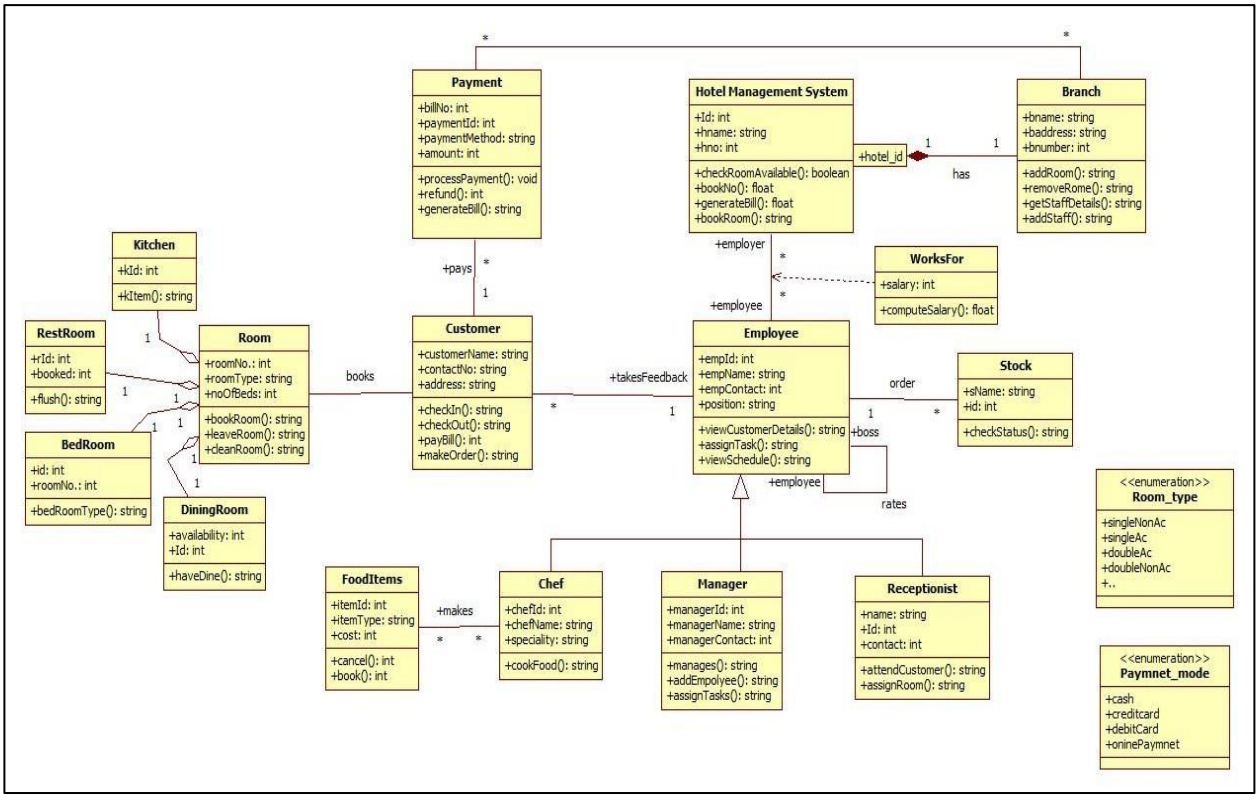
7.4 Portability
-> Cross-platform Support: web, Android, iOS.

7.5 Usability
-> Clean navigation and easy-to-use interface

8. Preliminary Schedule and Budget
8.1 Development Schedule
Estimated Duration: 6 months
Design :- 1 month
Implementation :- 3 months
Testing :- 1 month
Deployment and Support: 1 month

8.2 Budget
-> The estimated budget for the development of the Hotel Management System is $ 50,000.
-> This includes costs for development, testing, deployment, and initial support.

Class Diagram:



Overview

The class diagram represents the structure of a Hotel Management System, showing different components such as rooms, employees, customers, payments, and services, and how they interact with each other.

Hotel and Branches

The Hotel Management System contains one or more branches. Each branch manages rooms, employees, and services.

Rooms

There is a general Room class with properties like room number, type, and number of beds. Room has three specialized types:

- RestRoom
- BedRoom
- DiningRoom

Customers can book, leave, or use rooms depending on availability and requirements.

Customers

Customers check in, book rooms, order food, and make payments. They can also provide feedback after services.

Payments

Payment records include bill number, amount, and payment method.
Accepted payment modes include cash, credit card, debit card, and online payment methods.

Food and Kitchen

The system manages FoodItems, which customers can order.
Chefs prepare food based on customer orders, and the kitchen stores inventory and stock related to food preparation.

Employees

Employees work under the hotel and its branches.
There are three main types of employees:

- Manager – manages employees and hotel operations
- Chef – prepares food
- Receptionist – handles customer check-in, booking, and communication

There is also a class that helps calculate employee salary details.

Stock

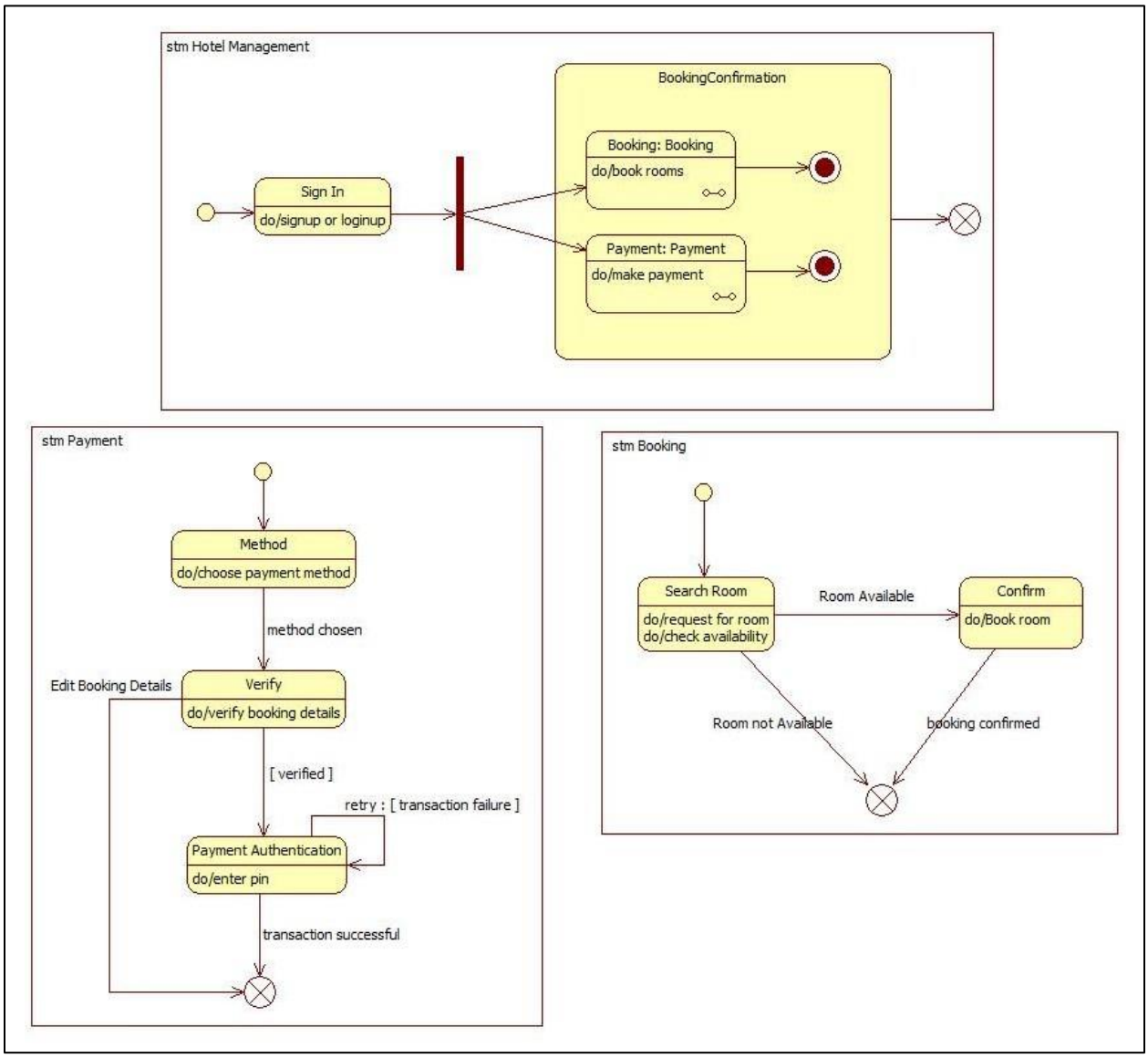
This class represents available resources and supplies required for hotel operations.
The system can check stock status and update records based on usage.

Enumerations

Two enumerations support the system:

- Room_type – defines types of rooms (single, double, AC, non-AC, etc.)
- Payment_mode – defines accepted payment types

State Diagram:



Overview

The state machine diagram represents the flow of states a user goes through during hotel booking and payment processing. It includes three main state machines:

1. Main Hotel Management State Machine
2. Booking State Machine
3. Payment State Machine

These show how the system responds to user actions such as sign-in, booking rooms, payment authentication, and confirmation.

Main State Machine: Hotel Management

- The process begins with the **Sign In** state, where the user either logs in or creates a new account.
- After signing in, the user is directed to a parallel region containing two independent processes:
    - **Booking**
    - **Payment**
- Each process must be completed before reaching the final state.
- This means users can complete booking and payment activities independently but both must finish for the booking to be confirmed.

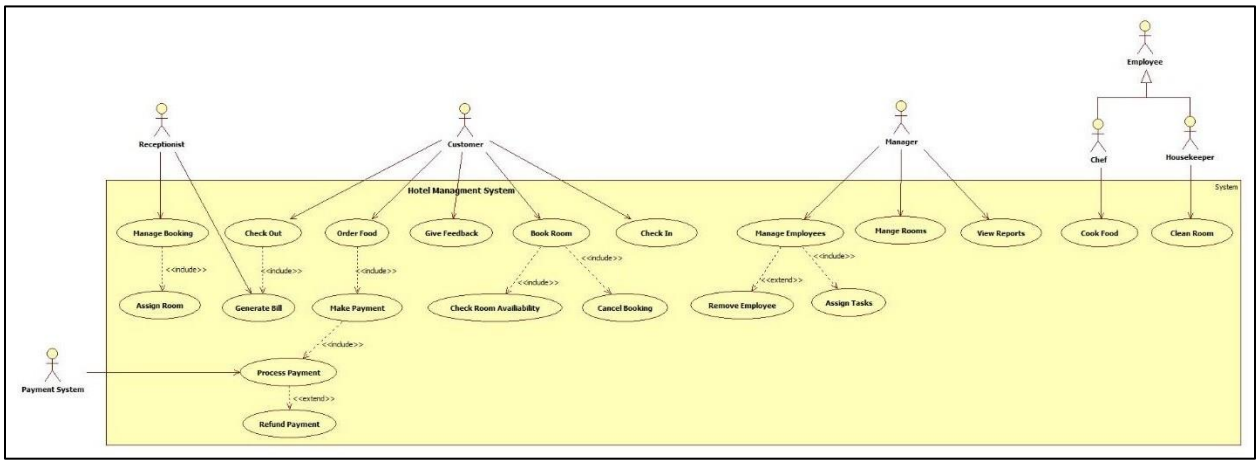
Booking Sub-State Machine

- The booking process begins with **Search Room**.
    - The system allows the user to request a room and check its availability.
- If a room is available, the system transitions to the **Confirm** state where the room is booked.
- If the room is not available, the process ends without confirmation.

Payment Sub-State Machine

- The payment process starts with **Method** where the user selects a preferred payment option.
- Next, the system moves to the **Verify** state to verify booking and payment details.
    - If the details are incorrect, the system requires editing and re-verification.
- After verification, the process goes to **Payment Authentication**, where the user enters credentials such as a PIN or OTP.
- Once authentication is successful, the payment process ends.

Use-Case Diagram:



Overview

The diagram represents a detailed use case model for the Hotel Management System. It shows the interactions between multiple system actors and the functionalities available to them. The diagram includes primary actors such as Customer, Receptionist, Manager, and Employee (with specialized roles such as Chef and Housekeeper), as well as an external Payment System.

Actors and Their Use Cases

1. Customer

A customer interacts with the system to access essential hotel services. The available use cases are:

- Book Room
- Check Room Availability (included in Book Room)
- Cancel Booking
- Check In
- Order Food
- Make Payment (includes Process Payment)
- Give Feedback
- Check Out (includes Generate Bill)

The customer is a primary user capable of initiating both service-related and transactional activities.

2. Receptionist

The receptionist manages front-desk operations. Their use cases include:

- Manage Booking

    - Includes Assign Room
- Check Out
    - Includes Generate Bill

The receptionist facilitates customer interactions and administrative booking processes.

3. Manager

The manager oversees hotel operations and administrative controls. Their use cases are:

- Manage Employees
    - Includes Assign Tasks
    - Extends Remove Employee
- Manage Rooms
- View Reports

The manager ensures smooth functioning of hotel staff and resources.

4. Employee

Employee is a general actor category with two specialized roles:

- Chef
    - Cook Food
- Housekeeper
    - Clean Room

These employees perform operational services within the hotel.

5. Payment System (External Actor)

The payment system supports financial transactions initiated by customers. It interacts with:

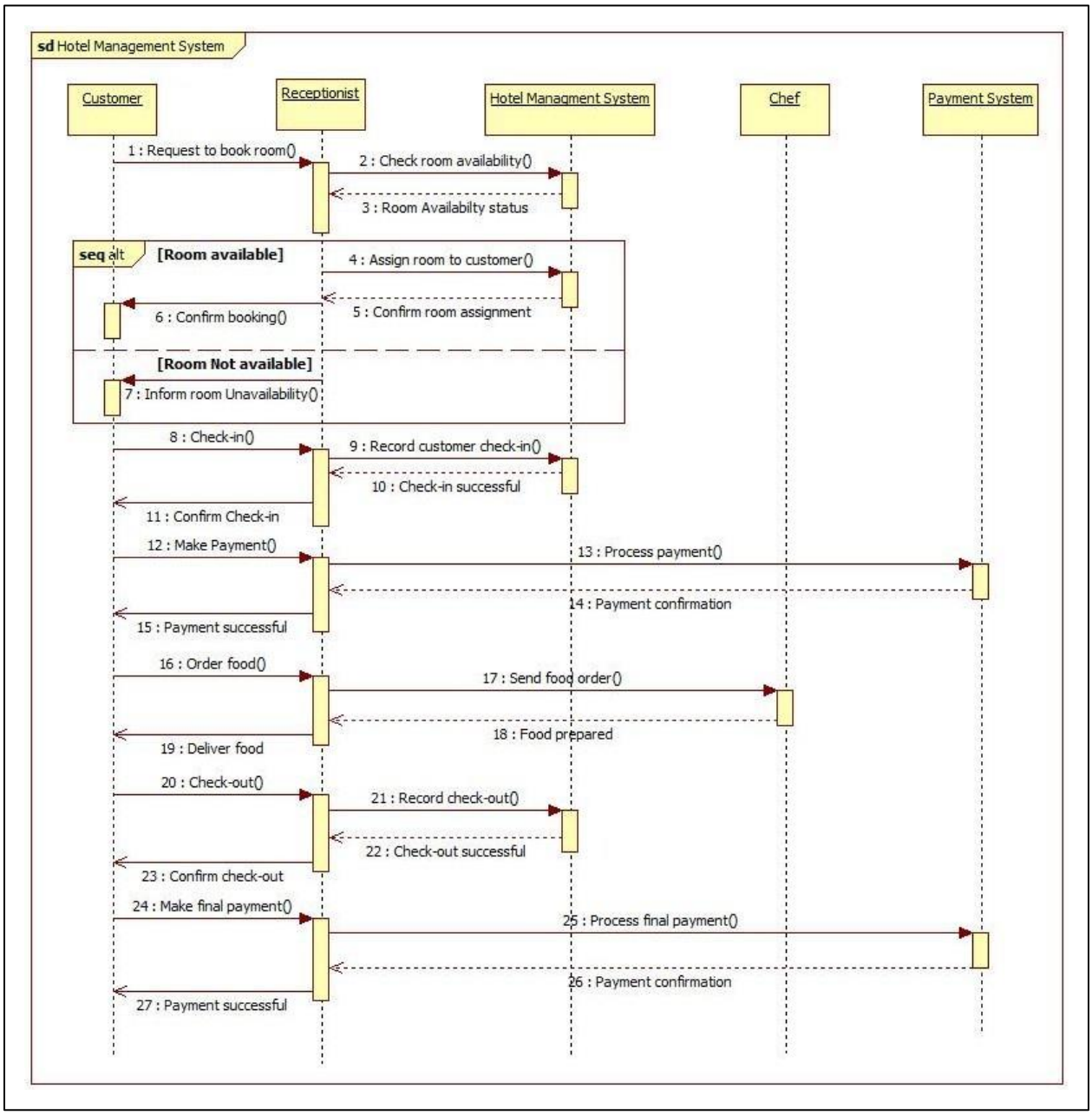
- Process Payment
    - Includes Refund Payment (optional extension)

This actor ensures secure and verified transaction processing.

System Behavior and Relationships

- Use case dependencies such as include and extend relationships are used to show reusable or conditional processes.
- Core tasks like Assign Room, Generate Bill, and Process Payment are shared among related use cases, improving modularity and clarity.
- Customer-facing actions are supported by backend operational use cases handled by employees, receptionists, and the manager.

Sequence Diagram:



Overview

This sequence diagram represents the flow of interactions between different entities during a hotel service process. The actors involved are:

- Customer
- Receptionist
- Hotel Management System
- Chef

- External Payment System

The diagram outlines the steps from booking a room, checking in, making payments, ordering food, checking out, and completing the final payment.

Step-by-Step Interaction Description

1. The process begins when the **Customer sends a request to book a room** to the Receptionist.
2. The Receptionist sends a request to the **Hotel Management System** to check room availability.
3. The system responds with the availability status.

Room Availability Condition

- **If the room is available:**
    - The receptionist assigns the room to the customer.
    - The system confirms the assignment.
    - The customer receives confirmation.
- **If the room is not available:**
    - The receptionist informs the customer about unavailability.
    - The process ends there.

Check-In Process

4. The customer proceeds with check-in.
5. The receptionist records check-in details.
6. The system confirms the successful check-in.

Initial Payment

7. The customer makes a payment request.
8. The hotel system sends the payment request to the **Payment System**.
9. The Payment System confirms the successful transaction.
10. The system informs the customer that payment was successful.

