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:

Hitha Harish

1BM23CS115

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 Hitha Harish (1BM21CS115) during the 5th Semester August 2025-December 2025

Signature of the Faculty Incharge:
Dr Roopashree S
Associate Professor
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

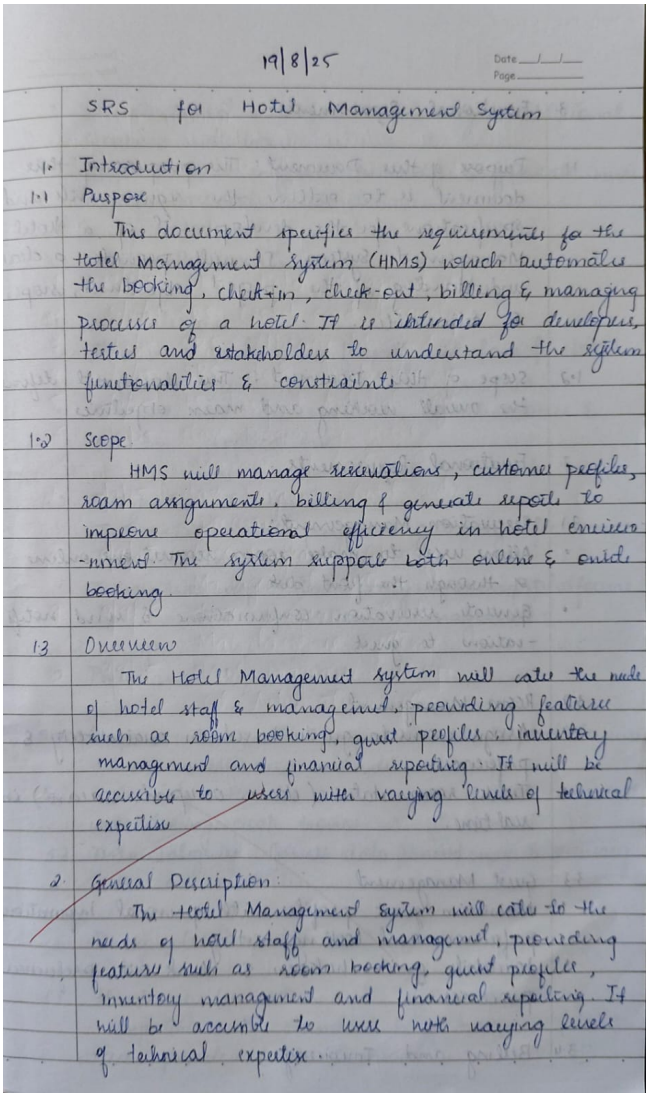
Table of Contents

1.Hotel Management System

Problem Statement

Hotels face challenges in managing reservations, guest check-in/check-out, room availability, billing, and staff operations due to manual or outdated processes. These inefficiencies lead to booking errors, reduced guest satisfaction, and operational delays. A centralized, automated Hotel Management System is required to streamline operations, improve accuracy, and enhance customer service.

Software Requirements Specification (SRS)



19/8/25

SRS for Hotel Management System

1. Introduction

1.1 Purpose

This document specifies the requirements for the hotel management system (HMS) which automates the booking, check-in, check-out, billing & managing processes of a hotel. It is intended for developers, testers and stakeholders to understand the system functionalities & constraints

1.2 Scope

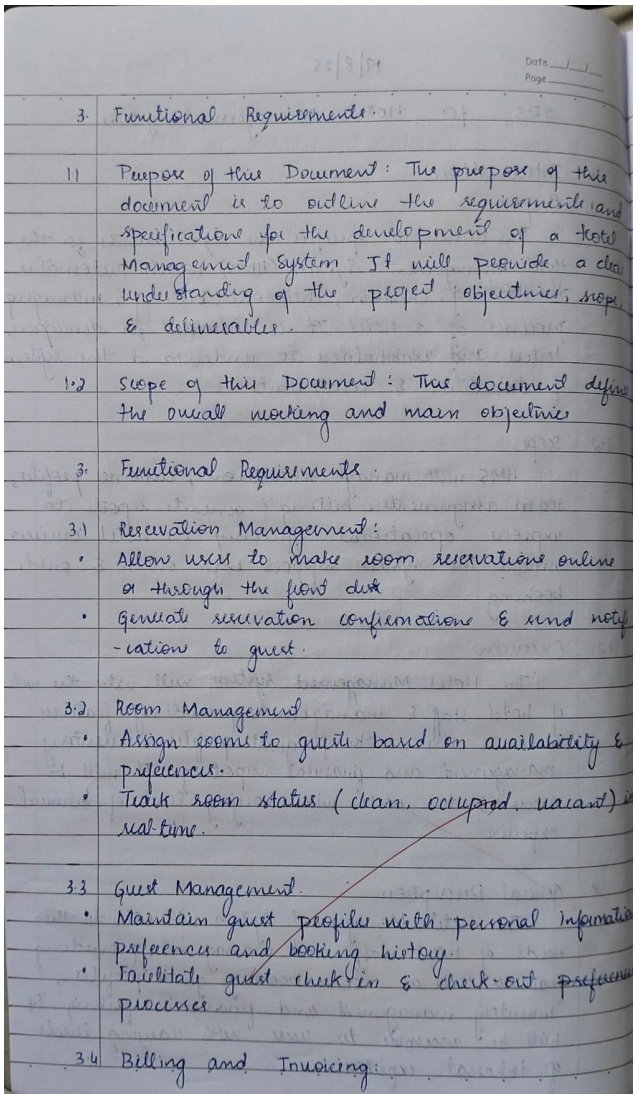
HMS will manage reservations, customer profiles, room assignments, billing & generate reports to improve operational efficiency in hotel environment. The system support both online & onsite booking.

1.3 Overview

The Hotel Management system will cater the needs of hotel staff & 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

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:

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 & deliverables.

1.2 Scope of this Document: This document defines the overall working and main objectives

3. Functional Requirements:

3.1 Reservation Management:

- Allow users to make room reservations online or through the front desk
- Generate reservation confirmations & send notification to guest.

3.2 Room Management

- Assign rooms to guests based on availability & preferences.
- Track room status (clean, occupied, vacant) in real-time.

3.3 Guest Management

- Maintain guest profile with personal information preferences and booking history.
- Facilitate guest check-in & check-out preferences processes.

3.4 Billing and Invoicing:

- Generate accurate bills for room charges, additional services, and taxes.
- Accept various payment methods & generate invoices for corporate clients.

4 Interface Requirements:

4.1 User Interface:
- Intuitive and user-friendly interface for hotel staffs & guests.
- Accessible via web browsers, mobile devices & desktop applications.

4.2 Integration Interfaces:
- Integration with payment gateways for secure transactions.
- Integration with third-party booking platforms for seamless reservation management.

5. Performance Requirements:

5.1 Response Time: The system should respond to user actions within 2 seconds.

5.2 Scalability: Handle a minimum of 1000 concurrent users during peak hours.

5.3 Data Integrity: Ensure data consistency & accuracy across all modules.

6 Design Constraints:

6.1 Hardware limitations: The system should be compatible with standard hotel hardware (computers, printers, POS terminals).

6.2 Software Dependencies:
- Utilize a relational database management system for data storage.
- Use programming languages & frameworks conducive to modeling (Java, Spring Boot).

7. Non-Functional Attributes:

7.1 Security: Implement robust authentication & authorization mechanisms to protect sensitive data.

7.2 Reliability: Ensure high availability & fault tolerance to minimize system downtime.

7.3 Scalability: Design the system to accomodate future growth & expansion.

7.4 Portability: Support multiple platforms & devices for user accessibility.

7.5 Usability: The system shall have a user-friendly interface with clear navigation.

7.6 Reusability: The system shall use modular code design to facilitate future enhancements & maintenance.

7.7 Compatibility: The system shall be compatible with common web browsers (Chrome, Firefox).

7.8 Data Integrity: The system shall ensure accurate & consistent data storage & retrieval.

8. Preliminary Schedule & Budget: The development of the HMS is estimated to take 6 months with a budget of $100,000. This includes project planning, development, testing & deployment phases.

Class Diagram:

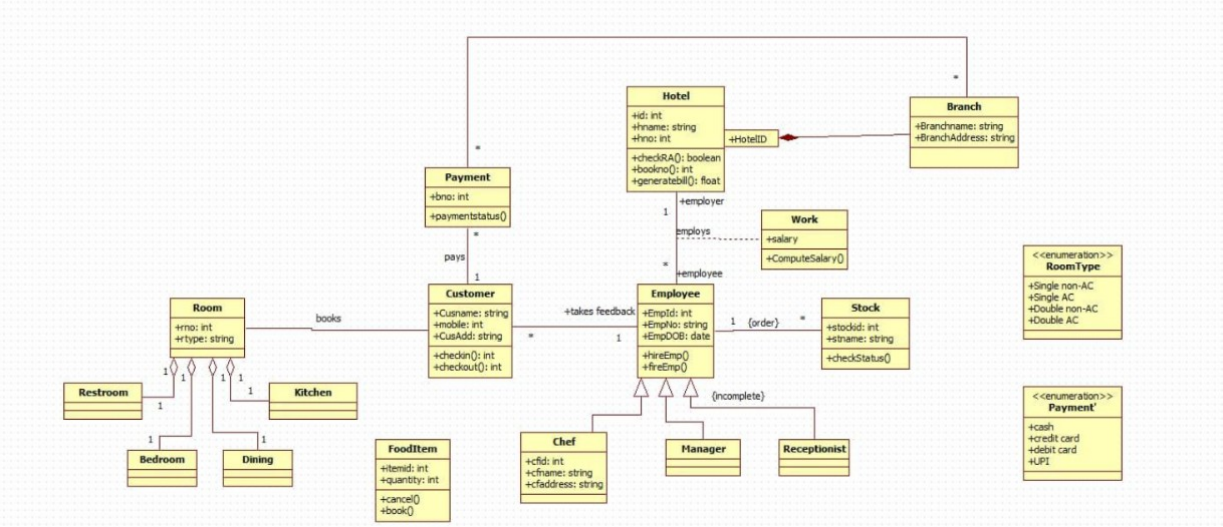


Fig 1.1 Class diagram of Hotel Management System

The diagram shows a Hotel Management System where a hotel has multiple branches, rooms, customers, employees, and payments. Customers book rooms and make payments. Employees (like chefs, managers, and receptionists) work for the hotel and handle tasks such as food orders and customer service. Rooms have different types and facilities, while stock and food items are also managed within the system.