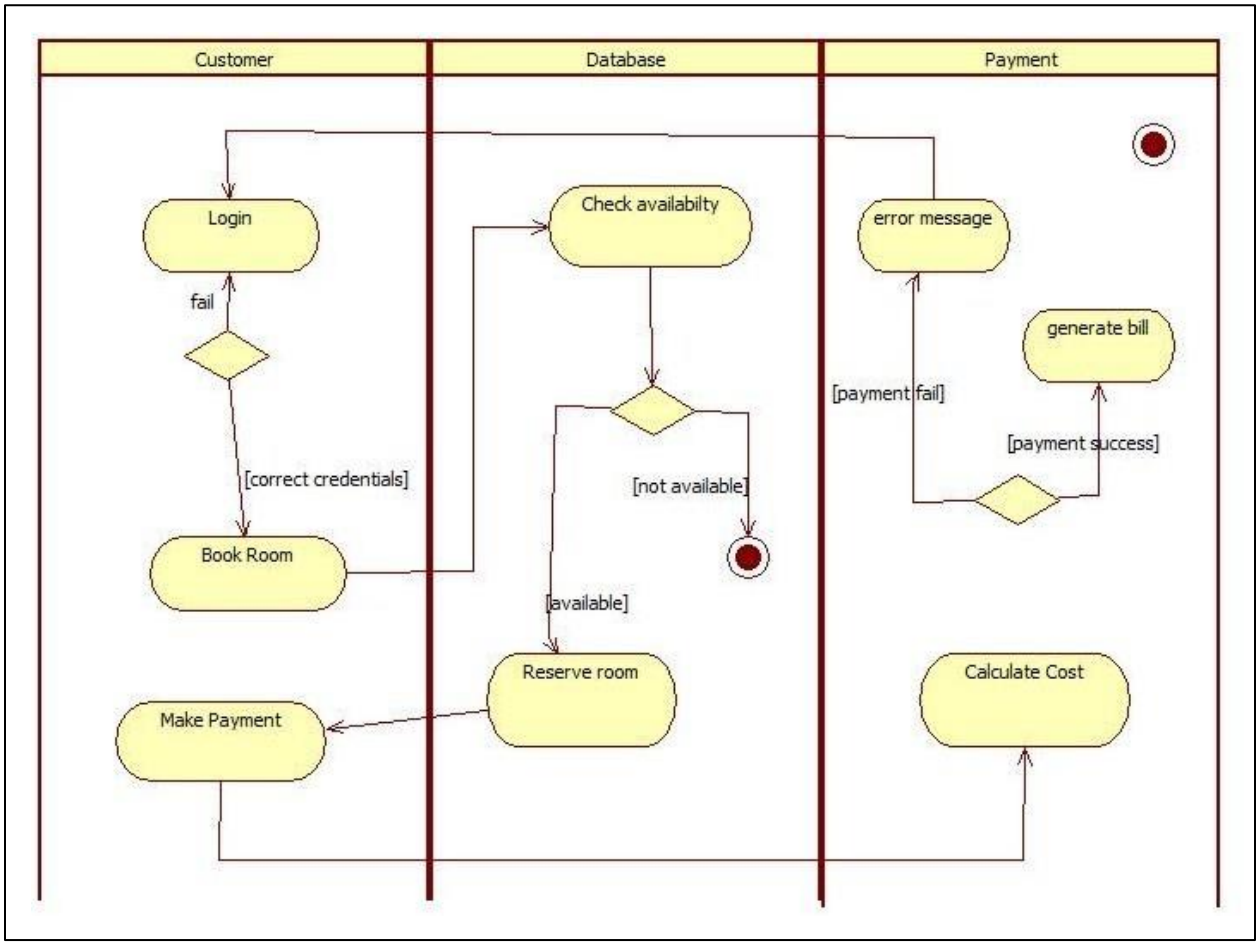
Food Ordering Process

11. The customer places a food order.
12. The hotel system forwards the order to the **Chef**.
13. The chef prepares the food and marks the task complete.

Check-Out Process and Final Payment

14. The customer requests check-out.
15. The receptionist records check-out details.
16. The system confirms check-out completion.
17. The customer makes the **final payment**.
18. The system forwards the request to the Payment System.
19. The payment system confirms the final payment.
20. The system informs the customer that payment was successful.

Activity Diagram:



Overview

This activity diagram represents the workflow for booking a room and making a payment in a hotel management system. The process involves three main components:

- Customer

- Database
- Payment System

Each swimlane shows which part of the system is responsible for each activity.

Workflow Description

1. **Login Process**
    - The customer begins by logging into the system.
    - A decision is made:
        - If credentials are incorrect, the flow returns to retry login.
        - If credentials are correct, the process continues to booking.
2. **Room Booking Process**
    - The customer attempts to book a room.
    - The system sends a request to the database to check room availability.
    - A decision point verifies availability:
        - If the room is **not available**, the process ends.
        - If the room is **available**, the database reserves the room.
3. **Payment Process**
    - After reservation, the customer proceeds to make payment.
    - The payment system calculates the final cost.
    - A decision is made on payment validation:
        - If **payment fails**, an error message is shown and the user may retry.
        - If **payment succeeds**, the system generates a bill.

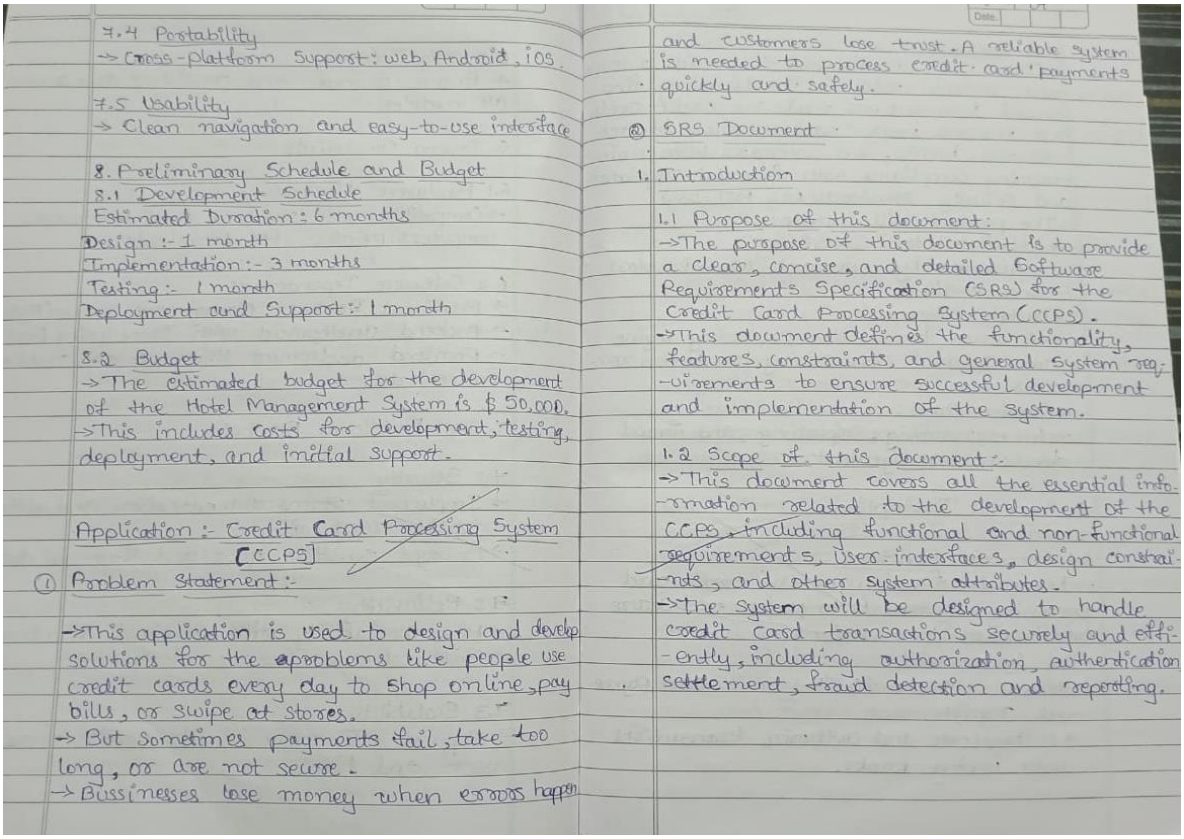
Final Outcome

The process ends after successfully generating the bill or if the room is unavailable. The flow ensures:

- Authentication before booking
- Room availability validation
- Secure payment processing
- Proper termination based on actions taken

2. Credit Card Processing System

Problem Statement & SRS-Software Requirements Specification:



7.4 Portability
→ Cross-platform support: web, Android, iOS.

7.5 Usability
→ Clean navigation and easy-to-use interface

8. Preliminary Schedule and Budget
8.1 Development Schedule
Estimated Duration: 6 months
Design :- 1 month
Implementation :- 3 months
Testing :- 1 month
Deployment and Support :- 1 month

8.2 Budget
→ The estimated budget for the development of the Hotel Management System is $ 50,000.
→ This includes costs for development, testing, deployment, and initial support.

Application :- Credit Card Processing System [CCPS]

① Problem Statement :-

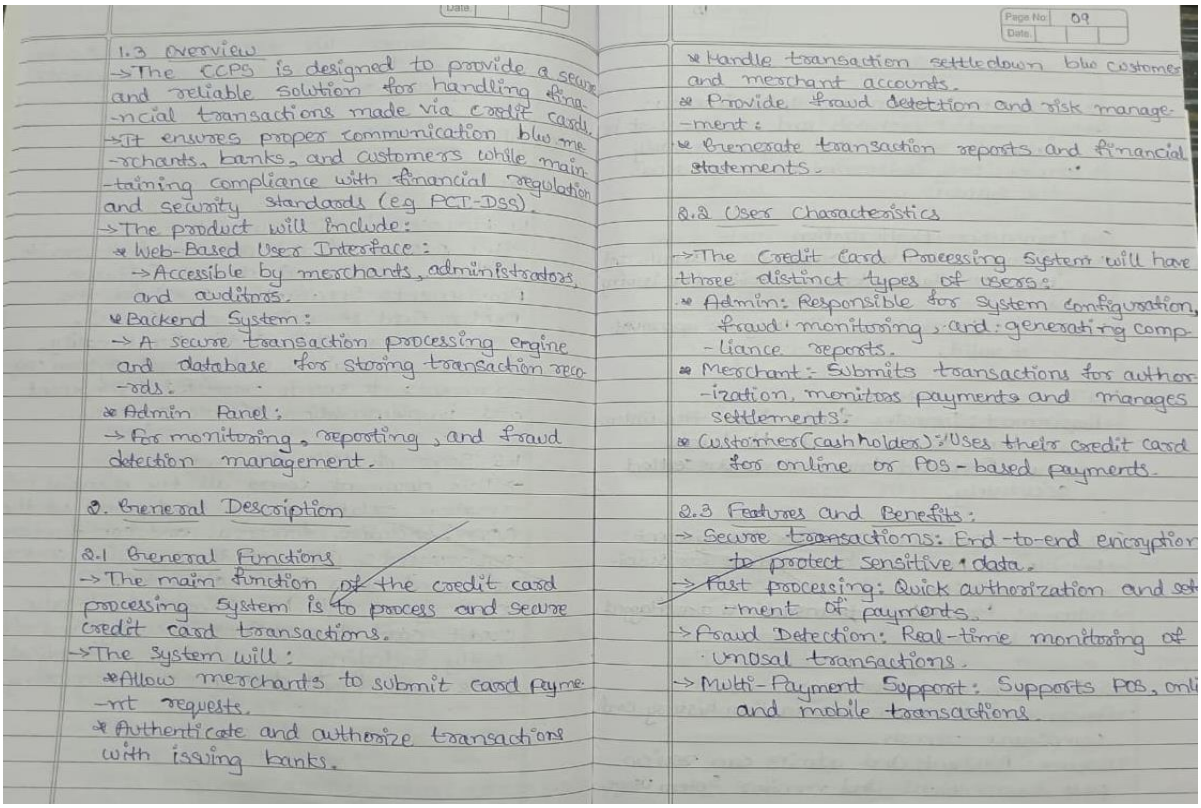
→ This application is used to design and develop solutions for the problems like people use credit cards every day to shop online, pay bills, or swipe at stores.
→ But sometimes payments fail, take too long, or are not secure.
→ Bussinesses lose money when errors happen and customers lose trust. A reliable system is needed to process credit card payments quickly and safely.

② SRS Document

1. Introduction

1.1 Purpose of this document:
→ The purpose of this document is to provide a clear, concise, and detailed Software Requirements Specification (SRS) for the Credit Card Processing System (CCPS).
→ This document defines the functionality, features, constraints, and general system requirements to ensure successful development and implementation of the system.

1.2 Scope of this document :-
→ This document covers all the essential information related to the development of the CCPS, including functional and non-functional requirements, users interfaces, design constraints, and other system attributes.
→ The system will be designed to handle credit card transactions securely and efficiently, including authorization, authentication settlement, fraud detection and reporting.



1.3 Overview
→ The CCPS is designed to provide a secure and reliable solution for handling financial transactions made via credit cards.
→ It ensures proper communication b/w merchants, banks, and customers while maintaining compliance with financial regulation and security standards (eg PCI-DSS).
→ The product will include:
* Web-Based User Interface :
→ Accessible by merchants, administrators, and auditors.
* Backend System :
→ A secure transaction processing engine and database for storing transaction records.
* Admin Panel :
→ For monitoring, reporting, and fraud detection management.

2. General Description

2.1 General Functions
→ The main function of the credit card processing system is to process and secure credit card transactions.
→ The System will :
* Allow merchants to submit card payment requests.
* Authenticate and authorize transactions with issuing banks.
* Handle transaction settledown b/w customers and merchant accounts.
* Provide fraud detection and risk management :
* Generate transaction reports and financial statements.

2.2 User Characteristics

→ The Credit Card Processing System will have three distinct types of users :
* Admin : Responsible for system configuration, fraud monitoring, and generating compliance reports.
* Merchant : Submits transactions for authorization, monitors payments and manages settlements.
* Customers (Cash holders) : Uses their credit card for online or POS-based payments.

2.3 Features and Benefits :
→ Secure transactions : End-to-end encryption to protect sensitive data.
→ Fast processing : Quick authorization and settlement of payments.
→ Fraud Detection : Real-time monitoring of unusual transactions.
→ Multi-Payment Support : Supports POS, online and mobile transactions.

3. Functional Requirements:

3.1 User Registration and Authentication:
→ Requirement: Merchants and admins must be able to register and log in securely.
→ Outcome: Only authorized users can access the system.

3.2 Transaction Authorization
→ Requirement: Validate credit card details and check available balance with the issuing bank.
→ Outcome: Transactions will only be approved if valid.

3.3 Transaction Settlement
→ Requirement: Transfer funds from the customer's bank to the merchant's bank.
→ Outcome: Completed transactions are settled accurately.

3.4 Fraud Detection and Risk Management:
→ Requirement: Monitor transactions for suspicious activities.
→ Outcome: Fraudulent transactions are flagged or blocked.

3.5 Reporting and Auditing:
→ Requirement: Generate transaction history and compliance reports.
→ Outcome: Mechants and admins can review past transactions and monitor system usage.

4. Interface Requirements:

4.1 Software Interfaces:
→ Database: Relational database for storing transaction records securely.
→ Admin Interface: Web-based admin panel for fraud monitoring and report generation.
→ Merchant Interface: Web/mobile dashboard for transaction tracking.

4.2 Communication Interfaces:
→ Web interface: Transactions and dashboards accessed via HTTPS protocols.
→ Data Exchange: REST APIs using JSON/XML for bank and merchant communication.

5. Performance Requirements:

5.1 System Response time:
→ Authorization responses within 2-3 seconds.

5.2 Scalability:
→ The system should handle 5000 concurrent transactions during peak loads.

5.3 Database performance
→ It should quickly query execute for transaction lookup, fraud checks, and reporting.

6. Design Constraints:

6.1 Technology Constraints
→ The system must be developed using secure, stable frameworks such as Java or Python.

6.2 Hardware Constraints
→ It should operate on secure servers with encryption modules, at least 16GB RAM and 8TB storage.

7. Non-functional Attributes:

7.1 Security:
→ All sensitive data must be encrypted.

7.2 Reliability
→ The system must ensure 99.9% uptime.

7.3 Scalability
→ Must support expansion to handle international transactions.

7.4 Usability
→ Easy-to-use dashboards for merchants and admins.

8. Preliminary Schedule and Budget

8.1 Development Schedule: 10 months
→ Design: 2 months
→ Implementation: 4 months
→ Testing: 2 months
→ Deployment and support: 2 months

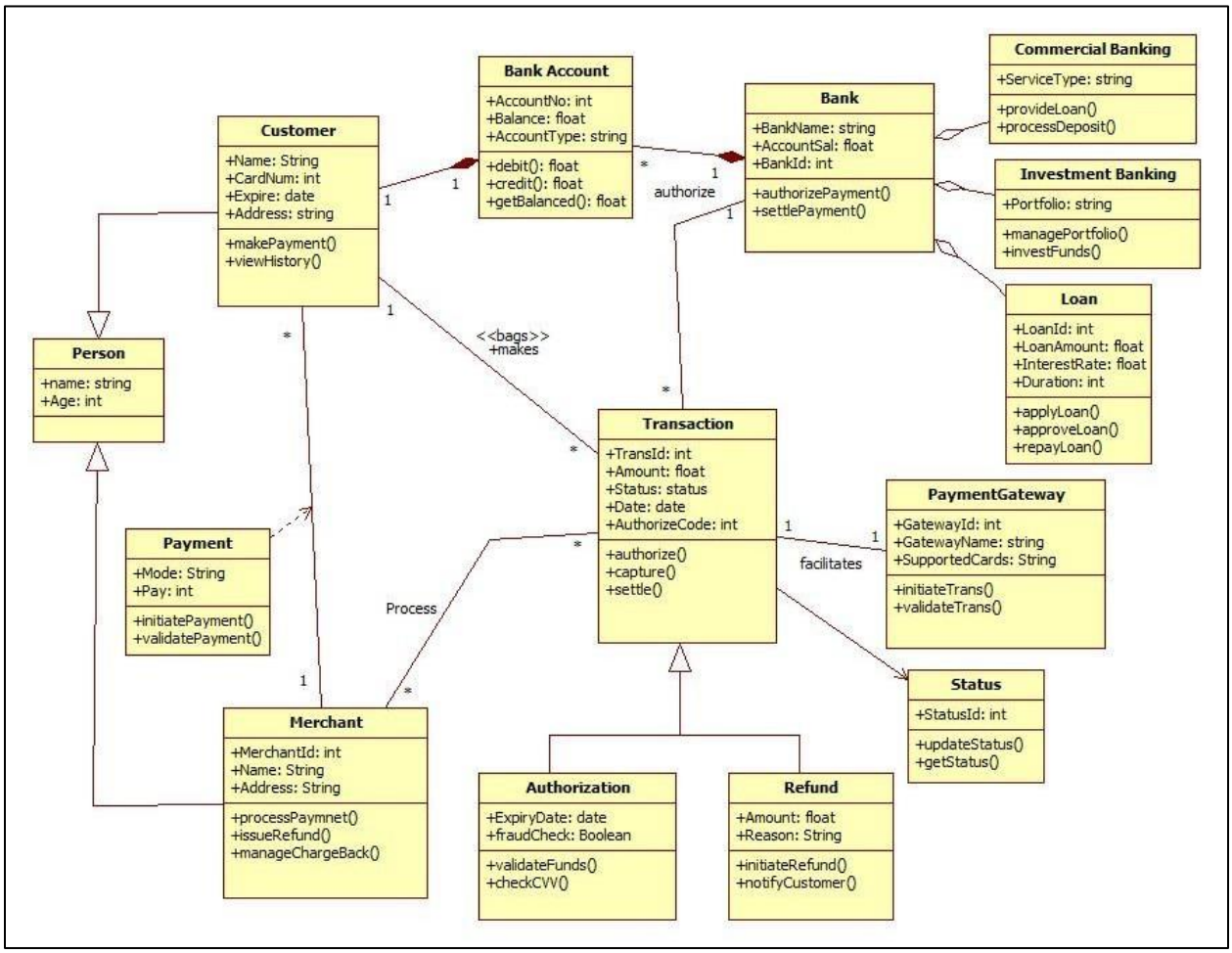
8.2 Budget
→ The estimated budget for the development of the CCPS is $500,000.
→ This includes costs for development, security, certification, testing, deployment, and maintenance.

Application: Library Management System (LMS)

① Problem Statement:

→ This application is used to design and develop solutions for the problems like libraries often struggle with keeping track of books and users.
→ If things are done manually, books get misplaced, fines are hard to calculate, and searching for a book takes a lot of time.
→ Students and librarians both face difficulties.
→ A system is needed to make borrowing returning and managing books simple and fast.

Class Diagram:



Overview

This class diagram represents the structure of a banking and payment system. It defines the main entities involved in processing payments, authorizing transactions, managing bank accounts, and handling refunds or loans. The relationships between these classes show how customers, banks, merchants, and payment gateways interact during financial operations.

Key Classes and Their Roles

1. Person

- Represents a general human entity with basic attributes.
- The **Customer** class inherits from Person.

2. Customer

- Includes attributes like card number, expiry date, and address.
- Main responsibilities:
    - Make payments
    - View transaction history

3. Bank Account

- Stores financial details such as account number, balance, and type.
- Supports operations:
    - Debit
    - Credit
    - Get balance

Each customer is linked to one bank account.

4. Bank

- Stores details like name, total amount, and bank ID.
- Responsible for authorizing and initiating payments.

The bank authorizes both transactions and loans.

5. Merchant

- Represents a business receiving money.
- Capable of:
    - Processing payments
    - Issuing refunds
    - Handling chargeback disputes

A merchant can be associated with multiple transactions.

6. Payment

- Defines payment details such as mode and amount.
- Supports initiating and validating payment.

7. Transaction

- Central class representing a financial action.
- Attributes include transaction ID, amount, status, date, and authorization code.
- Methods include authorize, capture, and settle.

Customers and merchants interact through transactions.

8. Authorization

- Contains verification details, including expiry date and fraud check.
- Validates funds and security values.

9. Refund

- Manages refund requests when a customer disputes or cancels a transaction.

- Includes reason and refund amount.

10. Payment Gateway

- Acts as a bridge between the merchant and bank.
- Supports initiating and validating transactions.
- Also enforces card compatibility rules.

11. Status

- Tracks the state of a transaction, such as approved, pending, or failed.
- Supports updating and retrieving status values.

12. Loan and Related Banking Services

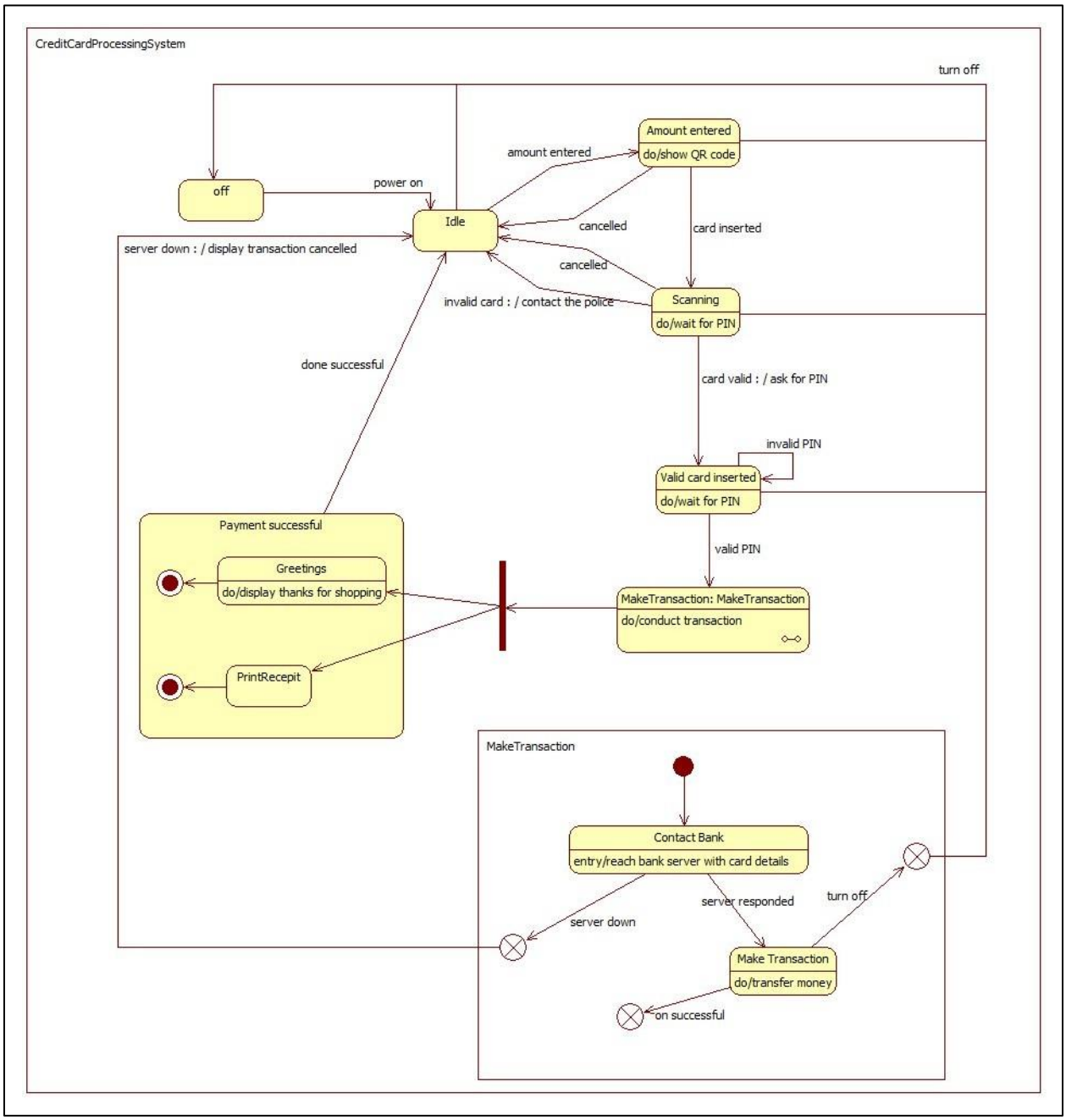
- The **Loan** class supports applying, approving, and repaying loans.
- Specialized banking branches:
    - Commercial Banking
    - Investment Banking

These provide services like deposits, portfolio management, and funding.

Relationships

- **Customer → Bank Account:** One-to-one
- **Customer → Transaction:** One-to-many
- **Merchant → Transaction:** One-to-many
- **Bank → Transaction:** Authorizes and processes multiple transactions
- **Payment Gateway → Transaction:** Facilitates and validates multiple transactions
- **Transaction → Authorization / Refund:** Optional relationships

State Diagram:



Overview

This state machine diagram represents the behavior of a credit card processing system from powering on to completing or cancelling a transaction. It shows various system states, transitions based on user actions or system responses, and final outcomes such as successful payment or system shutdown.

State Flow Description

1. Off State

- The system begins in an **Off** state.
- When powered on, it transitions to the **Idle** state.

2. Idle State

- In this state, the system waits for input.
- Possible actions include:
    - Card inserted
    - Amount entered
    - Cancel action
    - System shutdown
- If an **invalid card** is detected, an alert is displayed and the system returns to idle.
- If cancelled, the system returns to idle.

3. Scanning State

- Triggered when a card is inserted.
- System waits for PIN entry.
- Possible outcomes:
    - **Invalid PIN:** return to idle
    - **Valid PIN:** move to next state

4. Valid Card Inserted State

- The card has been validated and the system again waits for PIN verification.
- If the PIN is valid, the system proceeds to the transaction phase.

5. Make Transaction Region

This is a composite state containing two subprocesses:

- **Contact Bank**
    - The system sends transaction details to the bank server.
    - If the server is down, the transaction is cancelled.
    - If the server responds, the system continues.
- **Process Transaction**

- Funds are transferred.
- If successful, the system exits this state and proceeds to final actions.

6. Payment Successful State

- Once the transaction succeeds, the system displays a thank-you message and optionally prints a receipt.
- After completion, the system returns to the **Idle** state unless powered off.

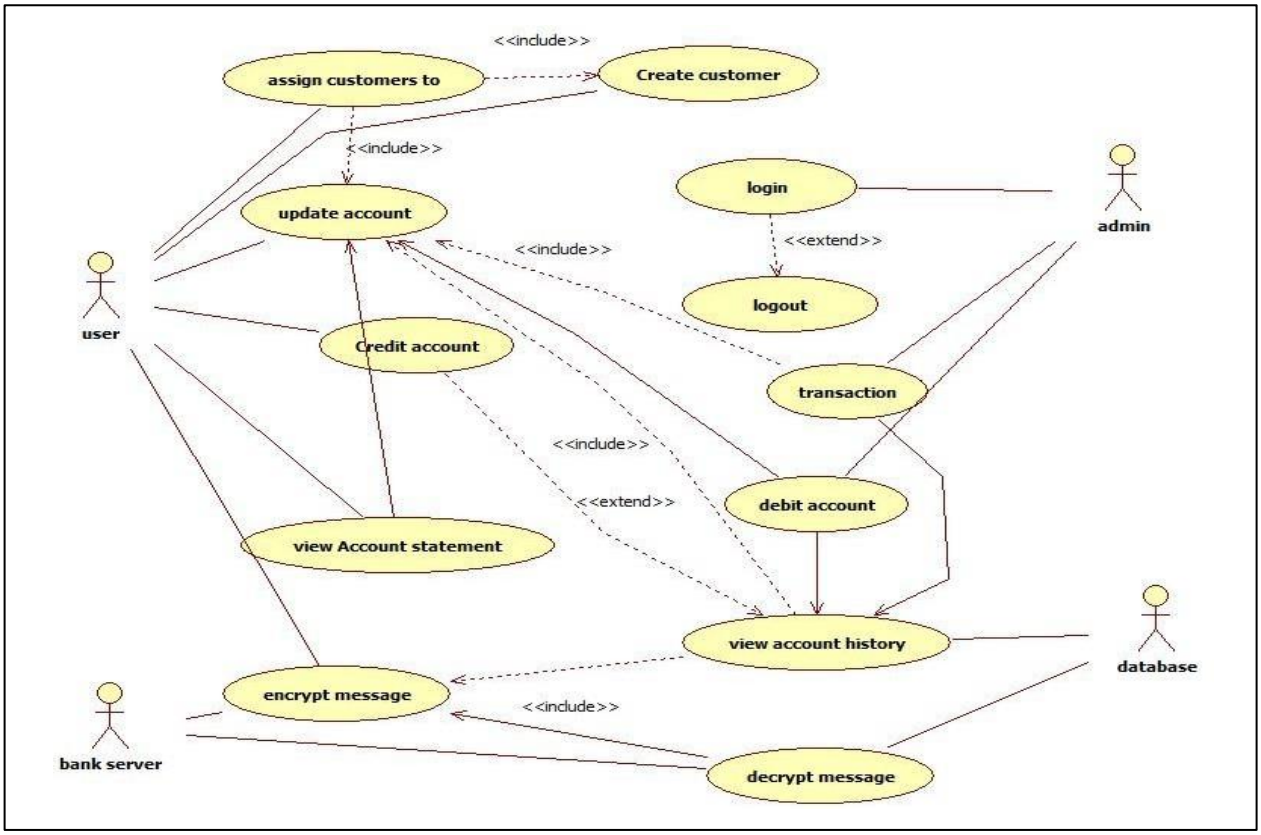
Error and Termination Paths

- If at any stage:
  - Server fails,
  - Card is invalid,
  - PIN is incorrect,
  - User cancels the operation,

The system returns to **Idle** or ends the process appropriately.

- The system may be turned off at any state, which leads back to the **Off** state.

Use-Case Diagram:



Overview

This use case diagram represents the functions of a banking system. It identifies the interactions between different actors and the system. The diagram includes multiple user roles such as:

- User
- Admin
- Database
- Bank Server

Each actor interacts with specific system functions based on their authority and purpose.

Actors and Their Use Cases

1. User

The user is the primary actor who performs standard account operations. The use cases available to the user include:

- Login

- Logout
- Credit Account
- Debit Account
- View Account History
- View Account Statement
- Update Account
- Transaction

Some of these use cases are interconnected:

- **Transaction** includes both **Credit Account** and **Debit Account**.
- **Update Account** includes assigning customers to account functions.

2. Admin

The admin manages bank user details and system maintenance tasks. Their available use cases include:

- Create Customer (includes Assign Customers)
- Login
- Logout

Admins have access to higher-level functions than regular users.

3. Bank Server

The Bank Server is responsible for message security and data handling. It supports:

- Encrypt Message
- Decrypt Message

These use cases are included in other relevant processes to ensure secure communication, especially during transactions and accessing account history.

4. Database

The database actor stores and retrieves required data. It interacts with:

- View Account History
- Decrypt Message

The database ensures stored information is accessible and secure.

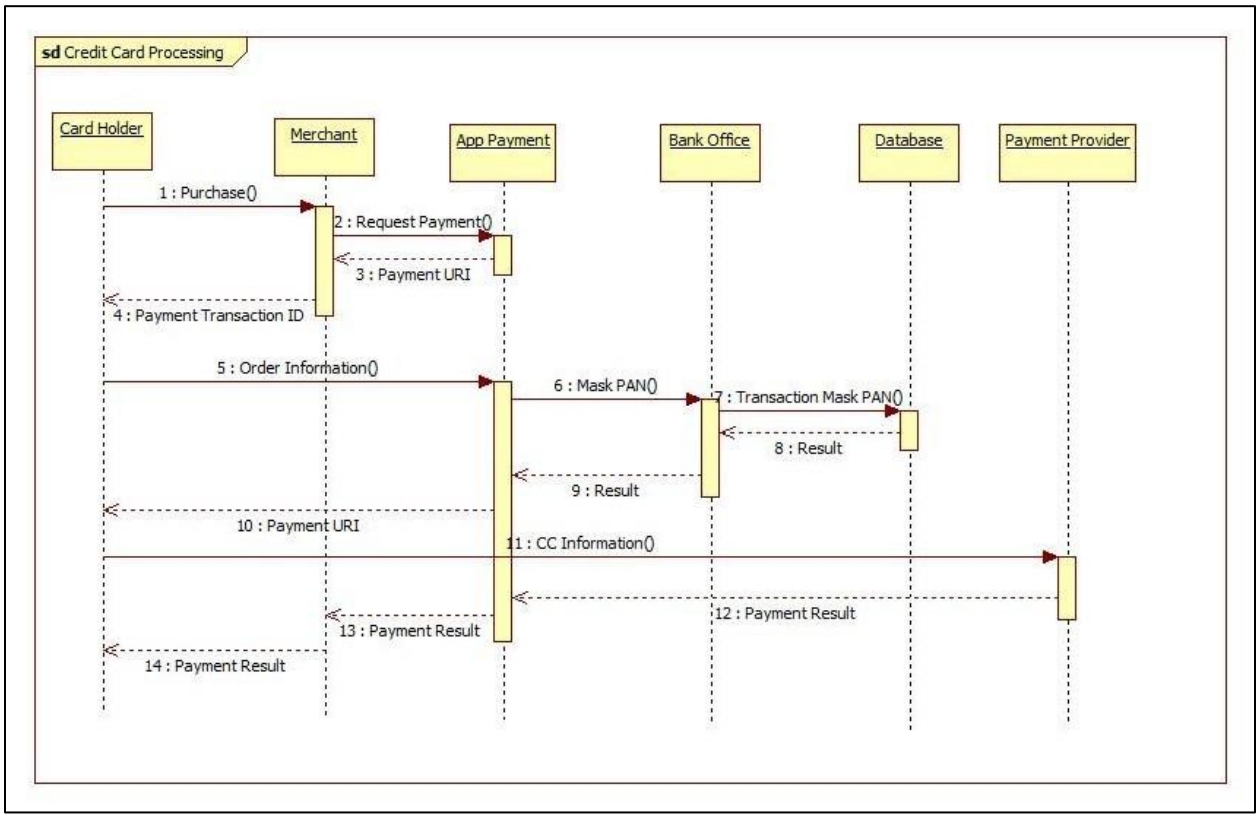
Dependency Relationships

The diagram includes:

- **Include relationships**, showing reusable actions (e.g., credit and debit as part of a transaction).
- **Extend relationships**, representing optional or conditional behavior (e.g., logout extending login).

These dependencies simplify the model and avoid redundant definitions.

Sequence Diagram:



Overview

This sequence diagram represents the flow of credit card processing during a purchase transaction. Multiple actors and systems work together to complete the payment securely. These include:

- Card Holder
- Merchant
- App Payment System
- Bank Office
- Database
- Payment Provider

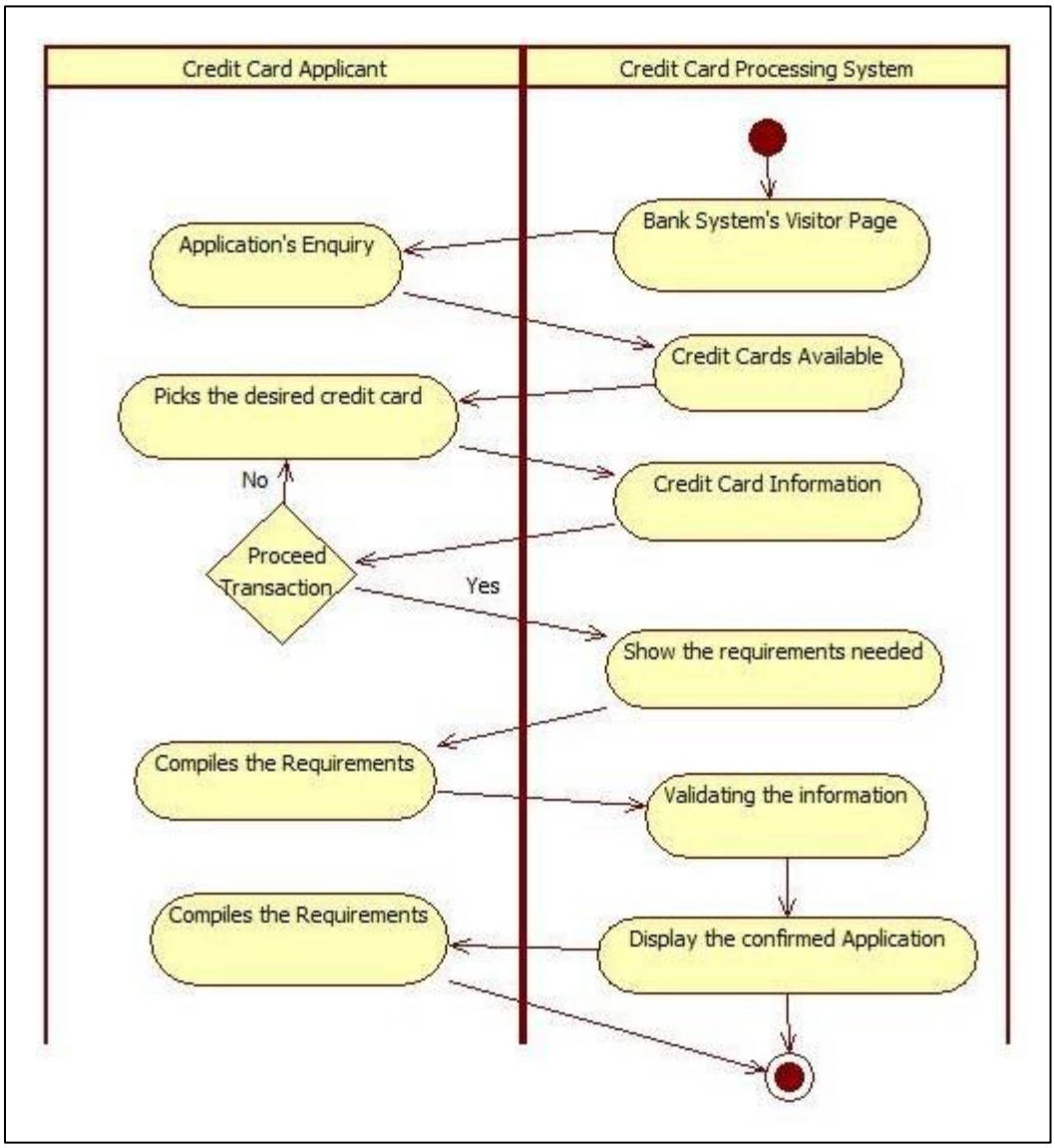
Step-by-Step Interaction Description

1. **Purchase Initiated**
    - The Card Holder initiates a purchase request.
    - The Merchant receives this request.
2. **Payment Request Handling**
    - The Merchant sends a *Request Payment* message to the App Payment System.
    - The App Payment System responds with a *Payment URI*.
    - The Merchant sends the *Payment Transaction ID* back to the Card Holder.
3. **Order Information Transfer**
    - The Merchant forwards the order details to the App Payment System.
4. **Masking Sensitive Information**
    - The App Payment System sends a request to the Bank Office to mask the PAN (Primary Account Number).
    - The Bank Office sends a masking request to the Database.
    - The Database returns the result to the Bank Office.
    - The Bank Office returns the masking result to the App Payment System.
5. **Customer Payment Processing**
    - The App Payment System sends masked credit card information to the Payment Provider for transaction approval.
6. **Transaction Result Handling**
    - The Payment Provider sends the payment result back to the App Payment System.
    - The App Payment System forwards the result to the Merchant.
    - The Merchant sends the result to the Card Holder.