State Diagram:

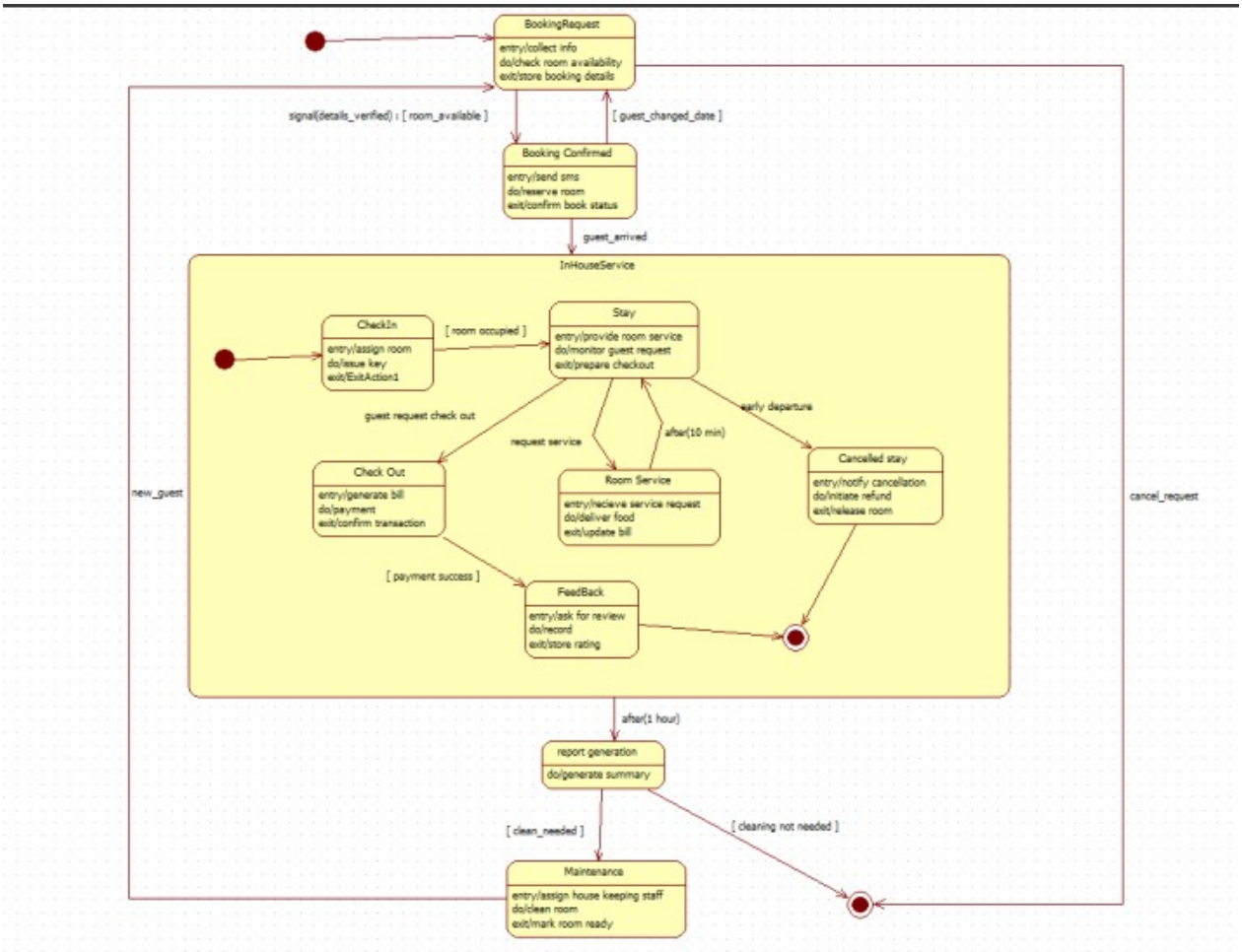


Fig 1.2 State diagram of Hotel Management System

The state diagram shows the lifecycle of a hotel booking and stay. It begins with a Booking Request, which becomes Booking Confirmed if a room is available. When the guest arrives, they move through states such as Check-in, Stay, requesting Room Service, giving Feedback, and finally Check-out. A guest may also enter a Cancelled Stay if the booking is canceled. After checkout, the system generates reports, and rooms may go into Maintenance if cleaning is needed.

Use-Case Diagram:

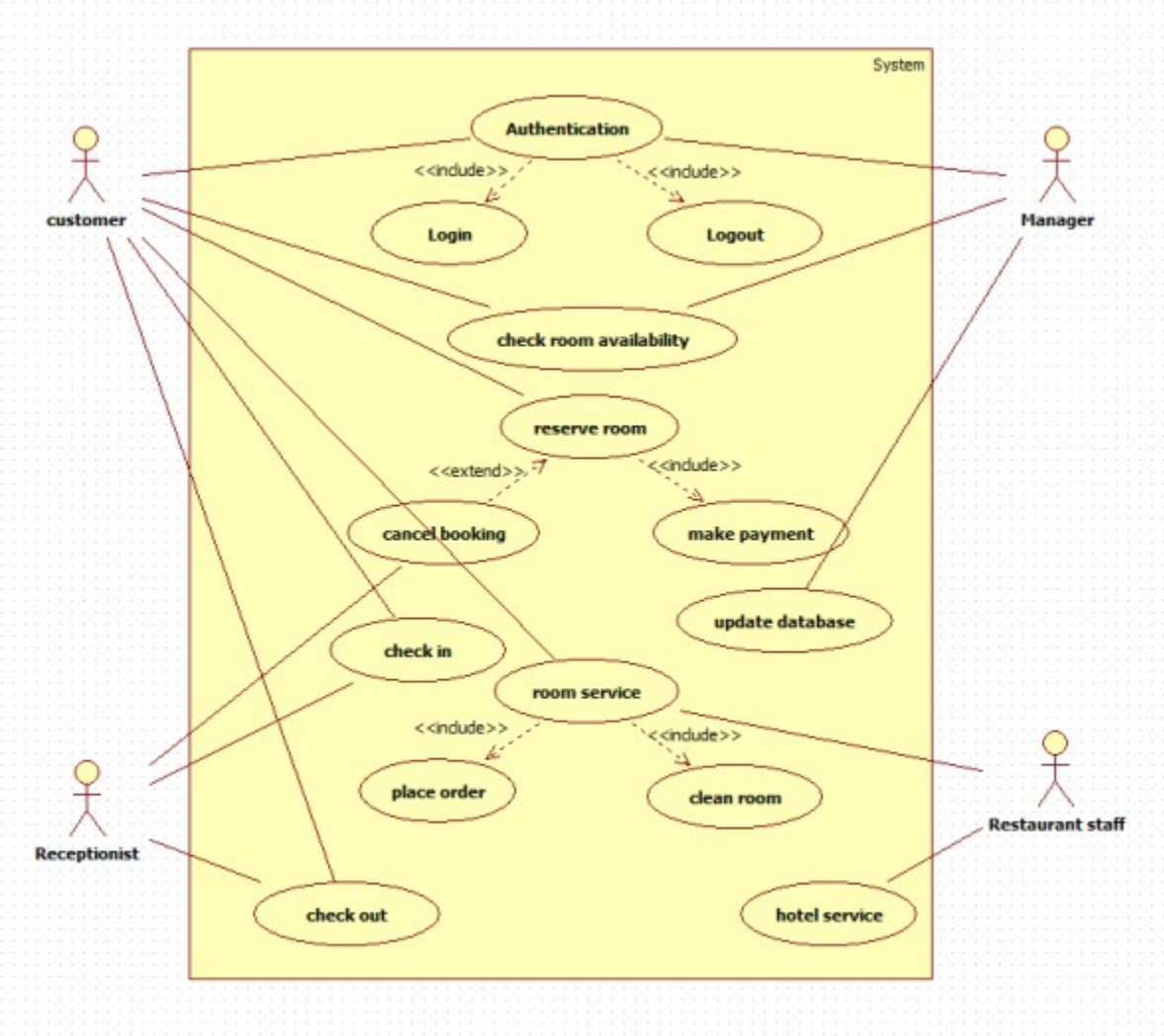


Fig 1.3 Use case diagram of Hotel Management System

The use-case diagram shows how different users interact with the Hotel Management System. Customers can log in, check room availability, reserve or cancel bookings, make payments, check in, request room service, and check out. Receptionists handle check-ins, check-outs, and room service tasks like placing orders or cleaning rooms. Managers can authenticate, update the database, and manage reservations. Restaurant staff provide hotel and food service support. The system centralizes all these actions to streamline hotel operations

Use-Case Diagram:

Sequence Diagram:

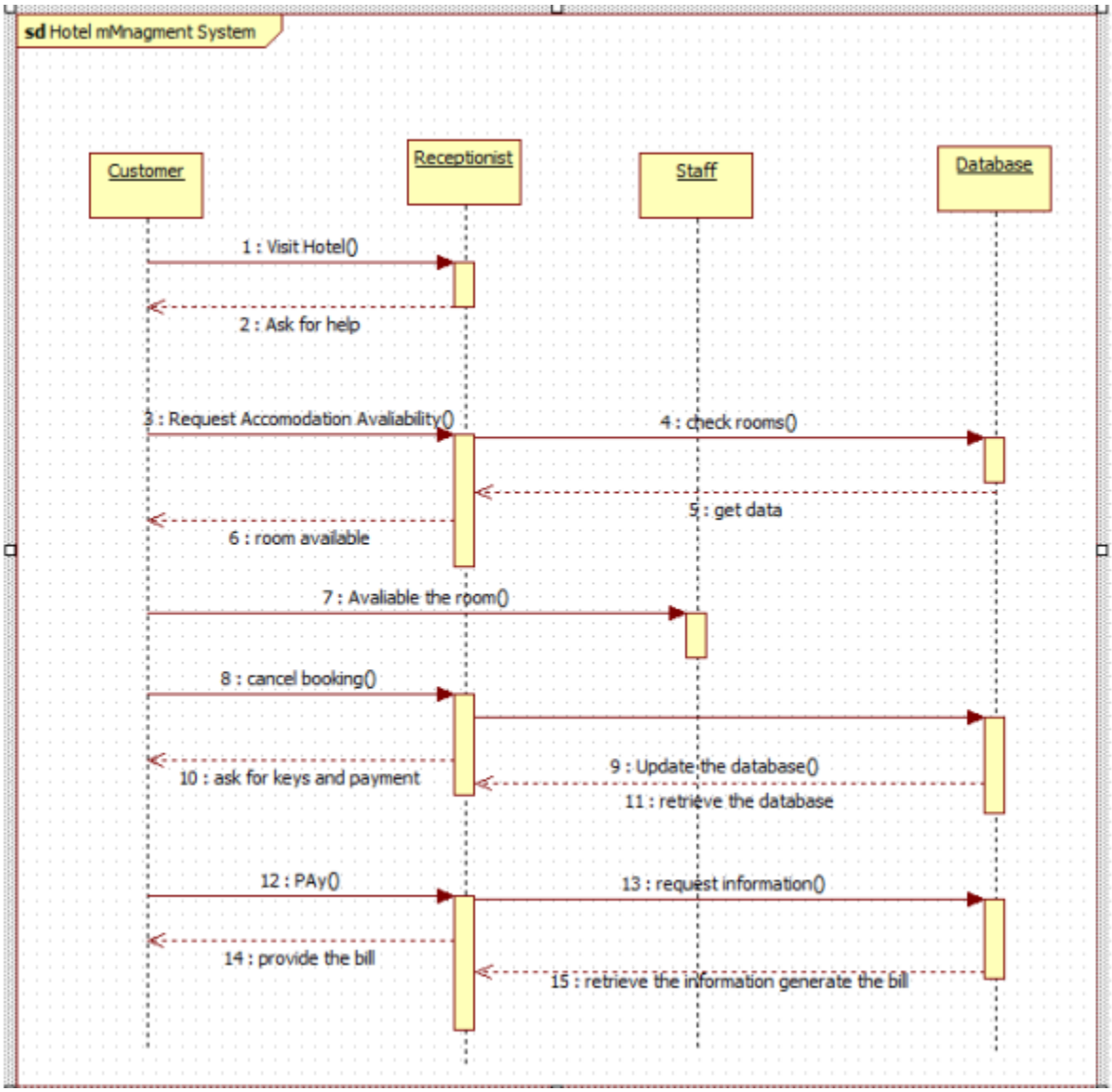


Fig 1.4 Sequence diagram of Hotel Management System

The sequence diagram shows the interaction between the customer, receptionist, staff, and database during a hotel booking process. The customer visits the hotel, requests accommodation, and the receptionist checks room availability through the staff and database. Once availability is confirmed, the customer can proceed to book or cancel. The receptionist updates the database, collects payment, and requests billing information. Finally, the receptionist provides the bill to the customer.

Activity diagram:

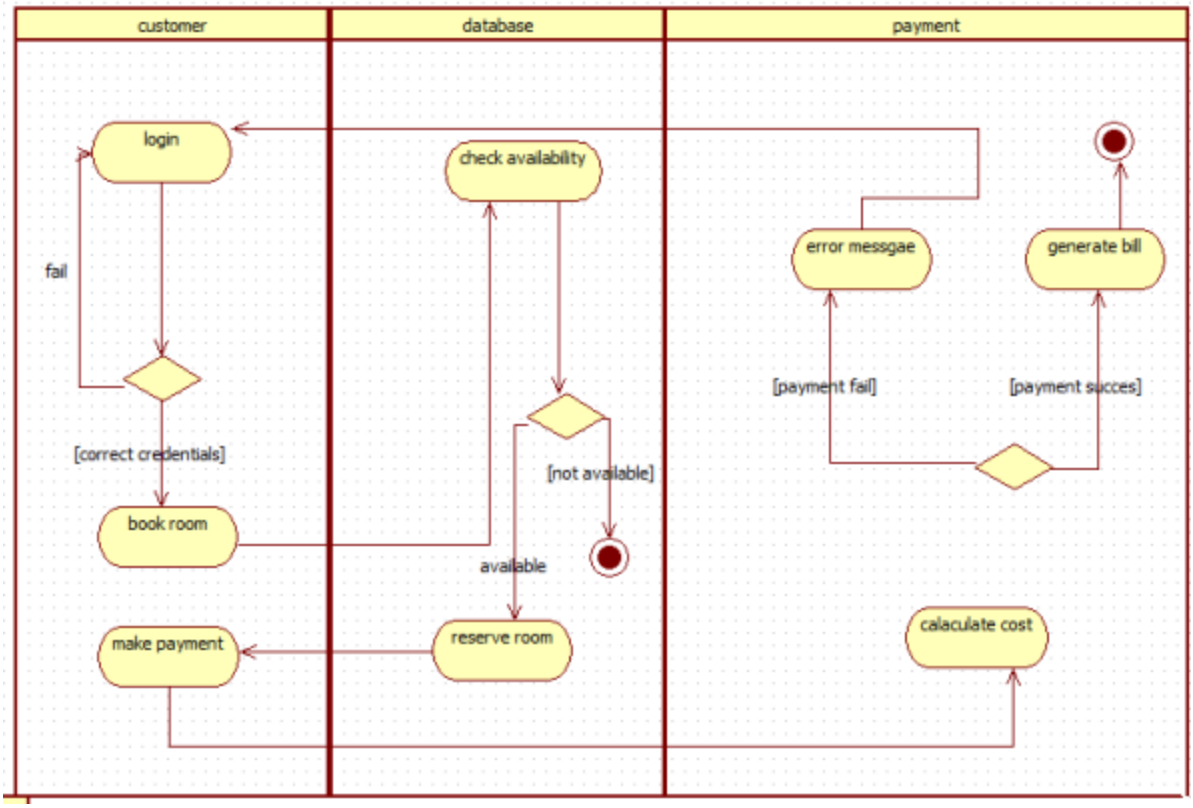


Fig 1.5 Activity diagram of Hotel Management System

The activity diagram shows the steps a customer follows to book a room in the hotel system. The customer logs in, and the database checks room availability. If a room is available, the customer proceeds to book and make a payment. The payment system calculates the cost and either generates the bill for a successful payment or shows an error message if the payment fails.

2. Credit Card Automation System

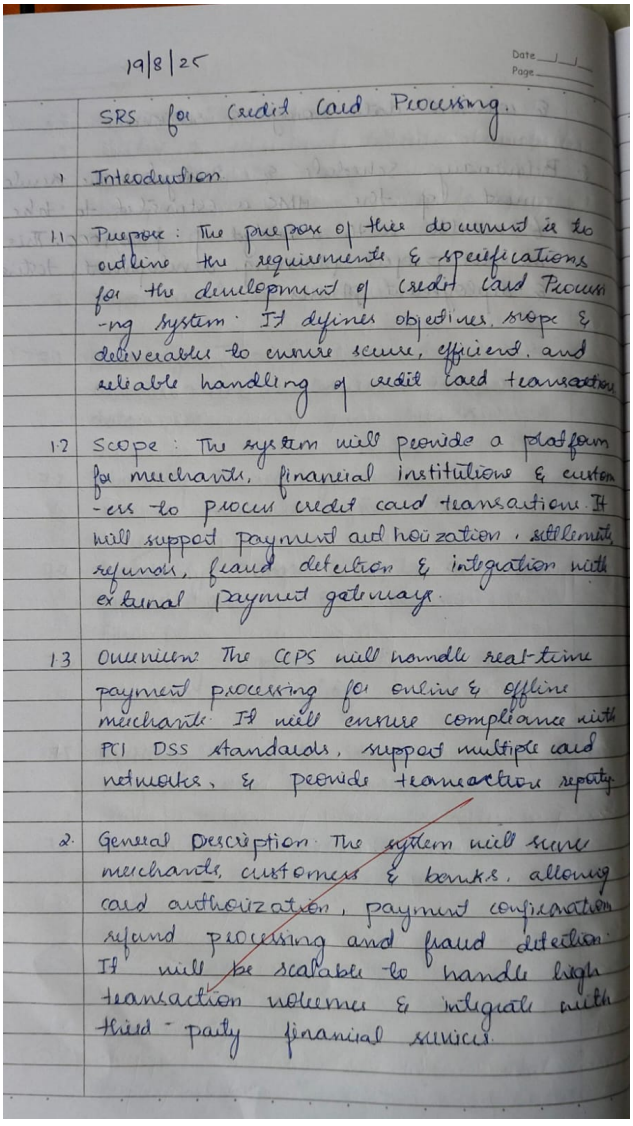
Activity diagram:

Problem Statement

Manual credit card processing—including application approval, transaction handling, billing, payments, and fraud detection—leads to delays, errors, and security risks. As customers and transactions increase, managing these tasks becomes more difficult for banks.

A Credit Card Automation System is required to automate account creation, real-time transaction processing, billing cycles, payment updates, fraud alerts, and reporting. The goal is to improve speed, accuracy, security, and overall efficiency through a centralized, secure, and user-friendly platform.

Software Requirements Specification (SRS)


19/8/25

SRS for Credit Card Processing.

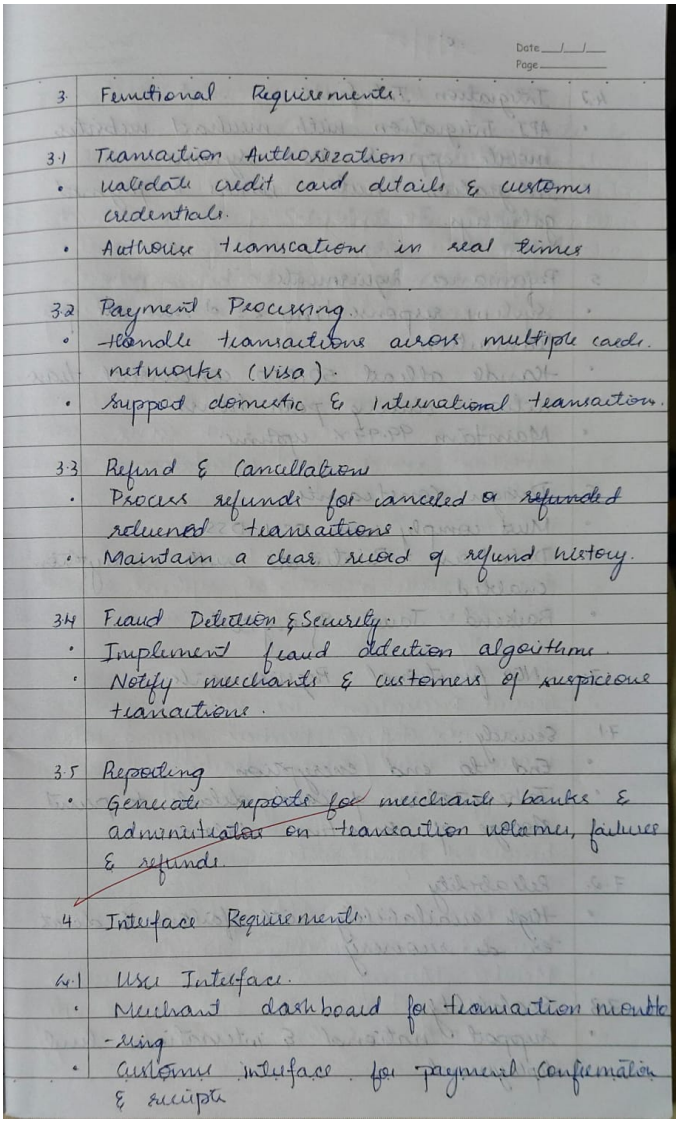
1. Introduction

1.1 Purpose: The purpose of this document is to outline the requirements & specifications for the development of credit card Process-ing system. It defines objectives, scope & deliverables to ensure secure, efficient, and reliable handling of credit card transactions.