Final Outcome

The process ends with the Card Holder receiving a payment success or failure message.

Activity Diagram:



Overview

The activity diagram illustrates the workflow involved in applying for a credit card. It shows the interaction between two main entities:

- Credit Card Applicant
- Credit Card Processing System

The process flows from initial inquiry to final confirmation of the application.

Workflow Description

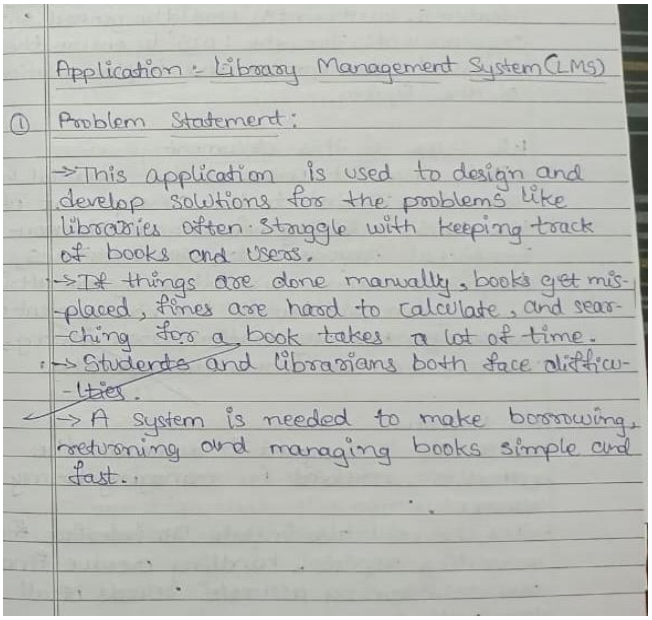
1. **Application Inquiry**
    - The applicant initiates the process by making an inquiry.

- The system responds by directing the applicant to the bank's visitor page.

2. **Viewing Available Credit Cards**
    - The system displays available credit card options.
    - The applicant selects a preferred card from the available list.
3. **Viewing Credit Card Details**
    - The system shows detailed information about the selected card (such as benefits, requirements, eligibility, fees).
4. **Decision Point: Proceed Transaction**
    - The applicant decides whether to continue with the application.
        - If the applicant chooses **No**, the process stops here.
        - If the applicant chooses **Yes**, the process continues.
5. **Requirement Submission**
    - The system displays the necessary application requirements.
    - The applicant collects and submits these required documents or information.
6. **Validation of Information**
    - The processing system validates the submitted documents and checks eligibility criteria.
    - If everything is correct, the application is accepted.
7. **Final Confirmation**
    - The system displays the confirmed and successfully processed application.

3. Library Management System

Problem Statement & SRS-Software Requirements Specification:



Application :- Library Management System (LMS)

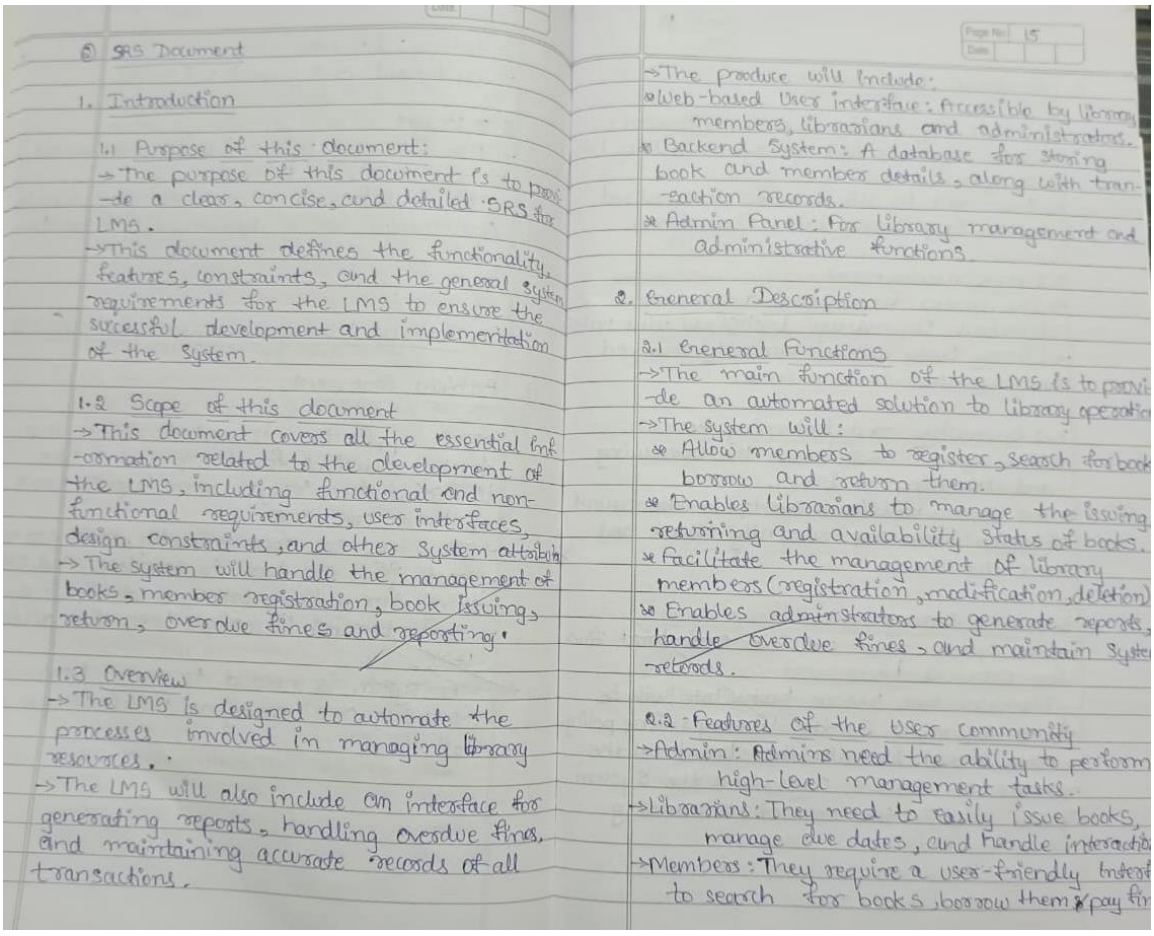
① Problem Statement :

→ This application is used to design and develop solutions for the problems like libraries often struggle with keeping track of books and users.

→ If things are done manually, books get misplaced, fines are hard to calculate, and searching for a book takes a lot of time.

→ Students and librarians both face difficulties.

→ A system is needed to make borrowing, returning and managing books simple and fast.



② SRS Document

1. Introduction

1.1 Purpose of this document:

→ The purpose of this document is to provide a clear, concise, and detailed SRS for LMS.

→ This document defines the functionality, features, constraints, and the general system requirements for the LMS to ensure the successful development and implementation of the system.

1.2 Scope of this document

→ This document covers all the essential information related to the development of the LMS, including functional and non-functional requirements, user interfaces, design constraints, and other system attributes.

→ The system will handle the management of books, member registration, book issuing, return, overdue fines and reporting.

1.3 Overview

→ The LMS is designed to automate the processes involved in managing library resources.

→ The LMS will also include an interface for generating reports, handling overdue fines, and maintaining accurate records of all transactions.

→ The produce will include:

* Web-based User interface: Accessible by library members, librarians and administrators.
* Backend System: A database for storing book and member details, along with transaction records.
* Admin Panel: For library management and administrative functions.

2. General Description

2.1 General Functions

→ The main function of the LMS is to provide an automated solution to library operations.

→ The system will:

* Allow members to register, search for books, borrow and return them.
* Enables librarians to manage the issuing, returning and availability status of books.
* Facilitate the management of library members (registration, modification, deletion).
* Enables administrators to generate reports, handle overdue fines, and maintain system records.

2.2 Features of the User community

→ Admin: Admins need the ability to perform high-level management tasks.

→ Librarians: They need to easily issue books, manage due dates, and handle interactions.

→ Members: They require a user-friendly interface to search for books, borrow them & pay fines.

Date

3. Functional Requirements:

3.1 User Registration and Authentication:
-> User must be able to register and log into the system.
-> Users will be able to access personalized services (e.g. borrow books, check fines)

3.2 Book management
-> The system must allow admins and librarians to manage books (add, delete, upda
-> Library books will be correctly managed and updated in real-time.

3.3 Borrowing and Returning Books:
->The system must track the borrowing and returning of books.
->The library will have an up-to-date record of which books are currently issued and their due dates.

3.4 Fine Management
->The system must calculate and track fine for overdues books.
-> Members who return books late will be fined according to the library's fine policy.
-> Fines should be calculated based on the number of overdue days, and the system should notify the user of outstanding fines.

Page No. 17
Date

4. Interface Requirements

4.1 Software Interfaces:
->Database: The system must interface with a relational database to store and retrieve data related to books, members & transactions
->Admin Interface: Admins will interact with the system via web-based interface to manage users, books, and generate reports.
->Members Interface: Members will interact with the system via a web interface to search for books, borrow books, check fines, and manage their accounts.

4.2 Communication Interfaces:
->Web Interface: The slw should communicate through HTTP/HTTPS protocols.
-> Data Stream: Data exchange blw the front-ted and backend should be done using REST API's (JSON Format).

5. Performance Requirements

5.1 System Respone time
-> The System should return search results for books within 2 sec.

5.2 Scalability.
->The system should be able to handle up to 1000 concurrent users without degradation in performance.

Page No. 18
Date

5.3 Database Performance
->The database should support quick query execution for tasks, like book search, user lookup, and transaction recording.

6. Design Constraints

6.1 Technology constraints
->The system must be developed using the latest stable versions of web technologies like HTML, CSS, JavaScript and Python.

6.2 Hardware constraints:
->The system must operate on standard hardware, which includes servers with at least 8GB of RAM and 1TB of storage for initial deployment.

7. Non-Functional Attributes

7.1 Security
->User data including login credentials must be encrypted by using modern encryption techniques.

7.2 Reliability
->The system should have 99.9% uptime by ensuring availability during peak library usage itemes.

7.3 Scalability
->The system should scale to handle increased usage.

Page No. 19
Date

8. Preliminary Schedule and Budget:

8.1 Development Schedule: 6 months
->Design : 1 month
->Implementation: 3 months
->Testing : 1 month
->Deployment and support : 1 month

8.2 Budget
->The estimated budget for the development of the LMS is $80,000.
->This includes costs for development, testing, deployment, and initial support.

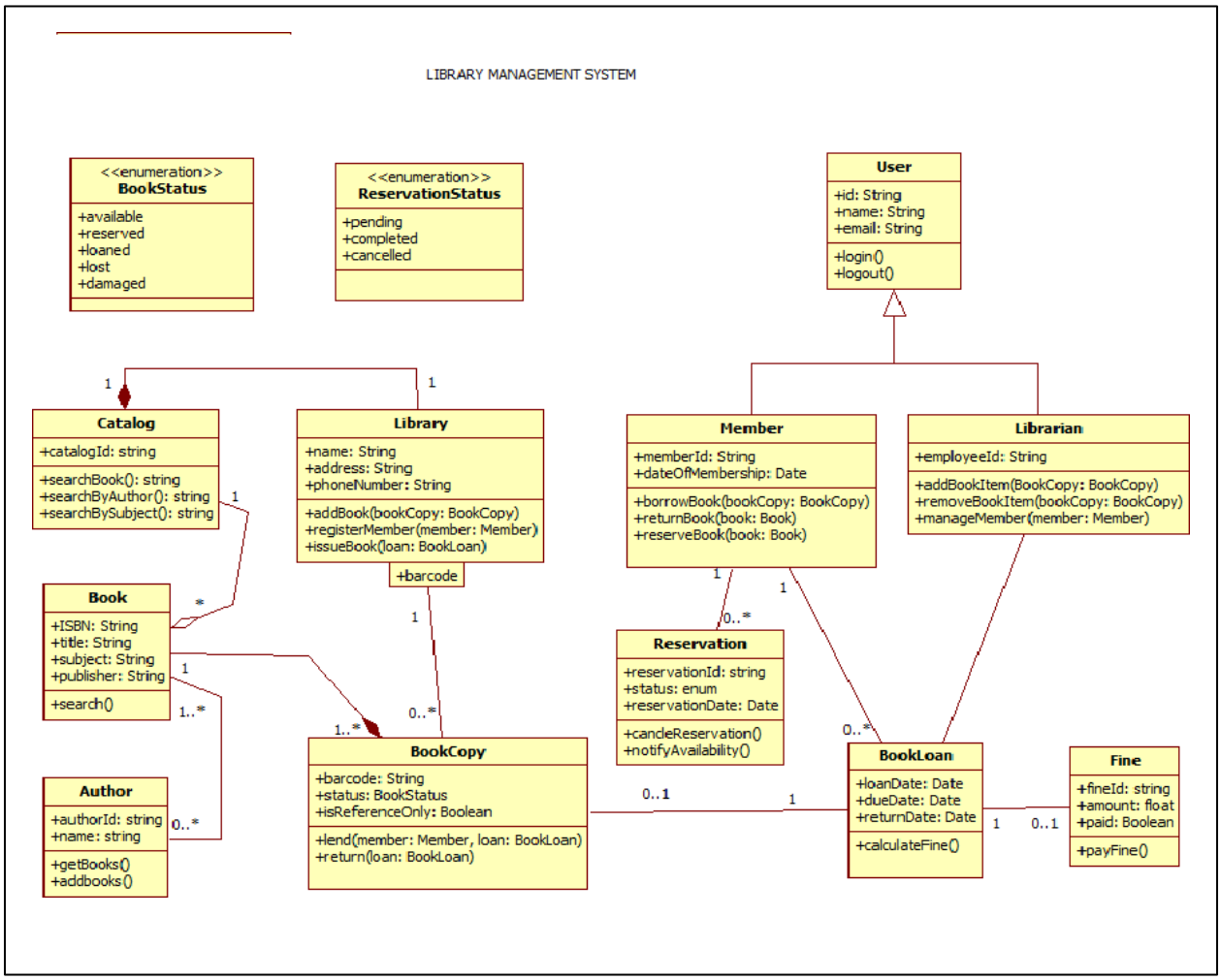
WEEK-02

Application:- Stock Maintenance System (SMS)

① Problem Statement:

->This application is used to design and develop solutions for the problem's like shops and warehouses need to know how much stock they have at any moment.
->If stock is managed manually, it can lead to missing items, overstocking, or expired goods.
->This causes losses and customer dissatisfactio
->A system is needed that can update stock levels automatically and give real-time alerts.

Class Diagram:



Overview

This class diagram represents the structure of a Library Management System including users, books, borrowing transactions, catalog services, and fines. It shows how different components interact to support book management, membership management, reservations, and loan operations.

Major Classes and Their Roles

1. User

- A general class representing any system user.
- Contains basic information such as id, name, and email.
- Supports login and logout functions.

Two specialized classes inherit from User:

- **Member**
- **Librarian**

2. Member

- Represents a library member who can borrow, return, and reserve books.
- Attributes:
    - memberId
    - dateOfMembership
- Main operations:
    - borrowBook()
    - returnBook()
    - reserveBook()

3. Librarian

- Represents a library staff member.
- Responsible for managing library operations.
- Key operations:
    - addBookItem()
    - removeBookItem()
    - manageMember()

Book-Related Structure

4. Catalog

- Used for searching and organizing books.
- Supports searching by title, author, or subject.

5. Book

- Represents a book title, not a physical copy.
- Attributes include ISBN, title, subject, and publisher.

6. Author

- Represents book authors.
- A single author can write multiple books.

7. BookCopy

- Represents physical copies of a book.
- Contains barcode and book status (from enumeration).
- Operations:
    - lend()

    - return()

Loan and Reservation System

8. Reservation

- Tracks member reservation details.
- Status values come from ReservationStatus enumeration (pending, completed, cancelled).
- Includes methods to cancel reservation and notify availability.

9. BookLoan

- Represents a loaned book instance.
- Stores loan date, due date, and return date.
- Includes method to calculate fines.

10. Fine

- Tracks overdue fines.
- Contains fine amount, paid status, and a method payFine().

Enumerations

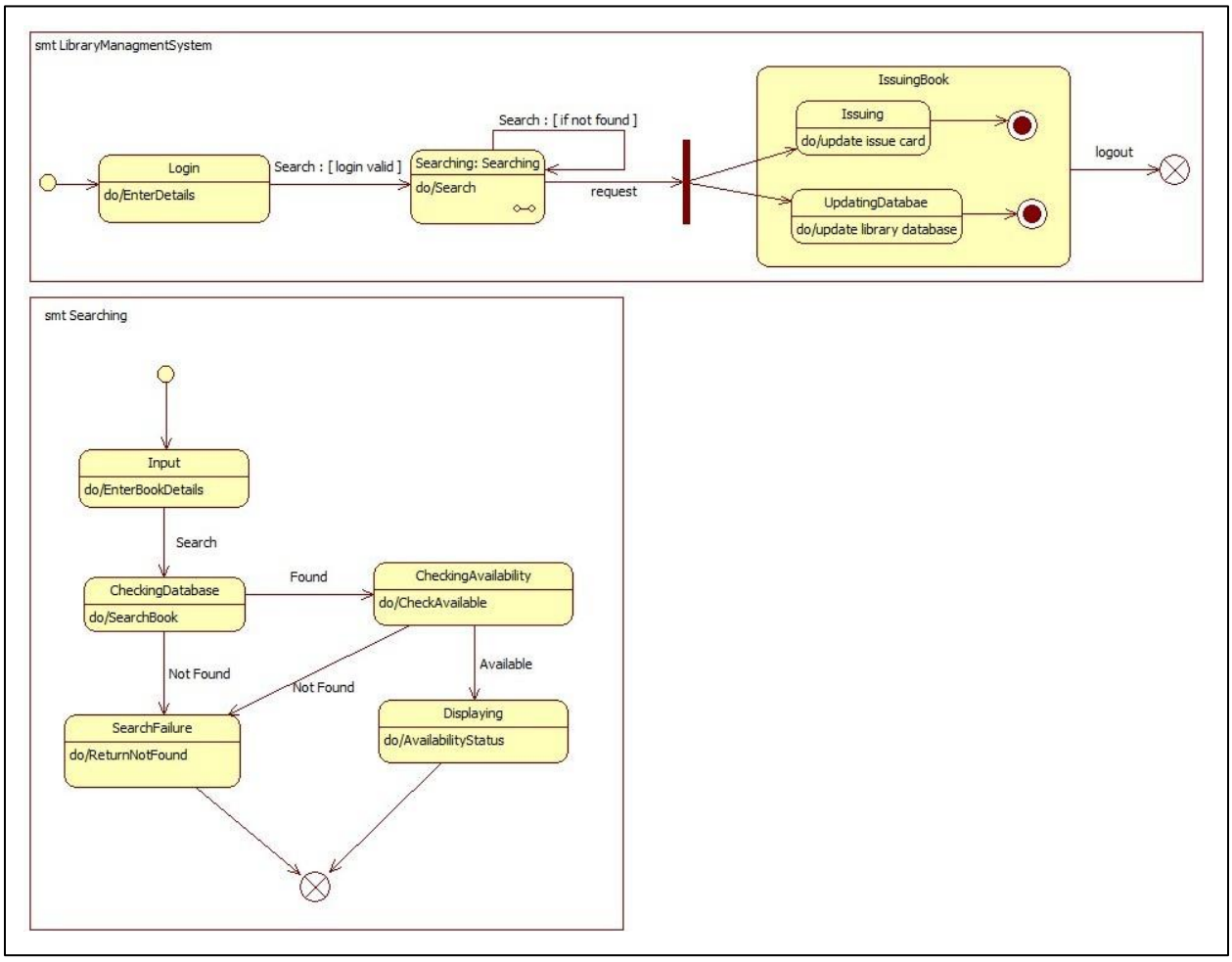
Two enumerations define book-related states:

- **BookStatus**
    - available
    - reserved
    - loaned
    - lost
    - damaged
- **ReservationStatus**
    - pending
    - completed
    - cancelled

Relationships Summary

- A **Library** has multiple **BookCopies** and Members.
- A **Member** can have multiple loans and reservations.
- A **Librarian** manages book copies and members.
- A **Book** can have multiple copies and multiple authors.
- A **BookCopy** may be associated with a BookLoan and Reservation.

State Diagram:



Overview

This state machine diagram models the internal behavior of a library management system during the process of book searching and issuing. It includes two main state regions:

- The overall system workflow (top section)
- The detailed searching process (lower section)

System-Level State Flow

1. **Login State**
    - The system begins in the login state.
    - The user enters credentials.
    - If the login is valid, the system transitions to the next state.
    - If the login fails, the user remains in the login state.

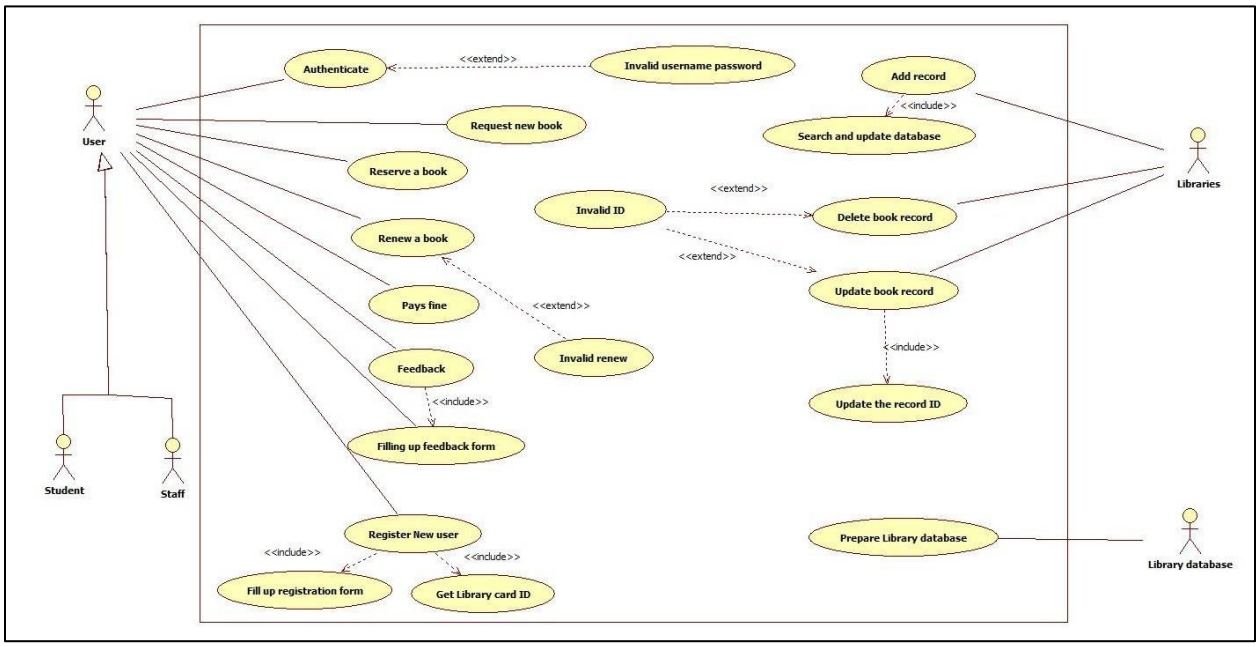
2. **Searching State**
    - Once logged in, the user enters a search request for a book.
    - The system remains in this state until a request is issued.
3. **Issuing Book Composite State**
    - After a book search request is successfully processed, the system transitions to this composite state.
    - It has two parallel sub-states:
        - **Issuing** (updates issue card or borrower record)
        - **Updating Database** (updates system records and book status)
    - Both tasks must finish before completing the issuing process.
4. **Logout**
    - The final state occurs once the issuing process is completed and the user logs out.

Detailed Searching Process

This section expands the internal behavior of the **Searching** state:

1. **Input**
    - The user enters book details (title, author, etc.).
2. **Checking Database**
    - The system queries the library database to find the book.
3. Decision: Book Found?
    - **If not found:**
        - The system enters the **Search Failure** state and returns a "not found" message.
    - **If found:**
        - The system proceeds to **Checking Availability**.
4. Decision: Available?
    - **If available:**
        - The system enters the **Displaying** state and shows book availability status.
    - **If unavailable:**
        - The flow ends without issuing.

Use-Case Diagram:



Overview

This use case diagram models the functionality of a library system and the interactions between three main actors:

- **User** (with two subtypes: Student and Staff)
- **Librarian**
- **Library Database**

It covers book-related actions (borrowing, reserving, renewing), account management, feedback, and record maintenance.

Actors and Their Roles

1. User (Student or Staff)

Users interact with the system to access library services. Their use cases include:

- Authenticate (login)
- Request a new book
- Reserve a book
- Renew a book
- Pay fines
- Submit feedback
- Register as a new member

Some use cases have support cases:

- If authentication fails → **Invalid username/password** (extends)
- If renewal fails → **Invalid renewal** (extends)

Registration involves two included use cases:

- Fill up registration form
- Get library card ID

Feedback submission includes:

- Fill up feedback form

2. Librarian

The librarian manages library data and book inventory. Their use cases include:

- Search and update database
- Add record
- Delete book record
- Update book record

Additional relationships:

- Updating a record includes updating record ID.
- Searching and updating database includes adding a record.

Invalid ID during update or delete extends from the transaction.

3. Library Database

Acts as the supporting system storing and responding to librarian operations.
It participates in:

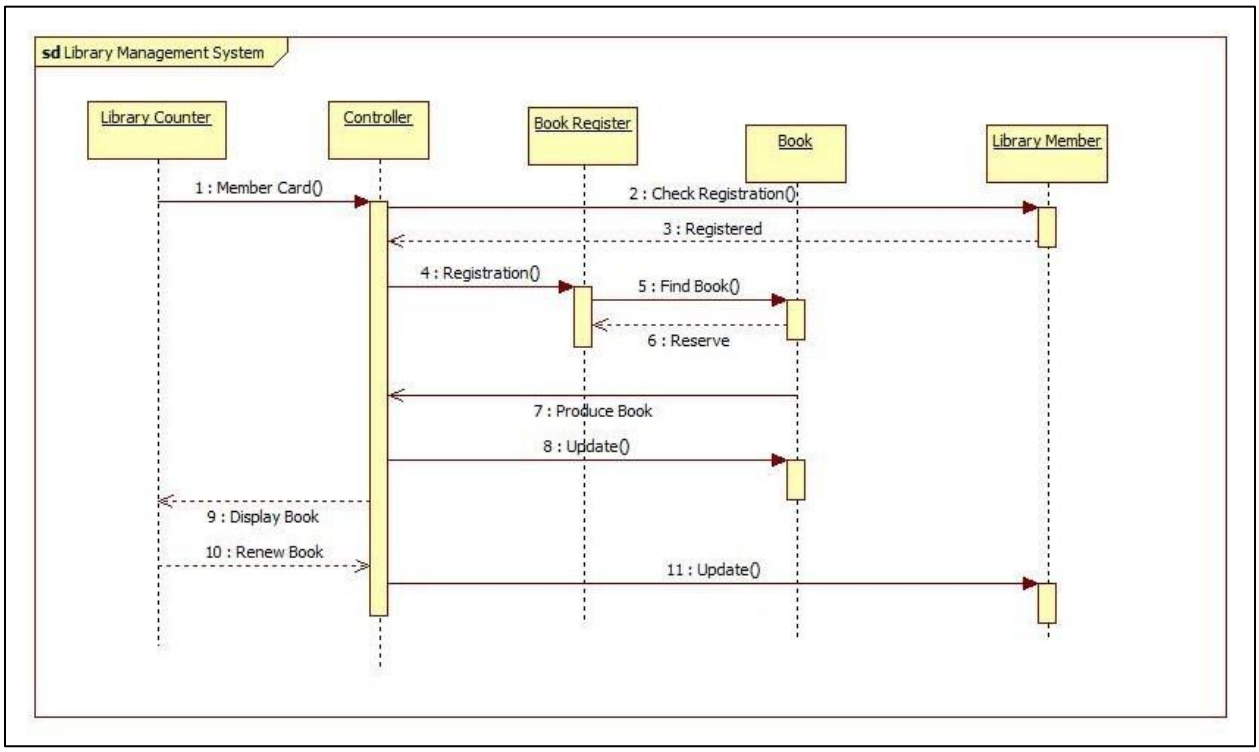
- Prepare library database
- Update record ID
- Manage stored data

Use Case Relationships

- **Include (mandatory steps)** is used when one use case depends on another:
    - Register → Fill registration form, Get Library ID
    - Search database → Add record
    - Update record → Update record ID

- **Extend (optional or conditional steps)** is used for error handling or alternate flows:
    - Authenticate → Invalid username/password
    - Renew book → Invalid renew
    - Search/update database → Invalid ID

Sequence Diagram:



Overview

This sequence diagram illustrates the interaction between different components of the library system when a library member attempts to borrow or renew a book. The entities involved are:

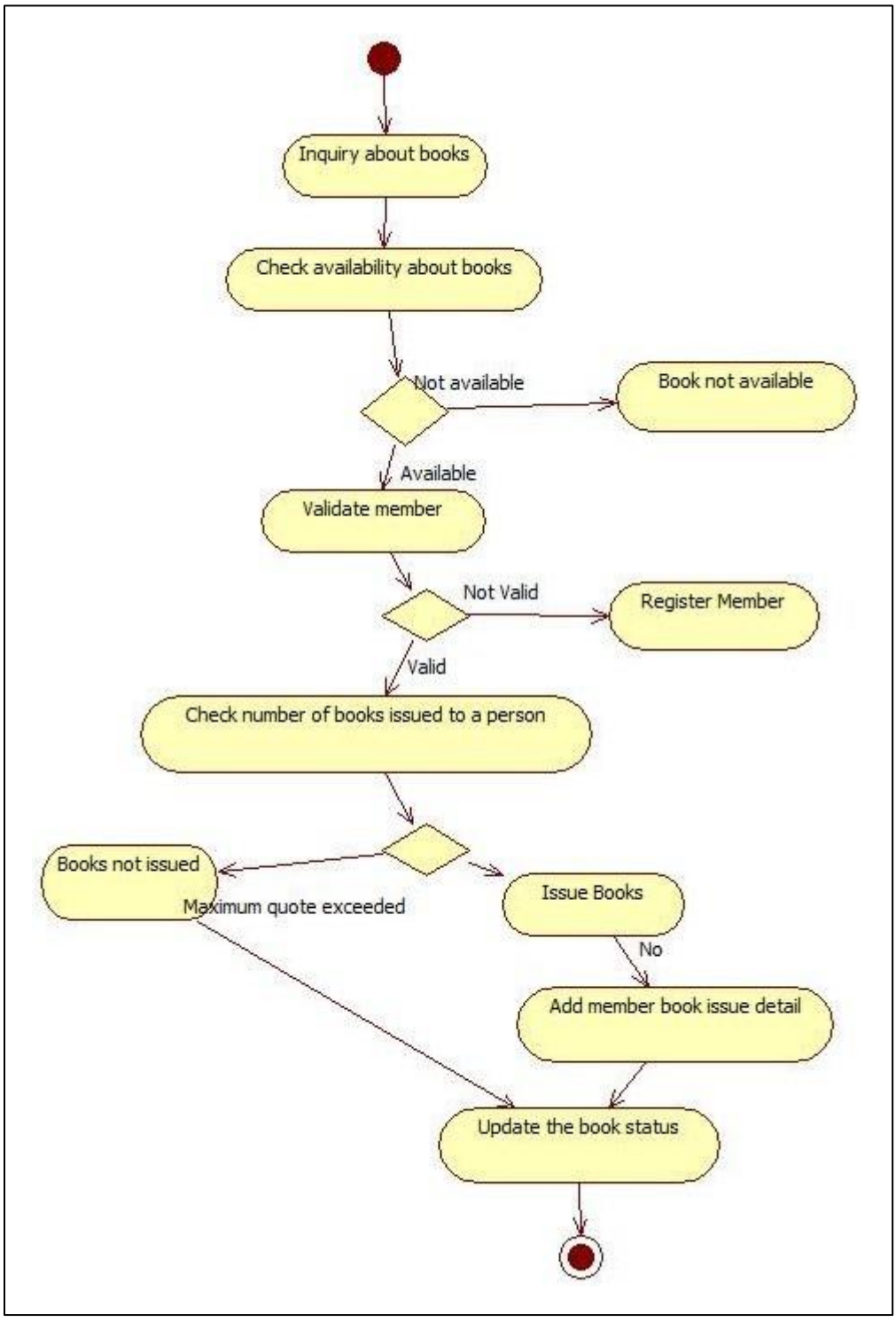
- Library Counter
- Controller
- Book Register
- Book
- Library Member

The diagram shows the flow of communication required to verify membership, locate the book, reserve it, and update system records.

Step-by-Step Flow

1. **Membership Verification**
    - The **Library Counter** begins the process by sending a *Member Card()* request to the **Controller**.
    - The controller forwards this request to the **Book Register** with *Check Registration()*.
    - The Book Register responds with *Registered* confirmation.
    - The system informs the **Library Member** that their registration is valid.
2. **Book Search and Reservation**
    - The Controller sends *Registration()* confirmation back to the Library Counter.
    - The Library Counter sends a *Find Book()* request to the **Book** object.
    - Once the book is located, the **Book** responds back.
    - The system then sends a *Reserve* request to the **Book** to temporarily allocate the book to the member.
3. **Book Issue Process**
    - After reservation, the **Book** sends *Produce Book()* back to the Controller.
    - The system then performs a database update with *Update()*.
4. **Book Display and Renewal**
    - The Counter displays the issued book to the member (*Display Book()*).
    - If needed, the member sends a *Renew Book()* request.
    - The system updates renewal details and sends another *Update()* to maintain accurate records.

Activity Diagram:



Overview

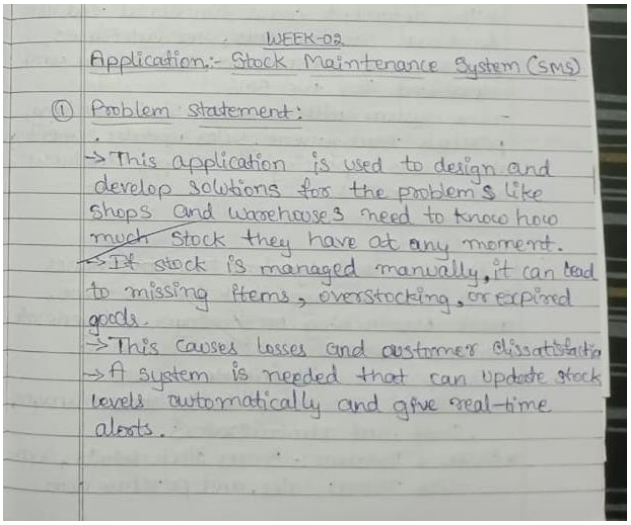
This activity diagram represents the workflow followed when a user attempts to borrow a book from the library. It includes steps such as checking availability, validating membership, verifying borrowing limits, and updating records. The diagram also includes alternate paths for cases where the process cannot continue.

Step-by-Step Process

1. **Inquiry About Books**
    - The process begins when a user makes an inquiry about a book.
2. **Check Availability**
    - The system checks whether the requested book is available.
    - Decision:
        - If the book is **not available**, the process ends with **Book not available**.
        - If the book is **available**, the process continues.
3. **Validate Member**
    - The system checks if the user is a registered library member.
    - Decision:
        - If the member status is **not valid**, the system directs the user to **Register Member** and stops the issuing process.
        - If valid, the process continues.
4. **Check Borrowing Limit**
    - The system checks how many books the member already has issued.
    - Decision:
        - If the borrowing limit is exceeded, the process stops with **Books not issued**.
        - If the member is within the allowed limit, the system proceeds.
5. **Issue Book**
    - The system approves the issuing of the book.
6. **Record Borrowing Details**
    - The system adds the issued book details to the member's records.
7. **Update Book Status**
    - The book status in the system is updated to reflect that it is loaned out.
8. **End**
    - The process successfully concludes.