1.2 Scope: The system will provide a platform for merchants, financial institutions & custom-ers to process credit card transactions. It will support payment authorization, settlement, refunds, fraud detection & integration with external payment gateways.

1.3 Overview: The CCPS will handle real-time payment processing for online & offline merchants. It will ensure compliance with PCI DSS standards, support multiple card networks, & provide transaction reports.

2. General Description: The system will serve merchants, customers & banks, allowing card authorization, payment confirmation, refund processing and fraud detection. It will be scalable to handle high transaction volumes & integrate with third-party financial services.


3. Functional Requirements:

3.1 Transaction Authorization
- Validate credit card details & customer credentials.
- Authorize transaction in real time.

3.2 Payment Processing
- Handle transactions across multiple card networks (Visa).
- Support domestic & international transactions.

3.3 Refund & Cancellation
- Process refunds for canceled or refunded returned transactions.
- Maintain a clear record of refund history.

3.4 Fraud Detection & Security.
- Implement fraud detection algorithms.
- Notify merchants & customers of suspicious transactions.

3.5 Reporting
- Generate reports for merchants, banks & administrators on transaction volumes, failures & refunds.

4. Interface Requirements

4.1 User Interface.
- Merchant dashboard for transaction monitor-ing.
- Customer interface for payment confirmation & receipts.

4.2 Integration Interfaces.

- API Integration with merchant websites, mobile apps & POS systems.
- Integration with banks & payment gateways.

5. Performance Requirements.

- System response time <= 2 seconds per transaction.
- Handle atleast 50,000 concurrent transactions during peak hours.
- Maintain 99.99% uptime.

6. Design Constraints.

- Must comply with PCI DSS.
- Database: Relational with encryption enabled.
- Backend: Java, Spring Boot.

7. Non functional Requirements.

7.1 Security:

- End to end encryption.
- Tokenization of card details to prevent storage of sensitive data.

7.2 Reliability

- High availability with failover & disaster recovery.

7.3 Scalability

- Support national & international fund deployment.

7.4 Usability

- Easy-to-use dashboard for merchants & clear payment flow for customers.

8. Preliminary Schedule & Budget

- Development Duration: 10 months.
- Estimated Budget: $ 400,000.

Class Diagram:

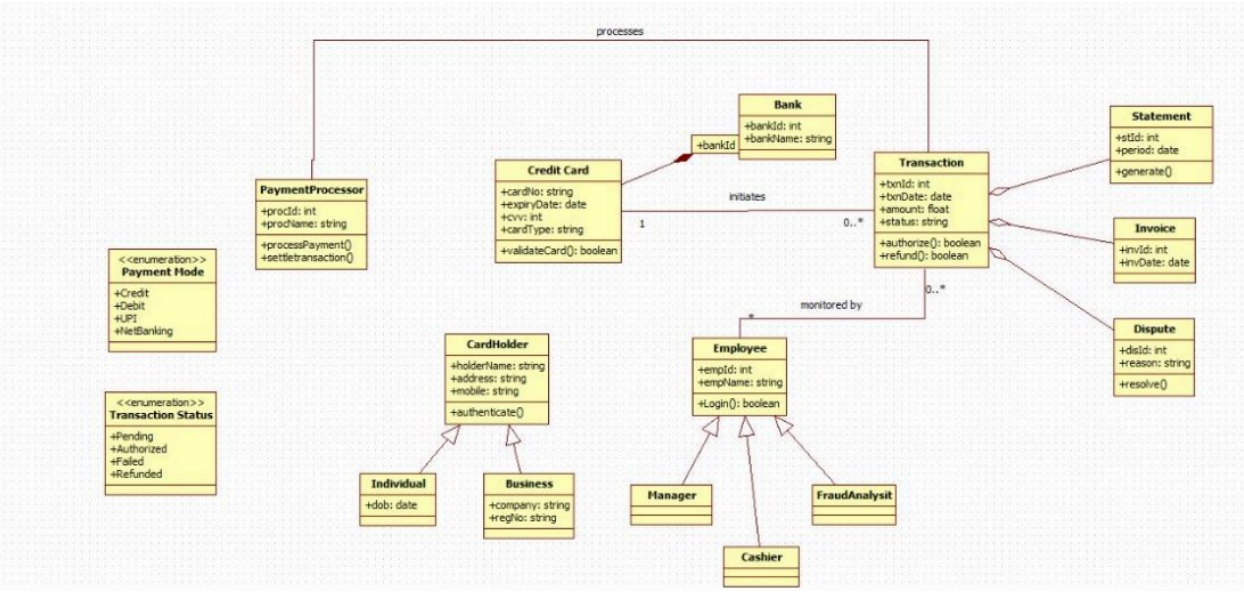


Fig 2.1 Class diagram of Credit Card Processing System

The class diagram represents a payment processing system that manages credit card transactions. A PaymentProcessor handles payment modes (like credit, debit, UPI, net banking) and interacts with CreditCard details, which are validated before initiating a transaction. Each Transaction is linked to a Bank and can generate related documents such as Statements, Invoices, or Disputes. Transactions carry statuses like pending, authorized, failed, or refunded.

A CardHolder (either an individual or business) owns the credit card and can authenticate themselves. Employees—including managers, cashiers, and fraud analysts—monitor transactions and log into the system. Overall, the diagram shows how payments flow from cardholder verification to bank interaction, transaction processing, and post-transaction documentation.

State diagram:

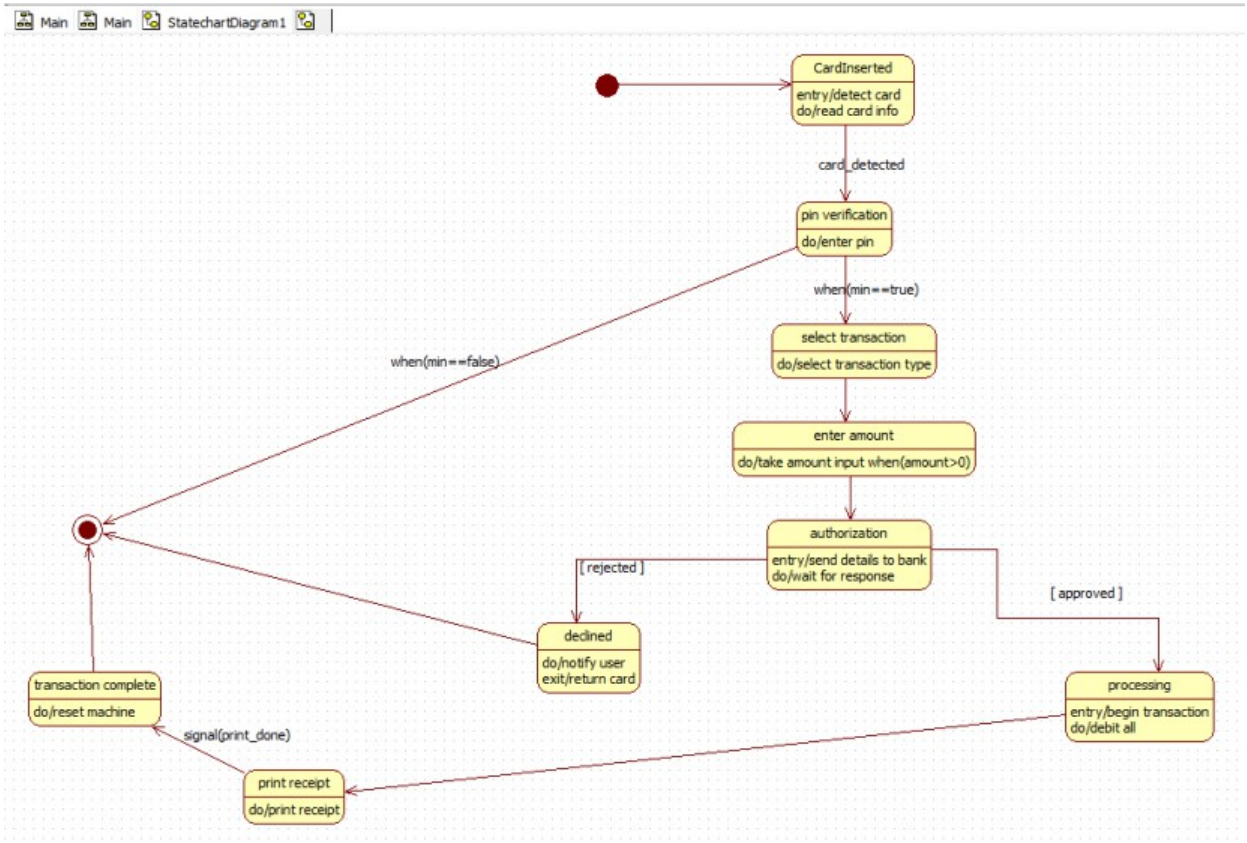


Fig 2.2 State diagram of Credit Card Processing System

The state diagram illustrates the workflow of a card-based transaction process in a payment machine (like an ATM or POS). The process begins when a card is inserted and detected, followed by PIN verification. If the PIN is valid, the user selects a transaction type and enters an amount. The machine then sends the transaction details to the bank for authorization.