4. Stock Maintenance System

Problem Statement & SRS-Software Requirements Specification:

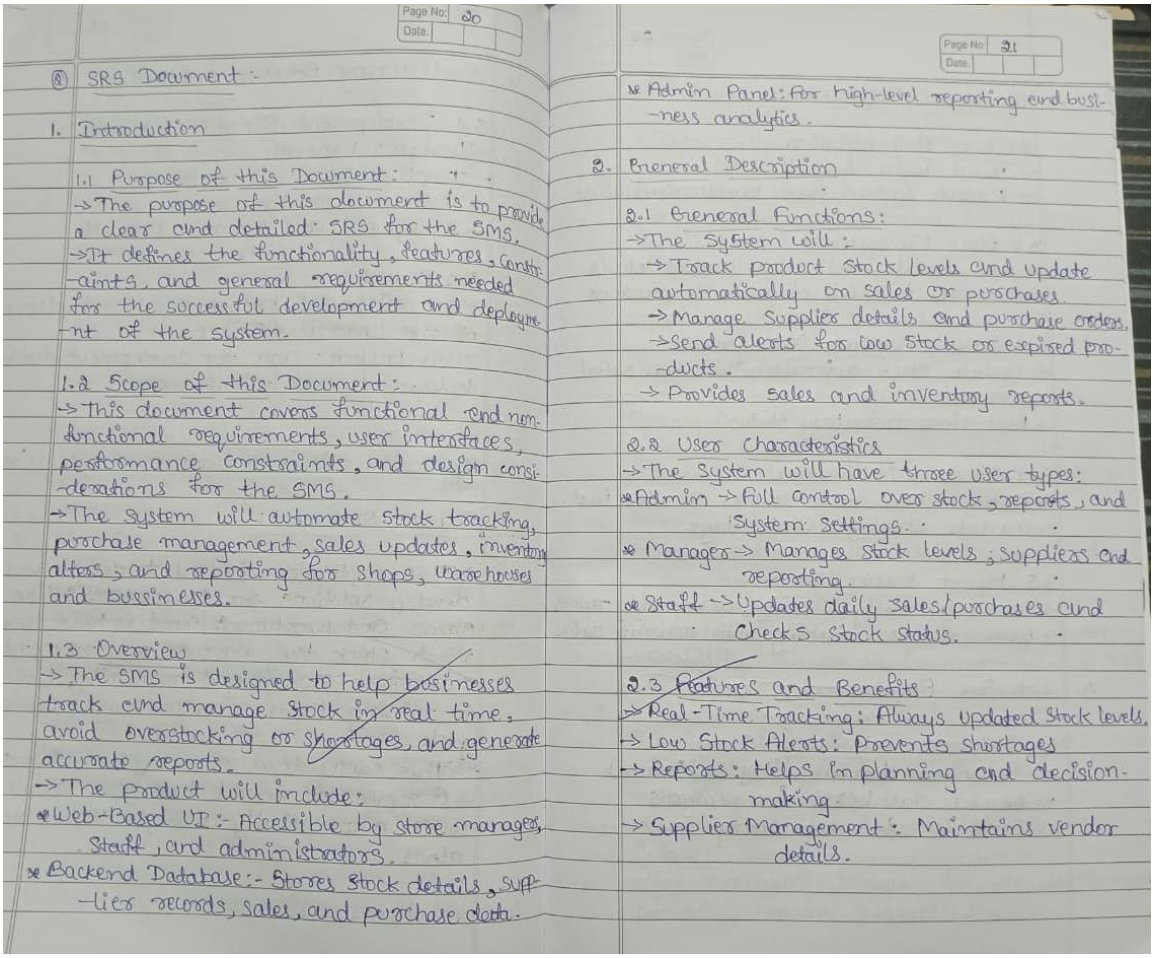


WEEK-02

Application:- Stock Maintenance System (SMS)

① Problem Statement:

->This application is used to design and develop solutions for the problems like shops and warehouses need to know how much stock they have at any moment.
->If stock is managed manually, it can lead to missing items, overstocking, or expired goods.
->This causes losses and customer dissatisfaction
->A system is needed that can update stock levels automatically and give real-time alerts.



Page No: 20
Date:

② SRS Document:

1. Introduction

1.1 Purpose of this Document:
->The purpose of this document is to provide a clear and detailed SRS for the SMS.
->It defines the functionality, features, constraints, and general requirements needed for the successful development and deployment of the system.

1.2 Scope of this Document:
->This document covers functional and non-functional requirements, user interfaces, performance constraints, and design considerations for the SMS.
->The system will automate stock tracking, purchase management, sales updates, inventory alters, and reporting for shops, warehouses and bussinesses.

1.3 Overview
->The SMS is designed to help businesses track and manage stock in real time, avoid overstocking or shortages, and generate accurate reports.
->The product will include:
* Web-Based UI:- Accessible by store managers, staff, and administrators.
* Backend Database:- Stores stock details, suppliers records, sales, and purchase data.

Page No: 21
Date:

* Admin Panel: For high-level reporting and business analytics.

2. General Description

2.1 General Functions:
->The System will:
->Track product stock levels and update automatically on sales or purchases.
->Manage Supplier details and purchase orders.
->Send alerts for low stock or expired products.
->Provides sales and inventory reports.

2.2 User Characteristics
->The System will have three user types:
* Admin -> Full control over stock, reports, and System settings.
* Manager -> Manages stock levels, suppliers and reporting.
* Staff -> Updates daily sales/purchases and Checks stock status.

2.3 Features and Benefits
->Real-Time Tracking: Always updated stock levels.
->Low Stock Alerts: Prevents shortages
->Reports: Helps in planning and decision-making.
->Suppliers Management: Maintains vendor details.

3. Functional Requirements

3.1 User Authentication
-> Secure login for all users.
-> Protects system from unauthorized access.

3.2 Stock Management
-> Add, update, and delete product details.
-> Accurate tracking of available products.

3.3 Sales & Purchase Management
-> Update stock automatically when sales/purchases occur.
-> Real-time inventory status.

3.4 Alerts & Notification:
-> Notify when stock reaches reorder level.
-> Prevents shortages and delays.

3.5 Report Generation:
-> Generate daily, weekly, or monthly reports.
-> Better decision-making with accurate data.

4. Interface Requirements:

4.1 Software Interfaces
-> Relational databases (MySQL/Oracle)
-> Admin dashboard for analytics
-> Staff UI for data entry.

4.2 Communication Interfaces
-> Web interface over HTTPS
-> Data exchange through REST APIs

5. Performance Requirements
-> Stock updates must reflect in real time. Should be < 2 sec.
-> System must handle 2000 concurrent transactions.
->Database queries must respond within 3sec.

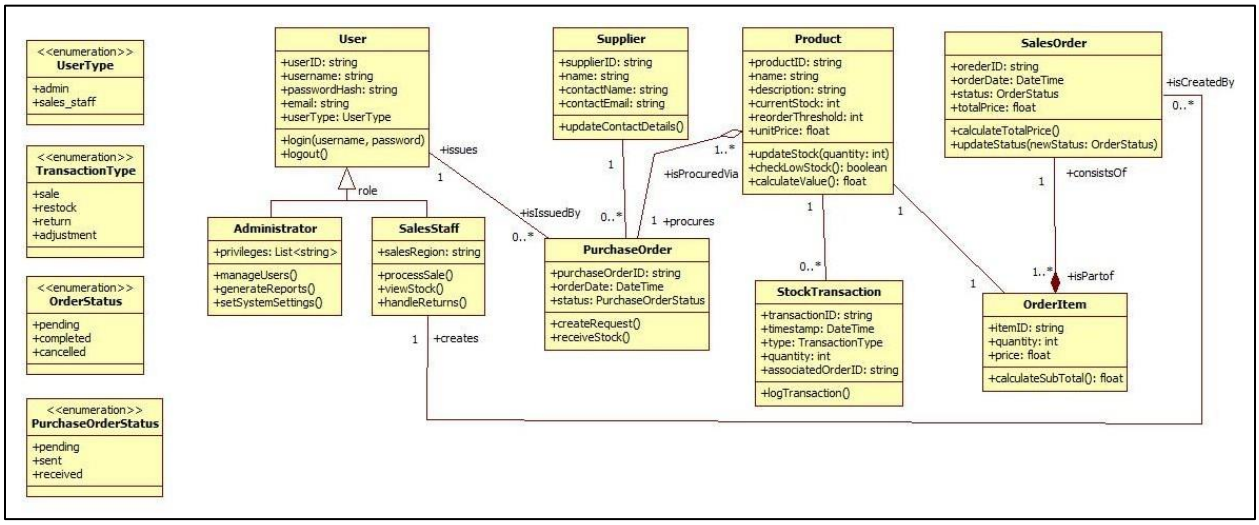
6. Design Constraints
-> Must be used modern frameworks (Java Spring Boot / Python Django)
->Must support web and mobile devices.
-> Standard hardware : 8GB RAM, 1TB storage.

7. Non-Functional Attributes:
-> Security: Role-based access, encryption of bussiness data.
-> Reliability: 99% Uptime.
-> Scalability: Expandable for large warehouses.
->Usability: Simple UI for non-technical staff.

8. Preliminary Schedule and Budget: (6 months)
-> Design : 1 month
->Implementation : 2 months
-> Testing : 1 month
->Deployment and support : 1 month

Budget:
->The estimated budget for the development of SMS can be $120,000.

Class Diagram:



Overview

This class diagram models the structure and workflow of a **Stock Maintenance System**, where inventory levels, product records, suppliers, and sales activities are tracked and managed digitally. The system also includes user access roles and transaction tracking.

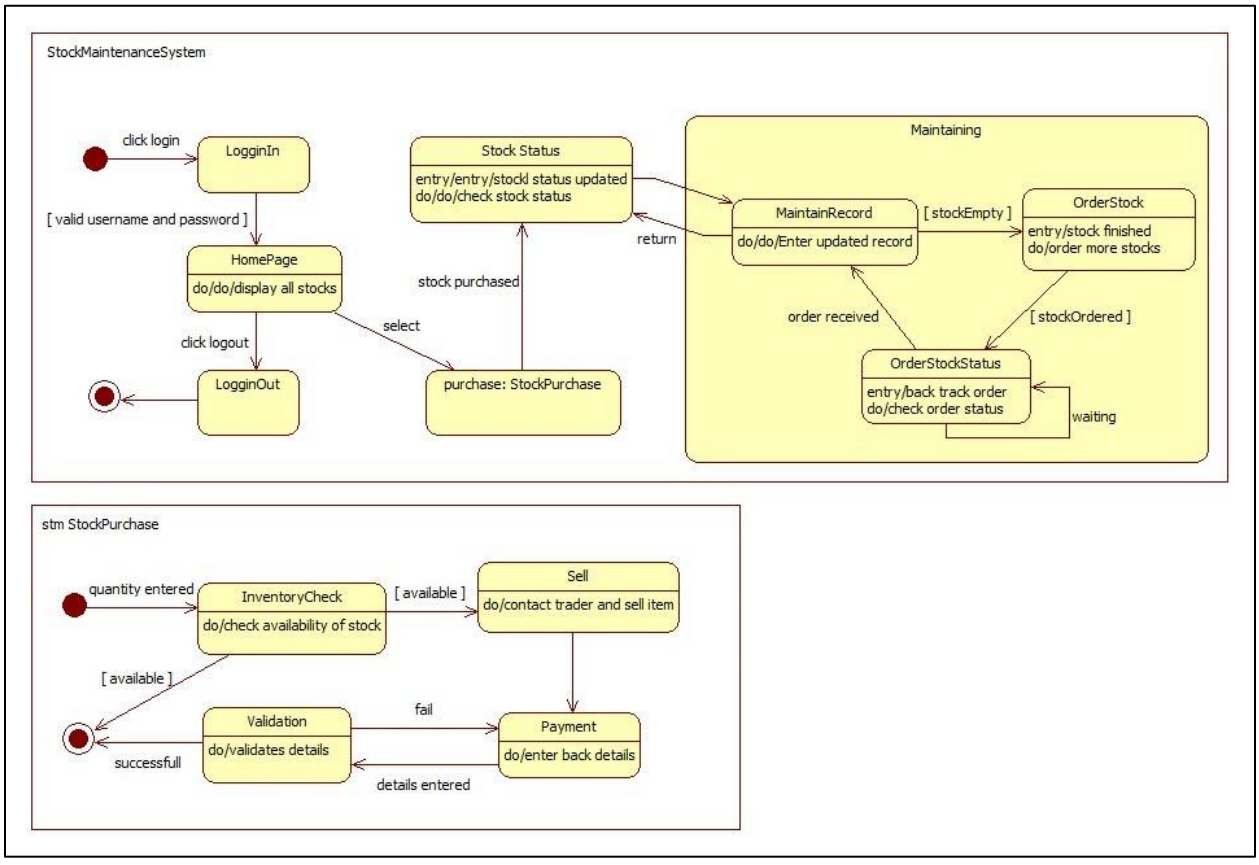
Key Components and Roles

1. **User & Roles**
    - The system has a main User class with attributes like username, passwordHash, email, and userType.
    - Two specialized user roles extend it:
        - **Administrator**
            - Manages system settings, user accounts, and reports.
        - **SalesStaff**
            - Handles day-to-day stock operations such as selling items, returns, and viewing stock.
2. **Product Management**
    - Product represents each stock item and stores details like product name, current stock, reorder threshold, and unit price.
    - Methods like updateStock(), checkLowStock(), and calculateValue() help monitor and manage product inventory levels.
3. **Order Handling**
    - SalesOrder represents customer purchases and includes IDs, dates, and order status.

- Each order contains one or more OrderItem entries, which define product, quantity, and price.

4. **Supplier and Purchase Orders**
    - Supplier stores external vendor details.
    - PurchaseOrder is used to procure stock from suppliers. It includes order status tracking (pending, sent, received).
5. **Stock Transactions**
    - All stock adjustments (sales, returns, restocking, adjustments) are logged in StockTransaction.
    - This keeps a detailed audit history and link to related documents.

State Diagram:



Overview

This state machine diagram represents how the Stock Maintenance System behaves during login, stock checks, purchasing, and maintaining stock records. It shows major system states, transitions, decisions, and embedded behavior for processing stock purchases.

The diagram is divided into two parts:

1. **Main System State Flow**
2. **Detailed Stock Purchase State Machine**

1. Main System State Flow

Login Phase

- The system starts when the user clicks login.
- The system enters the **LoggingIn** state.
- The system checks the entered username and password.
    - If credentials are valid → transition to **HomePage**
    - If not valid → remain in LoggingIn until corrected.

Homepage

- Once logged in, the HomePage displays available stock and actions.
- The user may:
    - Select a purchase request → moves to **StockPurchase**
    - View or update records
    - Logout → enters **LoggingOut** (final state)

Stock Status and Maintenance Phase

Once interacting with stock data, the system enters the **Stock Status** state where it:

- Updates stock levels
- Checks status (low stock, active stock, sold stock, etc.)

From here, the system may transition into a composite state called **Maintaining**, which contains three internal states:

a) MaintainRecord

- Used when inventory needs to be updated manually.
- Happens when stock is modified.

Decision:

- If stock is **empty**, transition to **OrderStock**.

b) OrderStock

- The system orders new stock.
- After ordering, it moves to **OrderStockStatus**.

c) OrderStockStatus

- Tracks supplier order status.
- Once stock arrives, the system returns to the **Stock Status** state.

2. Sub State Machine: Stock Purchase

This section details what happens when a user attempts to purchase stock.

Inventory Check

- System verifies if requested quantity is available.
- Decision:
    - If not available → flow stops.
    - If available → continue.

Sell

- The system contacts the trader or seller and proceeds with transaction.

Validation

- System validates details such as:
    - Quantity
    - Price
    - Purchase permission
- If validation fails → return to Payment for correction.

Payment

- User enters necessary payment information.
- Upon success → purchase is completed.

Final Steps

After the purchase is completed:

- Stock levels are updated.
- User is returned to **Stock Status** screen.
- The process ends with updated records.

Use-Case Diagram:

Overview

This use case diagram represents the functional requirements of a Stock Management System. The diagram identifies multiple actors and their interactions with the system, showing how stock is managed, purchased, tracked, updated, and validated. It also includes optional and reusable behaviors via *include* and *extend* relationships.

Actors and Their Roles

1. User

A general user interacts with the system to:

- Login / Logout
- View stock status
- Check stock availability
- Filter stock by category

- Purchase stock
- Make payments
- Request refunds
- Cancel purchase

The user initiates the main stock-related workflow.

2. Admin (inherits from User)

The Admin has additional privileges beyond a regular user, including:

- Validate transactions
- Update database
- Maintain stock records
- Generate reports
- Order new stock
- Check order status

Administrators help manage backend processes and stock control.

3. Trader

This actor interacts when stock needs to be sourced externally:

- Sell stock
- Provide replacement items
- Receive order details
- Confirm deliveries

The system interacts with the trader during replenishment or stock negotiation.

4. Stock Tracker System

An external system that assists with:

- Inventory checking
- Stock-level validation
- Tracking ordered stock
- Sending automated updates

Main Use Cases and Their Relationships

Login Process

- Includes: Validate Credentials
- Extends: Forgot Password

Purchasing Stock

- Includes:
    - Validation
    - Check Stock Status
    - Inventory Check
    - Payment
- Extends:
    - Apply Discount
    - Cancel Purchase

Payment

- Includes:
    - Refund Process (only if needed)
    - Validate Transactions

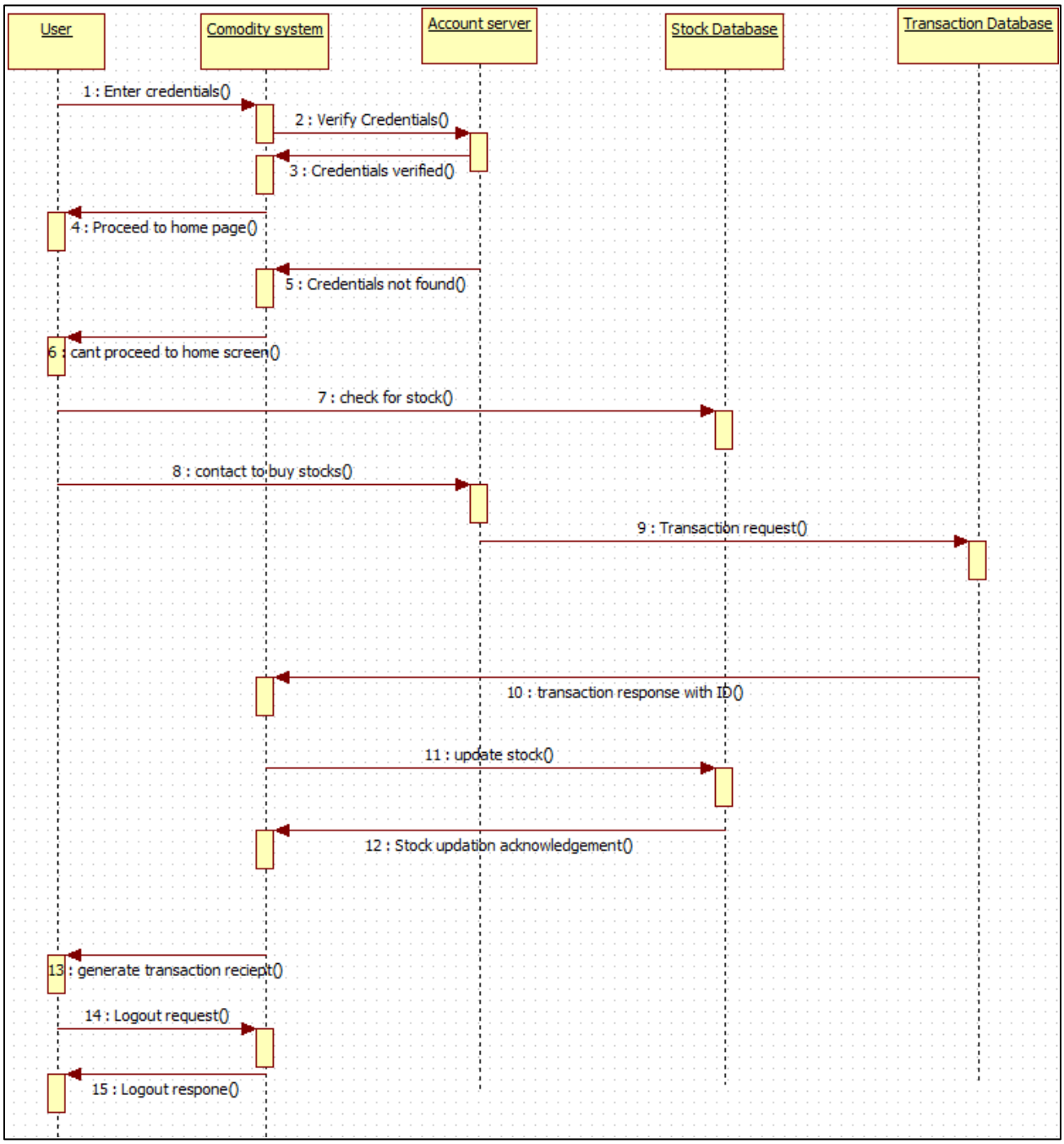
Administrative Use Cases

- Order New Stock (*includes:* Check Order Status)
- Maintain Record (*includes:* Update Database)
- Generate Report
- Terminate Session

Trader Interaction

- Includes:
    - Receive Order Details
    - Provide Replacement
    - Contact Trader

Sequence Diagram:



Overview

This sequence diagram describes the process of a user buying stock through a Stock Management System. It shows how the User, Commodity System, Account Server, Stock Database, and Transaction Database communicate during login, stock purchase, and logout.

Step-by-Step Interaction

1. **User Login**
    - The user enters login credentials.
    - The Commodity System sends a *Verify Credentials()* request to the **Account Server**.
    - The Account Server responds:
        - If valid → *Credentials verified()* is returned.
        - If invalid → *Credentials not found()* is sent and user cannot proceed to the home screen.
2. **Homepage Access**
    - If credentials are correct, the system sends a *Proceed to home page()* message to the user.

Stock Checking and Purchase Process

3. **Stock Availability Check**
    - The Commodity System sends a *Check for stock()* request to the **Stock Database** to verify item availability.
4. **Stock Purchase Request**
    - Once stocks are confirmed available, the user initiates a purchase request through:
        - *Contact to buy stocks()* sent from User to the Commodity System.
5. **Transaction Processing**
    - The Commodity System sends *Transaction request()* to the **Transaction Database**.
    - The Transaction Database processes the request and returns:
        - *Transaction response with ID()* → uniquely identifying successful transaction.

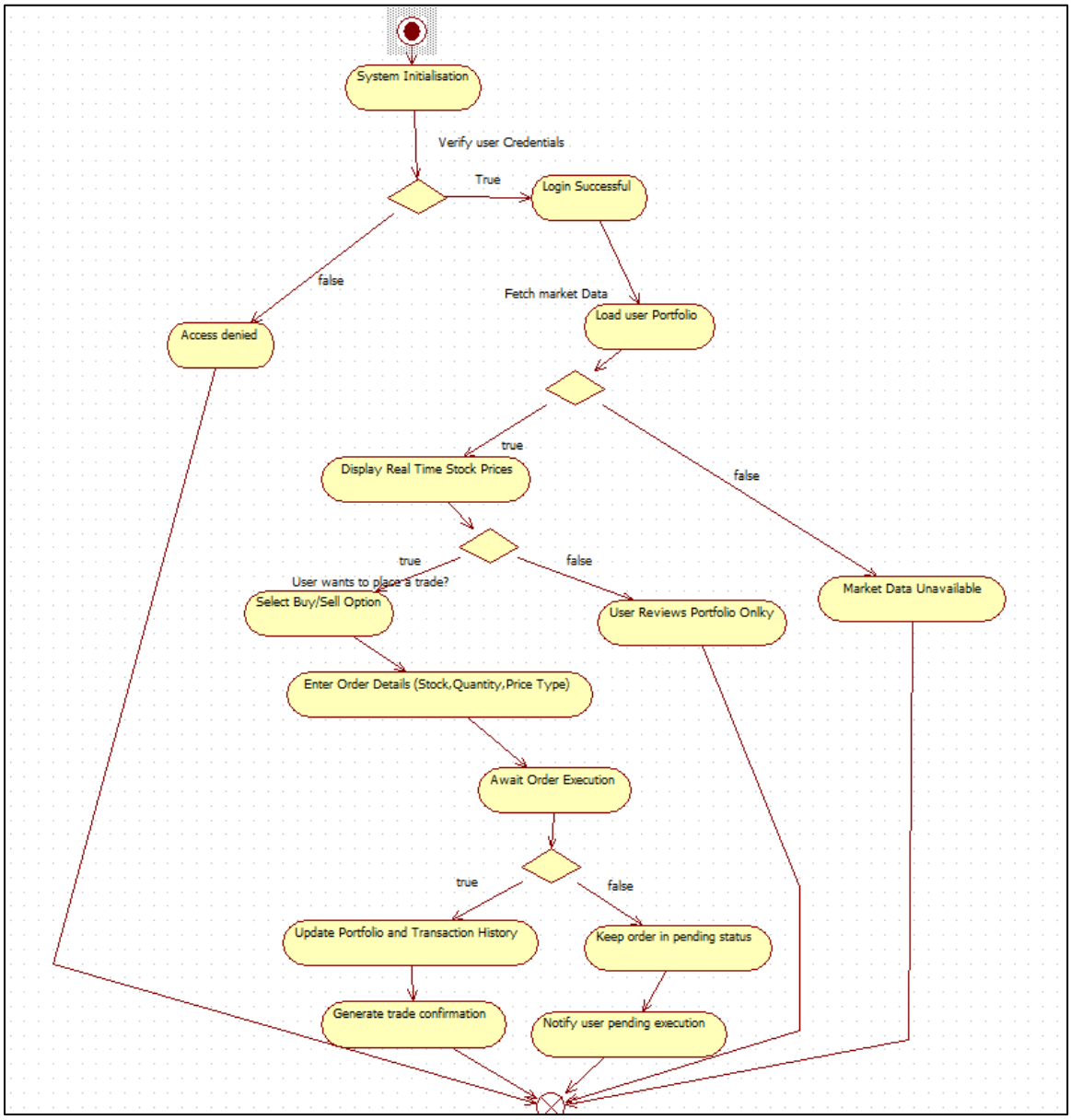
Stock Update

6. **Stock Deduction**
    - After a successful transaction, the Commodity System sends *Update stock()* to the Stock Database to reduce inventory quantity.
    - The Stock Database responds with:
        - *Stock update acknowledgement()* confirming completion.

Completion and Logout

7. **Receipt Generation**
    - The Commodity System sends *Generate transaction receipt()* to the User.
8. **Logout**
    - The user sends a *Logout request()*.
    - Commodity System confirms logout with a *Logout response()*.

Activity Diagram:



Overview

This activity diagram represents the workflow of an online stock management system from login to executing and managing a trade. It includes validation steps, decision points, and alternate flows depending on system response and user intent.

Step-by-Step Process Description

1. **System Initialization**
    - The process begins when the system starts.
2. **Credential Verification**
    - The system verifies the user credentials.
    - Decision:
        - If credentials are **invalid**, access is denied and the process ends.
        - If **valid**, the user logs in successfully.
3. **Load Market and Portfolio Data**
    - The system fetches live market data and loads the user's portfolio.
    - Decision:
        - If market data is unavailable, the system displays a message and ends or waits.
        - If available, the system continues.
4. **Display Real-Time Stock Prices**
    - The system shows live stock price updates.
5. **User Action Decision: Trade or View Only**
    - The user decides whether they want to perform a trade.
    - Decision:
        - If **no**, the user only reviews their portfolio and activity may end.
        - If **yes**, the system proceeds with the order process.
6. **Trade Order Entry**
    - The user selects Buy/Sell and enters order details including:
        - Stock name
        - Quantity
        - Price type (market/limit, etc.)
7. **Await Execution**

    - The order is submitted and waits for market execution.

8. **Execution Result Decision**
    - Decision:
        - If the trade is executed successfully:
            - The system updates the portfolio and transaction history.
            - A trade confirmation is generated.
        - If the trade is not executed:
            - The system keeps the order in pending state and notifies the user.

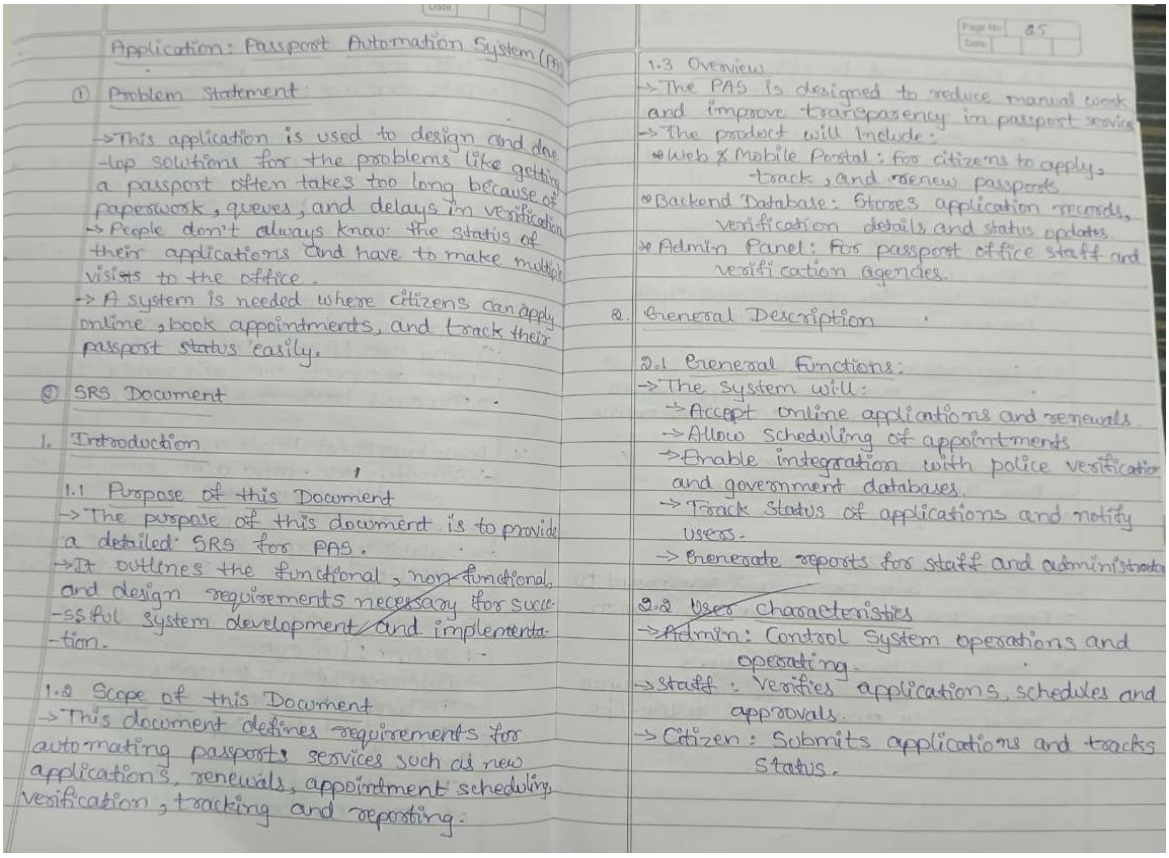
End State

The process ends after either:

- Successful trade confirmation, or
- Notification of pending execution, or
- User choosing only to review portfolio, or
- System error (like unavailable market data).

5. Passport Automation System

Problem Statement & SRS-Software Requirements Specification:



Application: Passport Automation System (PAS)

① Problem Statement

->This application is used to design and develop solutions for the problems like getting a passport often takes too long because of paperwork, queues, and delays in verification.
-> People don't always know the status of their applications and have to make multiple visits to the office.
-> A system is needed where citizens can apply online, book appointments, and track their passport status easily.

② SRS Document

1. Introduction

1.1 Purpose of this Document
->The purpose of this document is to provide a detailed SRS for PAS.
->It outlines the functional, non-functional, and design requirements necessary for successful system development and implementation.

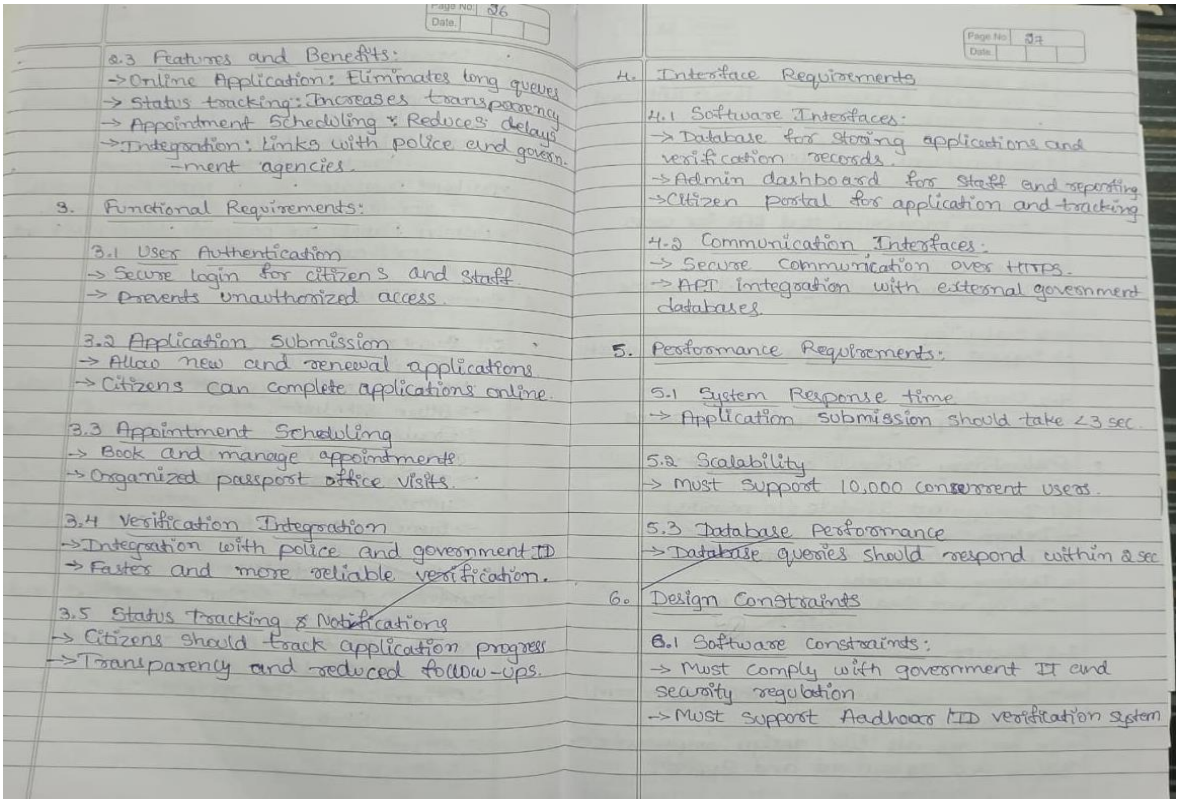
1.2 Scope of this Document
->This document defines requirements for automating passport services such as new applications, renewals, appointment scheduling, verification, tracking and reporting.

1.3 Overview
->The PAS is designed to reduce manual work and improve transparency in passport services.
->The product will include:
* Web & Mobile Portal: For citizens to apply, track, and renew passports.
* Backend Database: Stores application records, verification details and status updates.
* Admin Panel: For passport office staff and verification agencies.

2. General Description

2.1 General Functions:
->The system will:
->Accept online applications and renewals.
->Allow scheduling of appointments.
->Enable integration with police verification and government databases.
->Track status of applications and notify users.
->Generate reports for staff and administrators.

2.2 User Characteristics
->Admin: Control System operations and operating.
->Staff: Verifies applications, schedules and approvals.
->Citizen: Submits applications and tracks status.



2.3 Features and Benefits:
->Online Application: Eliminates long queues.
-> Status tracking: Increases transparency.
-> Appointment Scheduling & Reduces delays.
->Integration: Links with police and government agencies.

3. Functional Requirements:

3.1 User Authentication
-> Secure login for citizens and staff.
-> Prevents unauthorized access.

3.2 Application Submission
-> Allow new and renewal applications.
-> Citizens can complete applications online.

3.3 Appointment Scheduling
-> Book and manage appointments.
->Organized passport office visits.

3.4 Verification Integration
->Integration with police and government ID.
-> Faster and more reliable verification.

3.5 Status Tracking & Notifications
-> Citizens should track application progress.
->Transparency and reduced follow-ups.

4. Interface Requirements

4.1 Software Interfaces:
-> Database for storing applications and verification records.
->Admin dashboard for staff and reporting.
->Citizen portal for application and tracking.

4.2 Communication Interfaces:
-> Secure communication over HTTPS.
-> API Integration with external government databases.

5. Performance Requirements:

5.1 System Response time
-> Application submission should take < 3 sec.

5.2 Scalability
-> Must support 10,000 concurrent users.

5.3 Database Performance
->Database queries should respond within 2 sec.

6. Design Constraints

6.1 Software constraints:
-> Must comply with government IT and security regulation.
->Must support Aadhaar ID verification system.

6.2 Hardware Constraint:
→ Standard hardware of 16GB RAM and 8TB storage can be used.

7. Non-Functional Attributes:

7.1 Security:
→ Strong encryption and 2FA for users

7.2 Reliability:
→ The system should give 99.9% uptime.

7.3 Scalability
→ Support multiple regional passports offices

7.4 Usability
→ Clear, step-by-step application process.

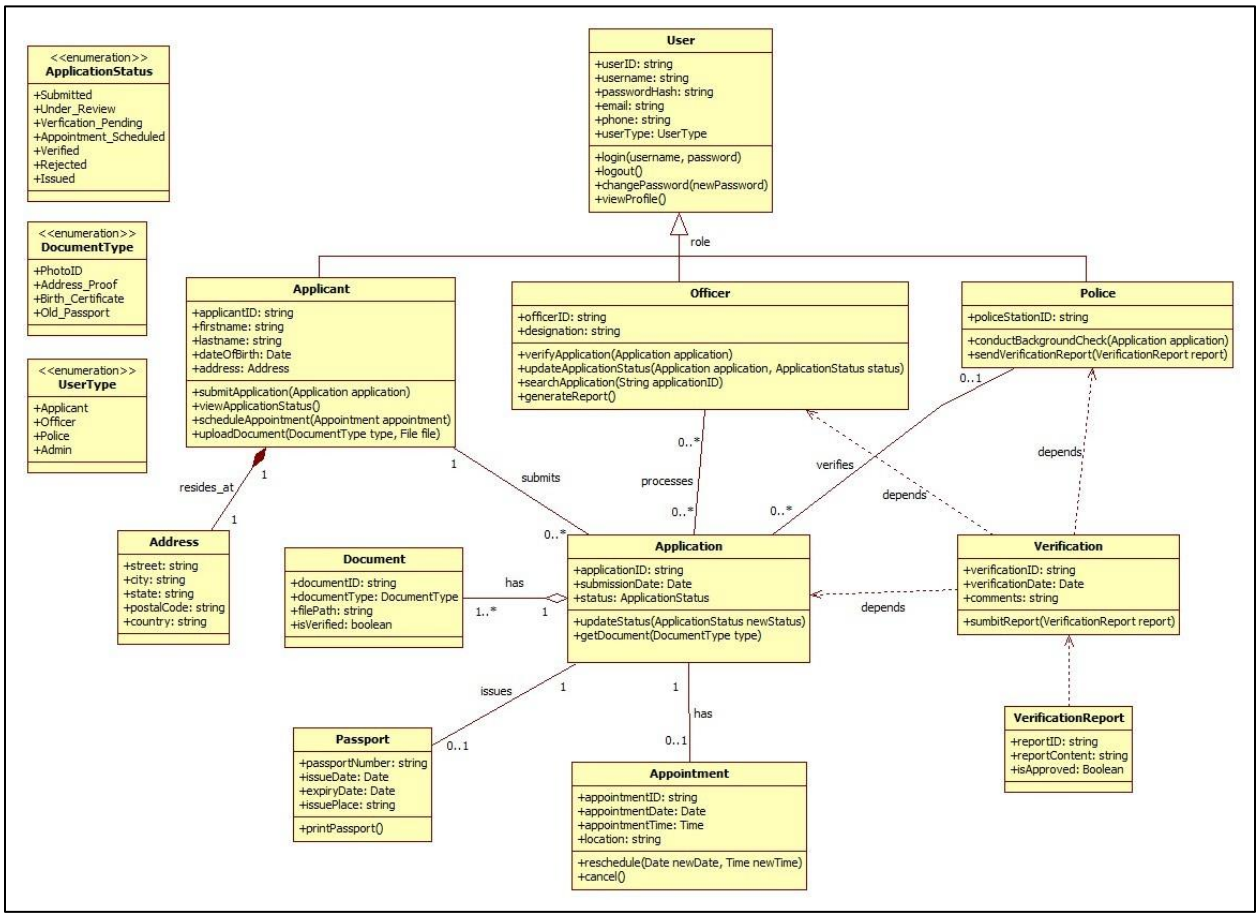
8. Preliminary Schedule and Budget:

8.1 Development Schedule: (10 months)
→ Design: 2 months
→ Implementation: 4 months
→ Testing: 2 months
→ Deployment and support: 2 months

8.2 Budget

→ The estimated budget of this PAS system is $700,000.
→ It includes all like design, implementation, testing and deployment and support.