If the bank approves the request, the system proceeds to process the transaction and later prints a receipt. If the request is rejected, the machine notifies the user and returns the card. In both scenarios, the process ends with the machine resetting itself and returning to the initial state, ready for the next transaction.

Use-case diagram:

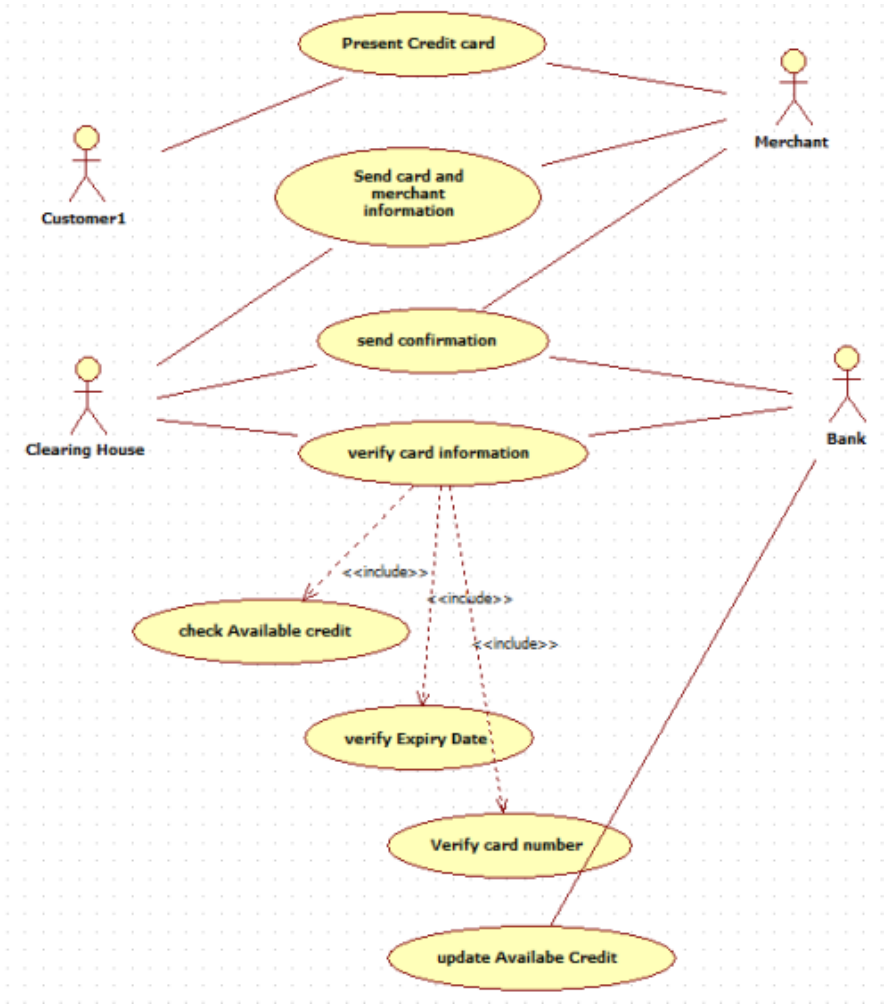


Fig 2.3 Use-case diagram of Credit Card Processing System

The use case diagram illustrates the overall credit card processing workflow involving the Customer, Merchant, Clearing House, and Bank. The process begins when the customer presents their credit card to the merchant, who sends the card and merchant details for verification. The Clearing House and Bank validate the card through a series of checks, including verifying the card number, expiry date, and available credit.After successful verification, the system updates the available credit and sends a confirmation back to the merchant. This enables the merchant to complete the transaction. The diagram highlights the interaction between different actors and the essential verification steps involved in processing a credit card payment.

Sequence diagram:

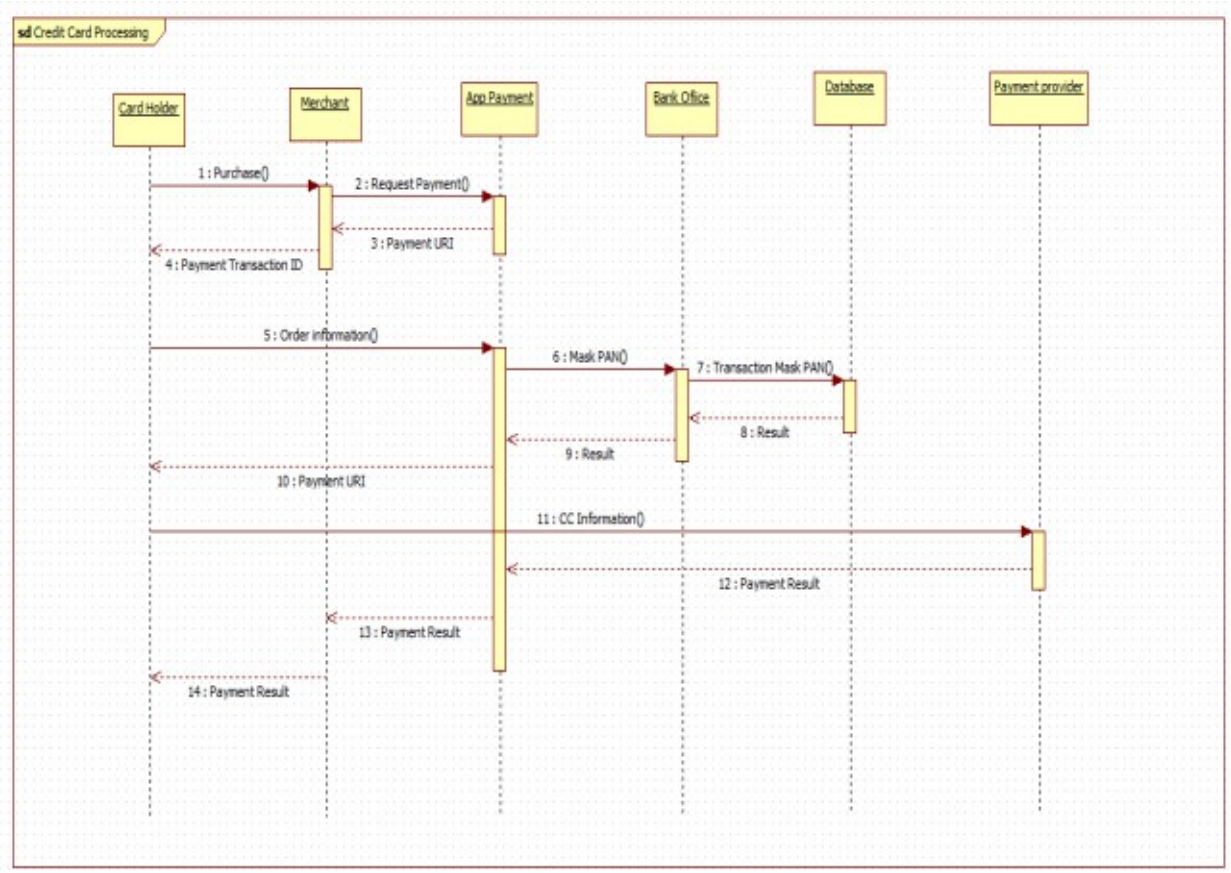


Fig 2.4 Sequence diagram of Credit Card Processing System

The sequence diagram shows the flow of a credit card payment from purchase to completion. The Card Holder initiates a payment, the Merchant sends the request, and the App Payment system processes the order by masking card details and communicating with the Bank Office and Database for verification. After validation, the payment result is returned through the App Payment system to the Merchant, Payment Provider, and finally to the Card Holder.

Activity diagram:

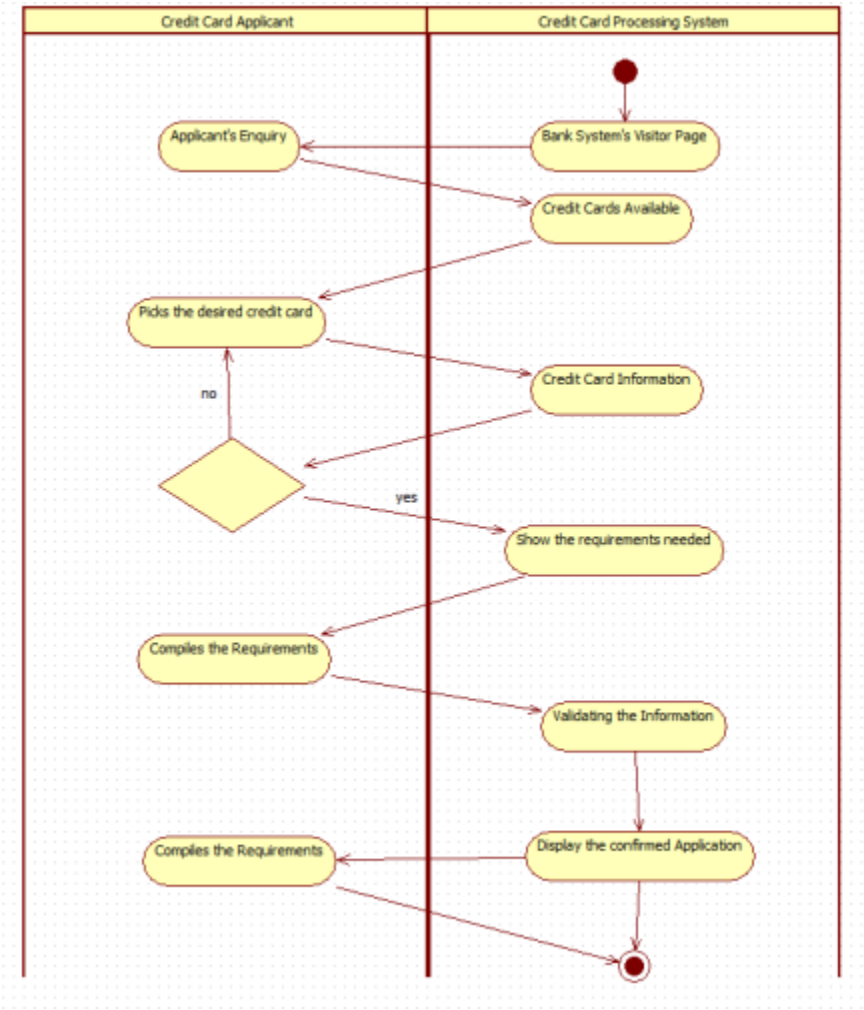


Fig 2.5 Activity diagram of Credit Card Processing System

The activity diagram illustrates the process of a credit card applicant applying through the bank's credit card processing system. The applicant sends an enquiry, views available credit cards, and selects one. The system displays the card information and required documents. After the applicant submits the necessary requirements, the system validates the information and finally displays the confirmed application.

3. Library Management System

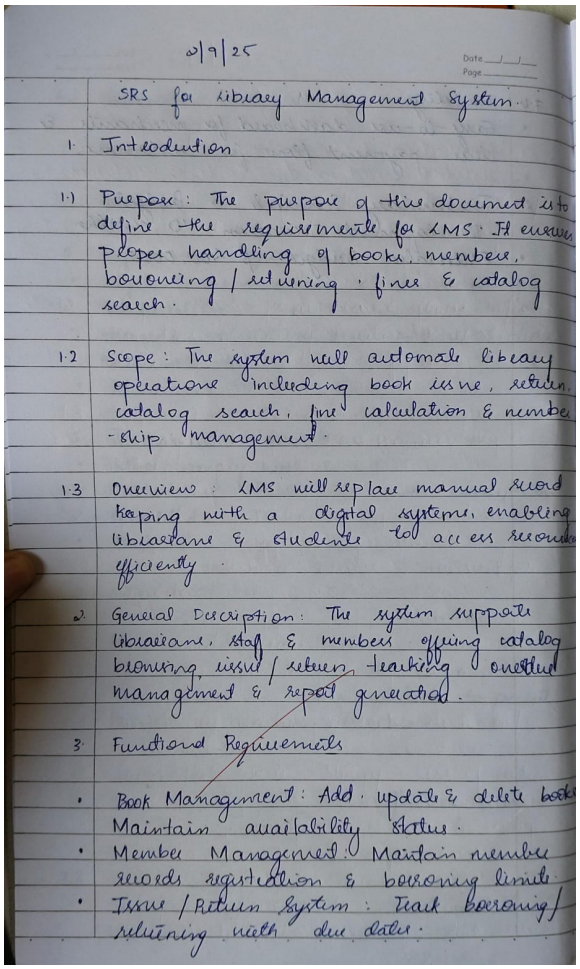
Activity diagram:

Problem Statement

Traditional library operations such as book issuing, returning, cataloging, fine calculation, and record maintenance are often done manually, leading to errors, delays, and difficulty in tracking books and users. As the number of books and members increases, managing inventory, reducing book loss, and maintaining accurate records becomes challenging.

A Library Management System is needed to automate book management, user registration, transactions, and report generation. The goal is to improve efficiency, accuracy, and accessibility for librarians and users. The system should provide quick search, real-time inventory updates, and role-based access while ensuring secure data handling.

Software Requirements Specification (SRS)



SRS for library Management System.

1. Introduction

1.1 Purpose: The purpose of this document is to define the requirements for LMS. It ensures proper handling of books, members, borrowing / returning, fines & catalog search.

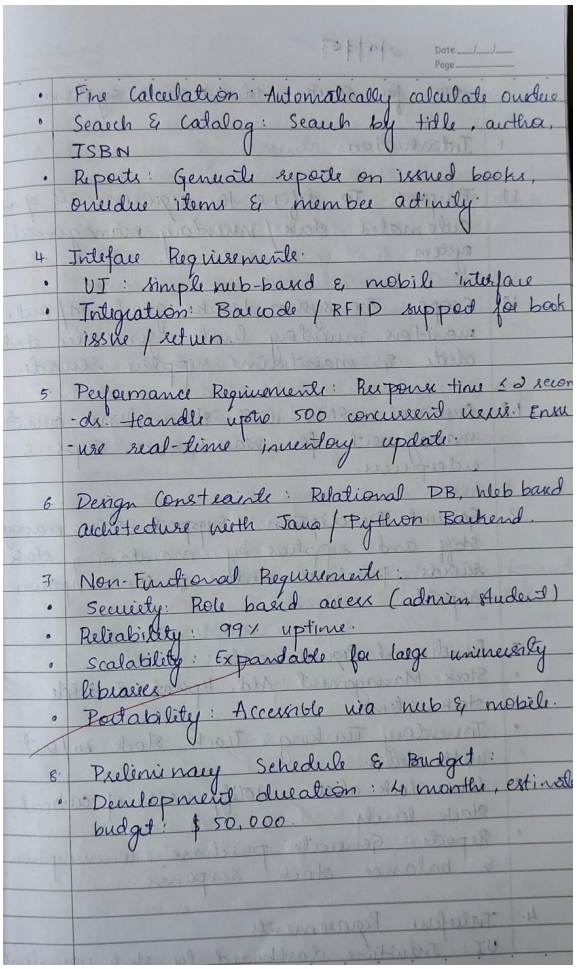
1.2 Scope: The system will automate library operations including book issue, return, catalog search, fine calculation & membership management.

1.3 Overview: LMS will replace manual record keeping with a digital system, enabling librarians & students to access records efficiently.

2. General Description: The system supports librarians, staff & members offering catalog browsing, issue / return tracking, overdue management & report generation.

3. Functional Requirements

- Book Management: Add, update & delete books. Maintain availability status.
- Member Management: Maintain member records, registration & borrowing limits.
- Issue / Return System: Track borrowing / returning with due dates.



- Fine Calculation: Automatically calculate overdue
- Search & Catalog: Search by title, author, ISBN
- Reports: Generate reports on issued books, overdue items & member activity.

4. Interface Requirements:

- UI: Simple web-based & mobile interface
- Integration: Barcode / RFID support for book issue / return

5. Performance Requirements: Response time < 2 seconds. Handle upto 500 concurrent users. Ensure real-time inventory update.

6. Design Constraints: Relational DB, Web based architecture with Java / Python Backend.

7. Non-Functional Requirements:

- Security: Role based access (admin, student)
- Reliability: 99% uptime.
- Scalability: Expandable for large university libraries.
- Portability: Accessible via web & mobile.

8. Preliminary Schedule & Budget:

- Development duration: 4 months, estimated budget: $ 50,000.

Class diagram:

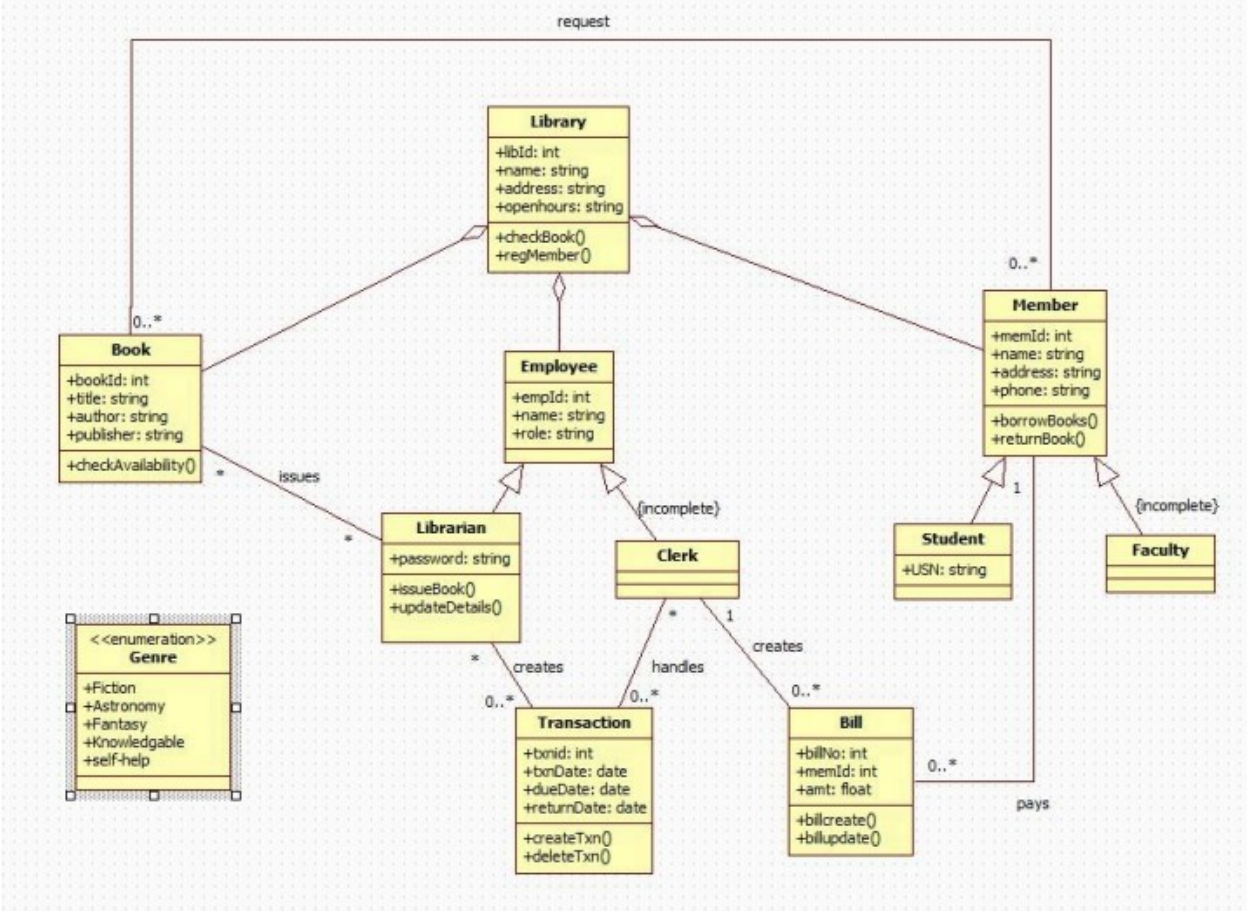


Fig 3.1 Class diagram of Library management system

The class diagram represents a library management system involving books, members, employees, and transactions. The Library maintains books and registered members. Members (students or faculty) can borrow and return books, while employees (librarians and clerks) manage book issuing, record updates, and transaction handling. Each transaction records borrowing and return dates, and bills are generated for any dues. The diagram also includes book genres and shows how different roles interact to support the overall library operations.