Class Diagram:



Overview

This class diagram represents the structure and key components of a **Passport Automation System**. It defines important classes such as applicants, officers, police, documents, applications, verification steps, and appointments. The diagram also showcases relationships, system behaviors, and enumerations that define valid values.

Main Classes and Their Responsibilities

1. User

- The base class for all system users.
- Attributes store login and contact details.
- Methods allow authentication and profile management.
- **UserType** enumeration specifies roles:
    - Applicant
    - Officer
    - Police
    - Admin

All specialized users inherit from this class.

2. Applicant

- Represents the passport applicant.
- Contains personal details such as name, DOB, and address.
- Key functions:
    - Submit application
    - Upload required documents
    - Schedule appointment
    - View application status

3. Officer

- Reviews and processes applications.
- Performs approval, rejection, and forwards cases.
- Important methods include:
    - verifyApplication()
    - updateApplicationStatus()
    - searchApplication()
    - generateReport()

4. Police

- Performs background verification.
- Methods include:
    - conductBackgroundCheck()
    - sendVerificationReport()

5. Application

- Core entity representing the passport request.
- Tracks status using the **ApplicationStatus** enumeration:
    - Submitted
    - Under Review
    - Verification Pending
    - Appointment Scheduled
    - Verified
    - Rejected

- Issued

Each application may contain multiple documents and may issue a passport if approved.

6. Document

- Represents uploaded proof documents.
- Uses **DocumentType** enumeration such as:
  - Photo ID
  - Birth Certificate
  - Address Proof
  - Old Passport
    Each document has a verification status.

7. Verification and VerificationReport

- Verification contains police and officer assessment details.
- Report holds review comments and approval status.

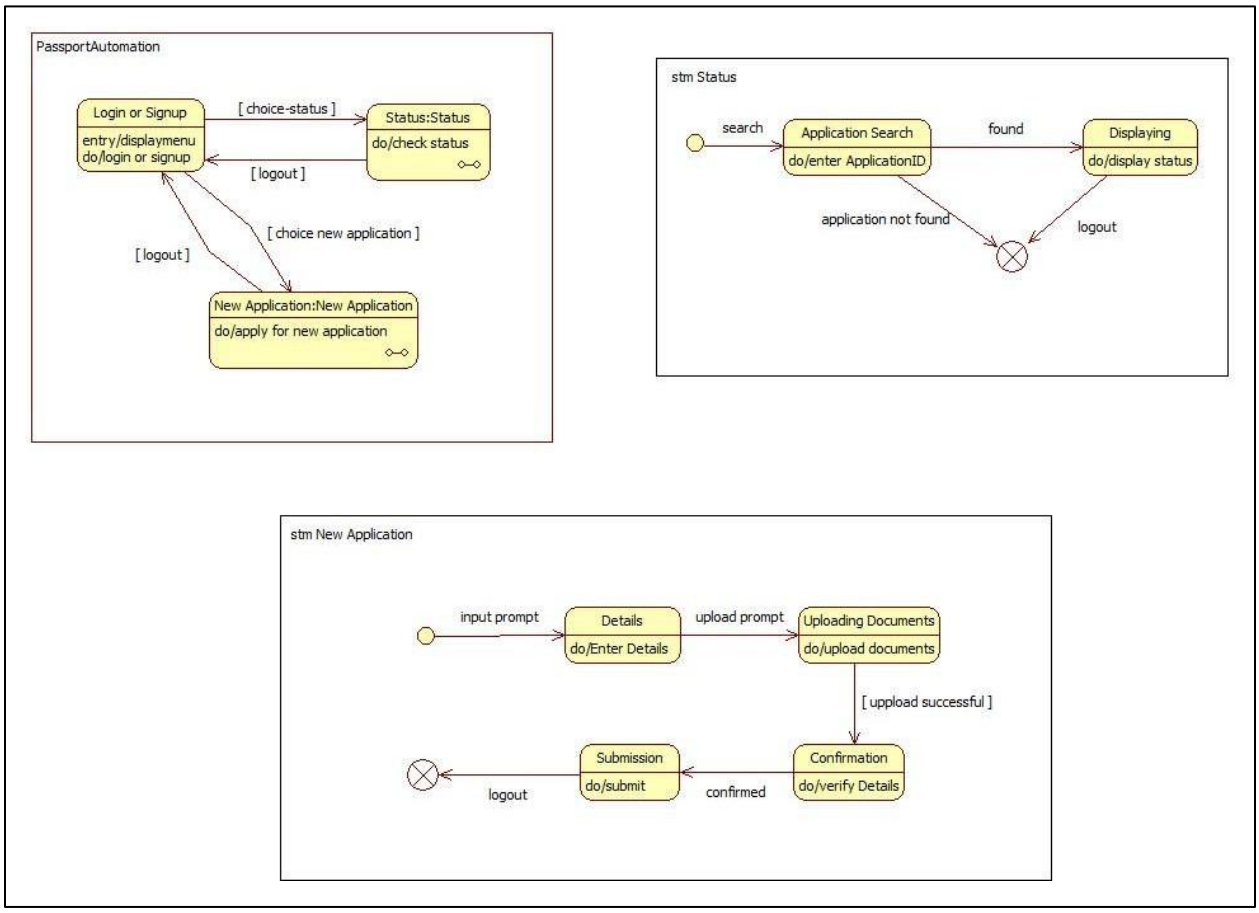
8. Appointment

- Allows scheduling or rescheduling of meetings for biometric check or verification.

9. Passport

- Issued only when the application reaches the **Issued** status.
- Contains passport number, issue date, and expiry date.

State Diagram:



Overview

This state machine diagram represents the behavior of a passport automation system. It describes how the system transitions between different states depending on user actions such as login, checking application status, or applying for a new passport.

The diagram is divided into three main parts:

1. **Main Passport Automation Flow**
2. **Status Check State Machine**
3. **New Passport Application State Machine**

1. Main System Flow

The system starts at the **Login or Signup** state where the user can either log in or create an account.

From this point, the user has two main choices:

- **Check existing application status**
- **Submit a new passport application**

Transitions:

Trigger	Next State
choice-status	Status state (check progress)
choice new application	New Application state
logout	Process terminates

2. Status Checking State Machine

This sub-state machine handles the process of checking the status of an already submitted passport application.

Steps:

1. User initiates search.
2. System goes to **Application Search** and prompts for an application ID.
3. Two possible outcomes:
    - If the ID exists → transition to **Displaying** state where the status is shown.
    - If not found → system ends.

This flow ensures tracking progress such as *submitted, under review, verified, or issued*.

3. New Passport Application State Machine

This sub-process handles submitting a new passport application.

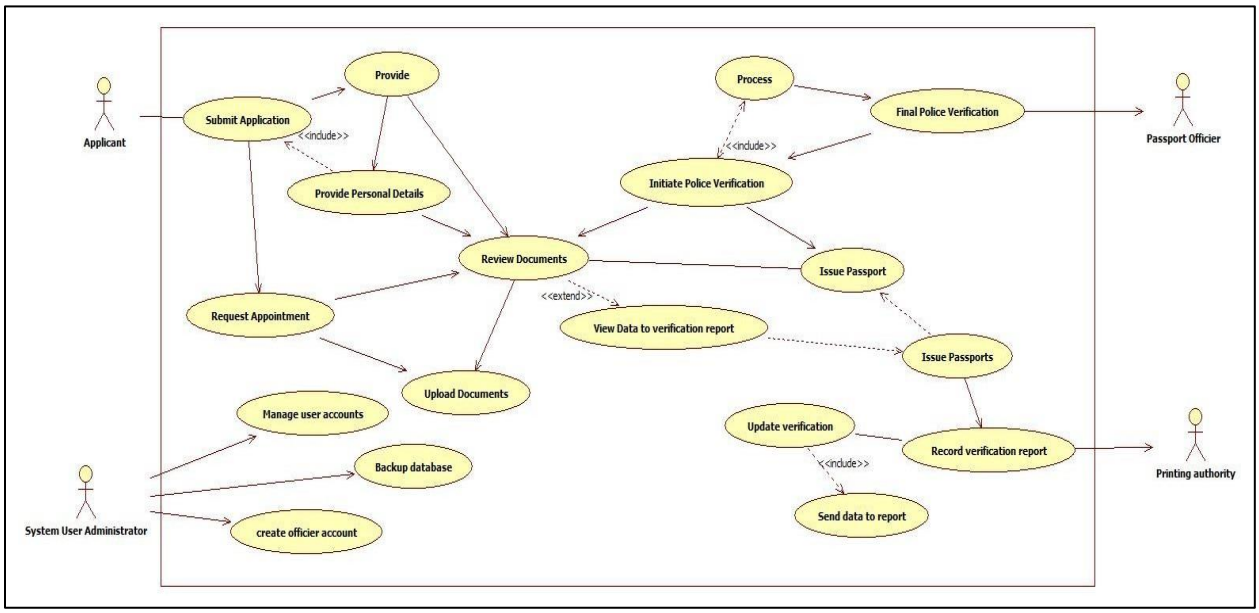
Flow of States:

State	Action
Details	User enters personal details
Uploading Documents	System prompts user to upload required documents
Confirmation	System verifies entered details and uploaded files
Submission	User submits the application

If details are verified successfully, the system submits the application; otherwise, the user may be prompted to retry.

A user may also **logout at any stage**, which terminates the process.

Use-Case Diagram:



Overview

This use case diagram represents the major functions of a **Passport Automation System** and shows how various actors interact with the system. The diagram includes four primary actors:

- **Applicant**
- **Passport Officer**
- **Printing Authority**
- **System User Administrator**

Each actor has specific tasks related to the passport application lifecycle, from submission to issuing the final passport.

Actors and Their Responsibilities

1. Applicant

The applicant interacts with the system to begin and follow through the passport application process. Their available use cases include:

- **Submit Application**
- **Provide Personal Details**
- **Upload Documents**
- **Request Appointment**

The use case **Provide Personal Details** is included within the *Submit Application* flow, meaning it is a mandatory step.

2. Passport Officer

The passport officer is responsible for reviewing and processing applications. Their role includes:

- **Review Documents**
- **Initiate Police Verification**
- **Final Police Verification**
- **Issue Passport**

Some actions such as **View Data to Verification Report** extend the *Review Documents* use case, meaning it occurs conditionally.

3. Printing Authority

This actor handles the physical passport creation after approval.

- **Record Verification Report**
- **Issue Passports**

The validation and report phases feed the printing authority process.

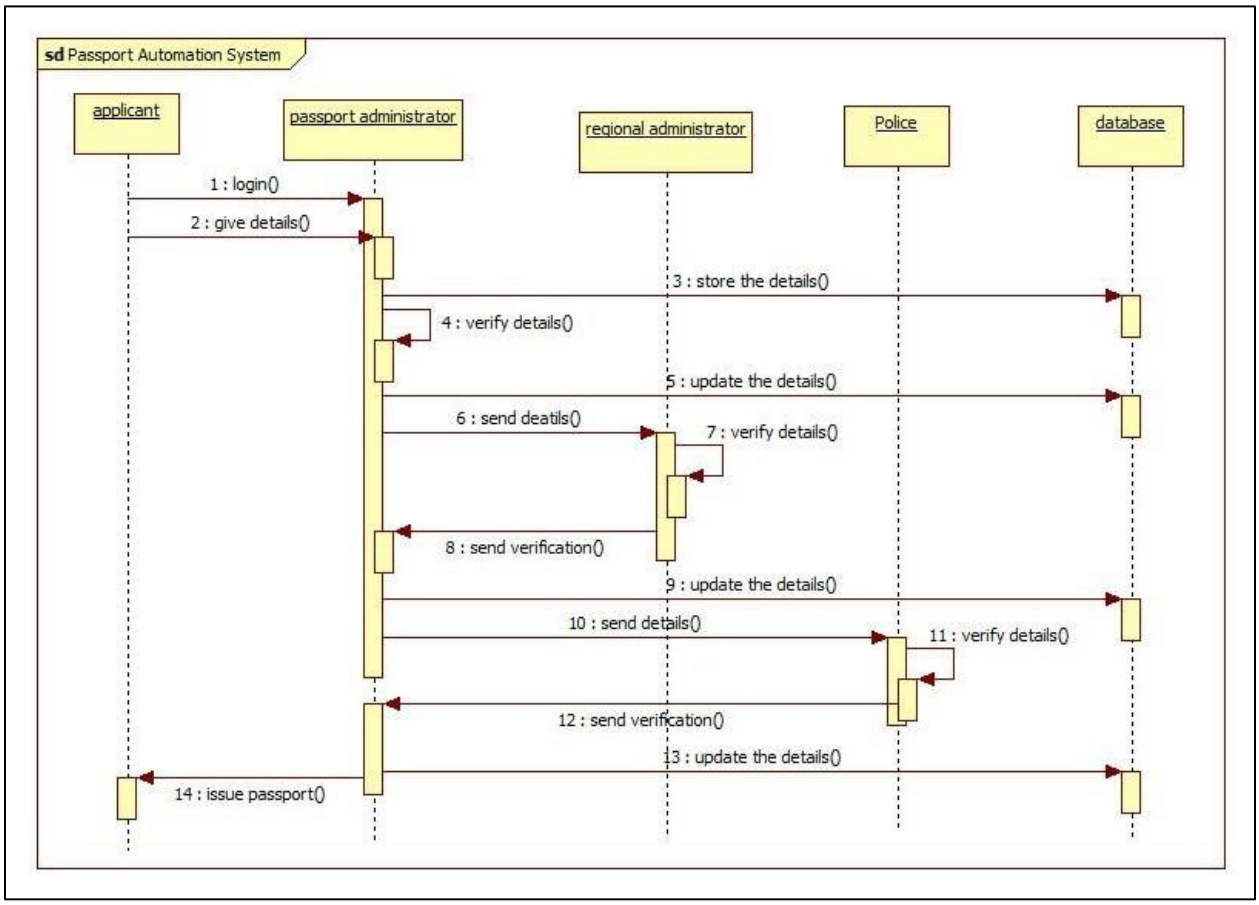
4. System User Administrator

This actor manages users and system maintenance. Their use cases include:

- **Manage User Accounts**
- **Create Officer Account**
- **Backup Database**

This ensures the system remains secure and operational.

Sequence Diagram:



Overview

This sequence diagram shows the workflow of applying for and approving a passport using an automated system. Multiple entities are involved, including the applicant, passport administrator, regional administrator, police department, and central database. The diagram focuses on how data flows and how verification happens before issuing a passport.

Step-by-Step Process

1. Applicant Submission

- The process begins when the **applicant logs in** to the system.
- The applicant provides personal details needed for the passport application.

2. Passport Administrator Review

- The passport administrator receives the submitted details.
- The administrator verifies the details internally and updates stored information.

3. Forwarding to Regional Administrator

- The passport administrator sends the verified details to the **regional administrator**.
- The regional office verifies the received details and updates the database.

4. Police Verification Stage

- The passport administrator then forwards the application details to the **police department**.
- The police perform background verification and update the system once completed.

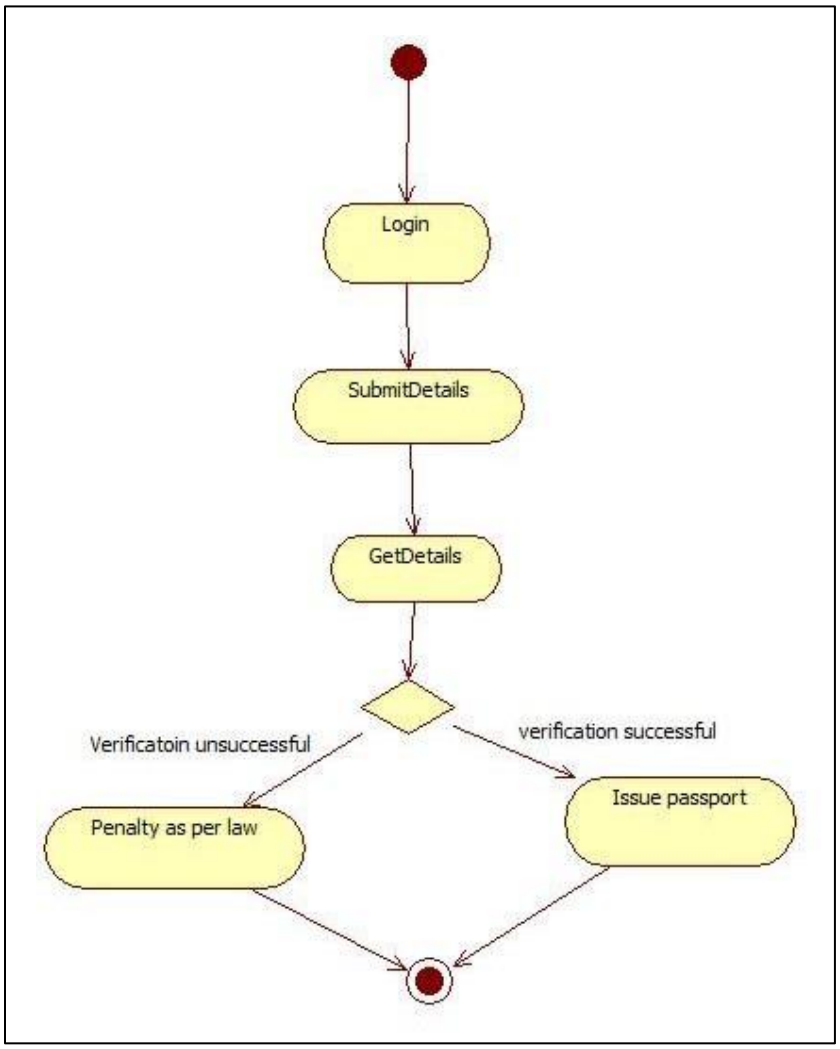
5. Final Approval

- Police verification results are sent back to the passport administrator.
- The administrator updates the database with the final verification decision.

6. Passport Issuance

- Once all verification steps are complete and confirmed, the passport administrator issues the passport to the applicant.

Activity Diagram:



Overview

This activity diagram represents the simplified workflow of applying for a passport. It depicts the main steps from user login to final passport issuance or rejection based on verification results.

Process Flow Description

1. **Start**
    - The process begins with the user accessing the system.
2. **Login**
    - The applicant logs into the system using valid credentials.
3. **Submit Details**
    - After logging in, the applicant submits personal and required identification details.

4. **Get Details**
    - The system receives and retrieves the submitted information for verification.
5. **Verification Decision**
    - A decision point evaluates whether the submitted details meet requirements and are valid.
        - **If verification is successful:**
            - The system proceeds to **Issue Passport**, and the application is approved.
        - **If verification is unsuccessful:**
            - The system applies **Penalty as per law**, which could involve rejection, fines, or legal action based on regulation.
6. **End**
    - The process reaches its final state after either issuing the passport or enforcing a penalty.