State diagram:

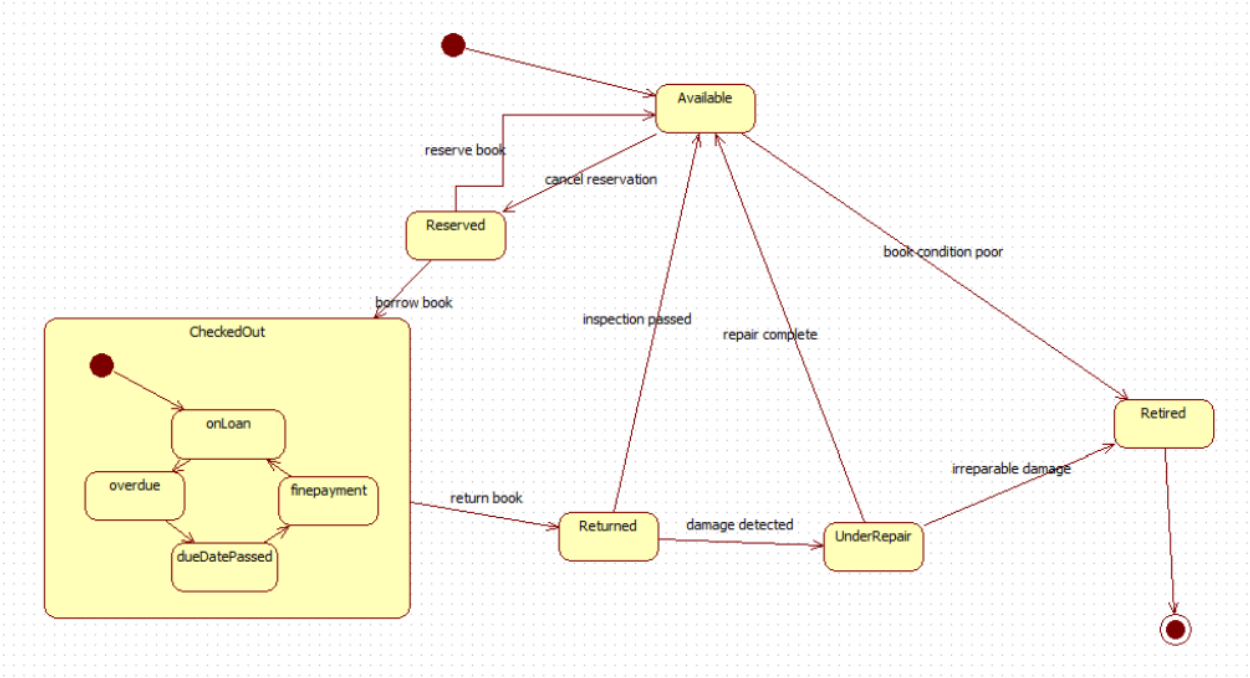


Fig 3.2 State diagram of Library management system

The state diagram shows the lifecycle of a library book. A book starts in the Available state and can be Reserved or Checked Out. When checked out, it may become overdue or require fine payment before return. Once returned, the book is inspected—if damaged, it goes Under Repair; if repair is completed, it becomes available again. If the book is irreparably damaged or in poor condition, it transitions to the Retired state and is removed from circulation.

Use-case diagram:

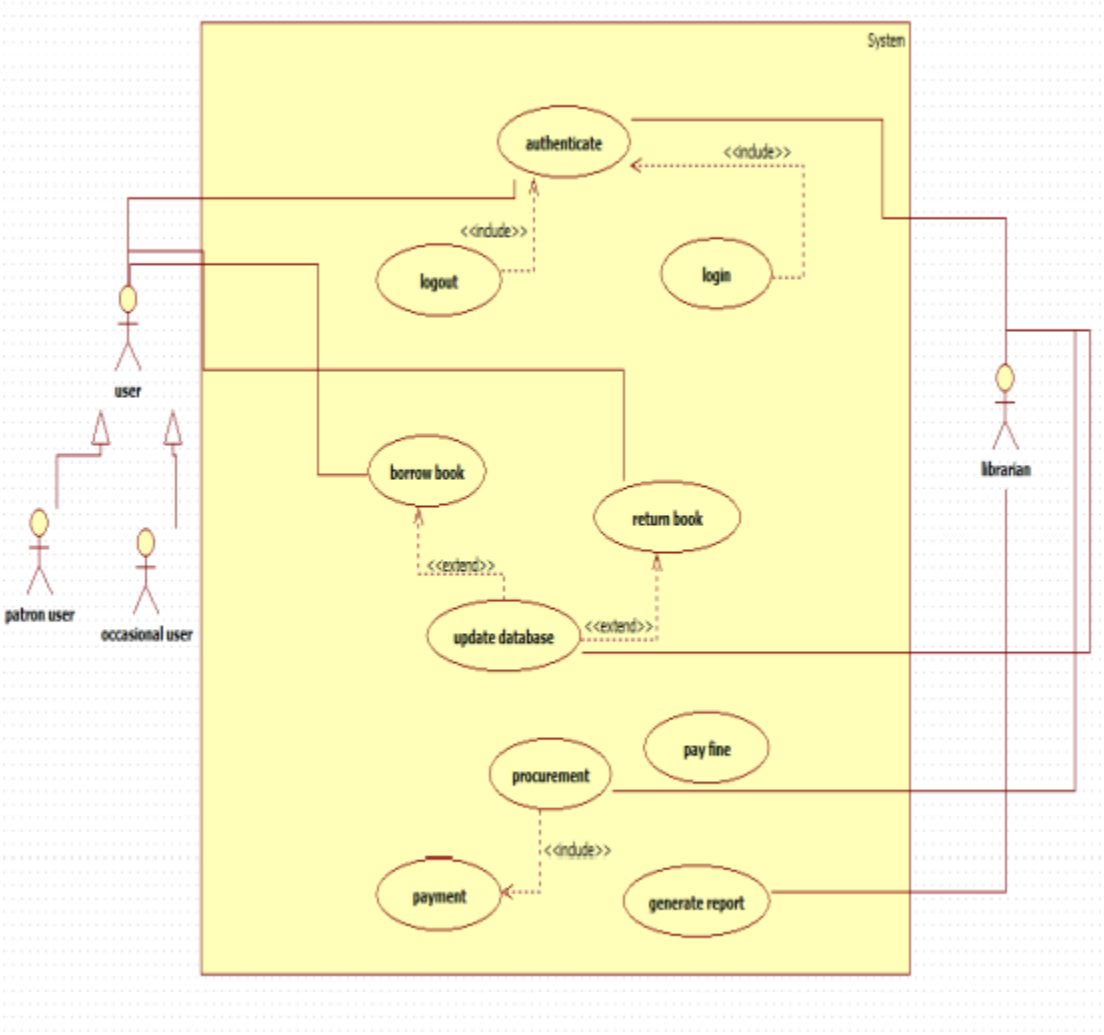


Fig 3.3 Use case diagram of Library management system

The use case diagram represents a library management system where users (patron and occasional users) and librarians interact with various system functions. Users can authenticate, log in, borrow books, return books, pay fines, and make payments. Librarians manage procurement, generate reports, and update the database. Several use cases include or extend others, showing how core actions like authentication and data updates support the overall library operations.

Sequence diagram:

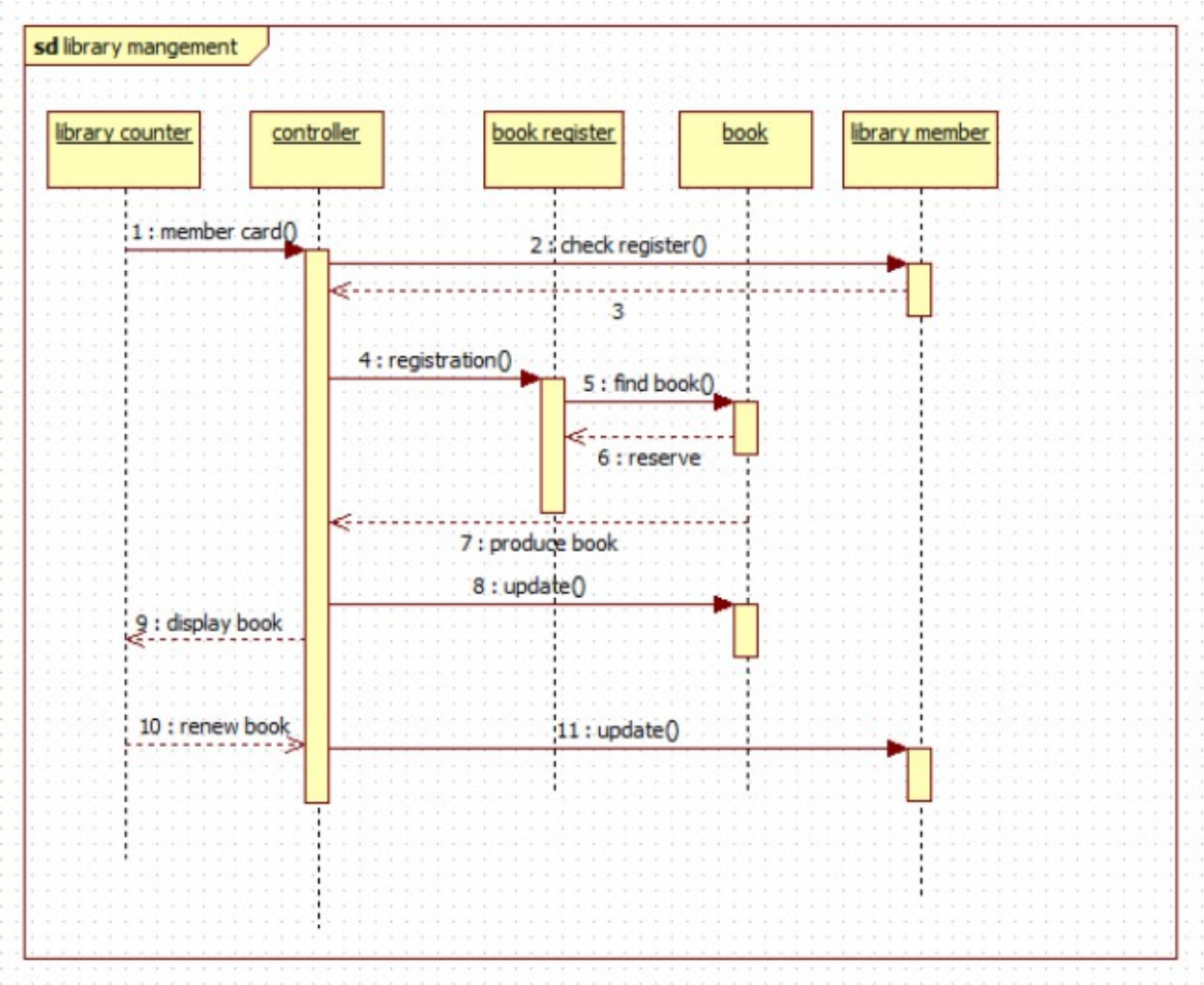


Fig 3.4 Sequence diagram of Library management system

The sequence diagram shows the process of managing book transactions in a library. A library member presents their card at the library counter, which triggers the controller to check the book register, register the member, and search for the requested book. Once the book is found and reserved, it is produced for the member. The system updates the book's status, displays it to the member, and also handles book renewal requests, updating the records accordingly.

Activity diagram:

Fig 3.5 Activity diagram of Library management system

The activity diagram outlines the process of issuing a book in a library. It begins with a user inquiring about a book and checking its availability. If available, the system validates the member; if not registered, the member is asked to register. The system then checks whether the member has exceeded the maximum number of issued books. If eligible, the book is issued, the member's issue details are added, and the book issue record is updated. Otherwise, the book is not issued.

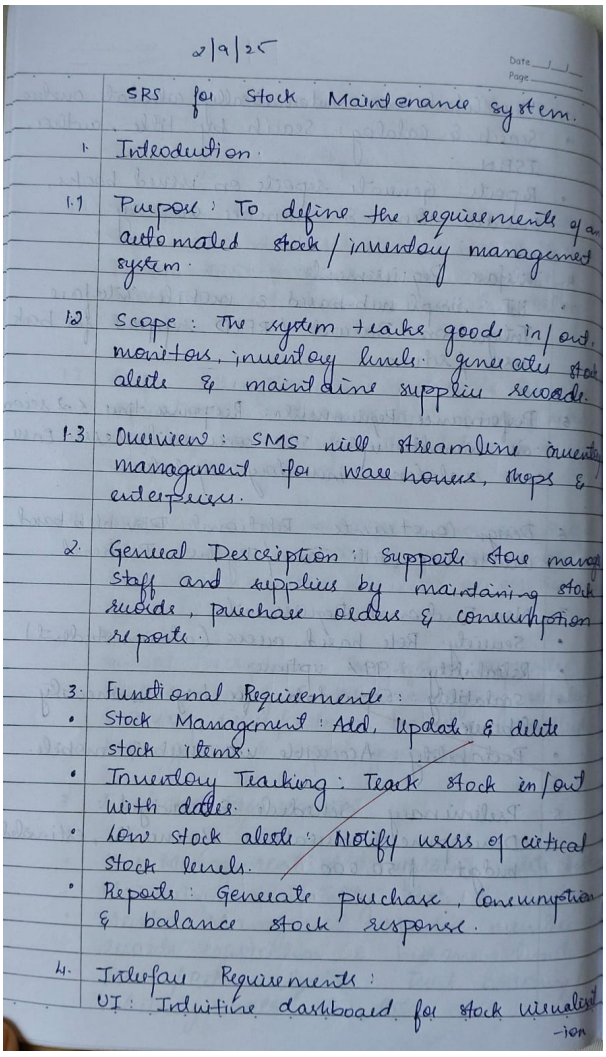
4. Stock Maintenance System

Problem Statement

Many businesses still rely on manual methods to record and track stock levels, purchases, and sales. This often leads to errors, mismatched inventory counts, delays in restocking, and difficulty in monitoring product movement. As stock increases, it becomes harder to maintain accurate records and prevent shortages or overstocking.

A Stock Maintenance System is needed to automate inventory tracking, update stock levels in real time, and generate reports on stock usage, shortages, and reorder requirements. The goal is to ensure accuracy, reduce manual errors, improve efficiency, and help businesses make better inventory decisions through a centralized and user-friendly system.

Software Requirements Specification (SRS)



2/9/25

SRS for Stock Maintenance system.

1. Introduction.

1.1 Purpose: To define the requirements of an automated stock / inventory management system.

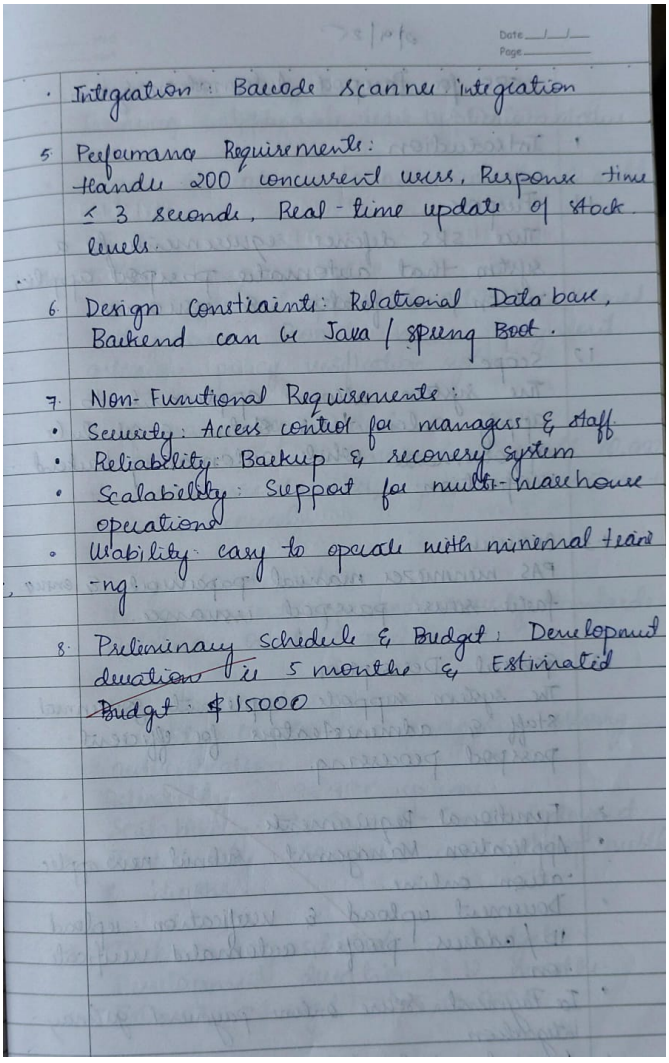
1.2 Scope: The system tracks good in/out, monitors inventory levels, generates stock alerts & maintains supplier records.

1.3 Overview: SMS will streamline inventory management for warehouses, shops & enterprises.

2. General Description: Support store managers staff and suppliers by maintaining stock records, purchase orders & consumption reports.

3. Functional Requirements:
- Stock Management: Add, Update & delete stock items.
- Inventory Tracking: Track stock in/out with dates.
- Low stock alerts: Notify users of critical stock levels.
- Reports: Generate purchase, Consumption & balance stock response.

4. Interface Requirements:
UI: Intuitive dashboard for stock visualiz-ion



- Integration: Barcode Scanner integration

5. Performance Requirements:
Handle 200 concurrent users, Response time < 3 seconds, Real-time update of stock levels.

6. Design Constraints: Relational Database, Backend can be Java / spring Boot.

7. Non-Functional Requirements:
- Security: Access control for managers & staff
- Reliability: Backup & recovery system
- Scalability: Support for multi-warehouse operations
- Usability: easy to operate with minimal train-ing.

8. Preliminary Schedule & Budget: Development duration is 5 months & Estimated Budget: $15000

Class diagram:

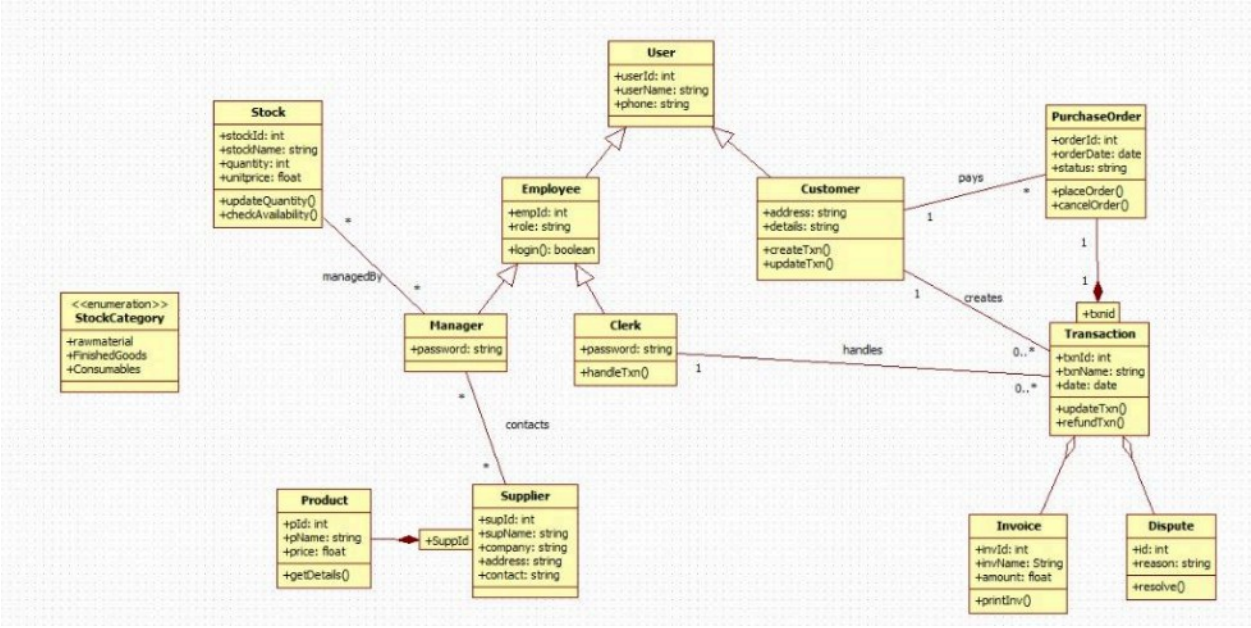


Fig 4.1 Class diagram of Stock management system

The class diagram represents an inventory and transaction management system involving users, employees, customers, suppliers, products, and stock. Customers create purchase orders, which generate transactions handled by clerks. Stock items belong to specific categories and are managed by managers who update quantity and availability. Suppliers provide products linked to stock, while invoices and disputes are associated with transactions for payment and issue resolution. Overall, the diagram shows how different entities interact to support purchasing, stock management, and transaction processing.

State diagram:
Class diagram:

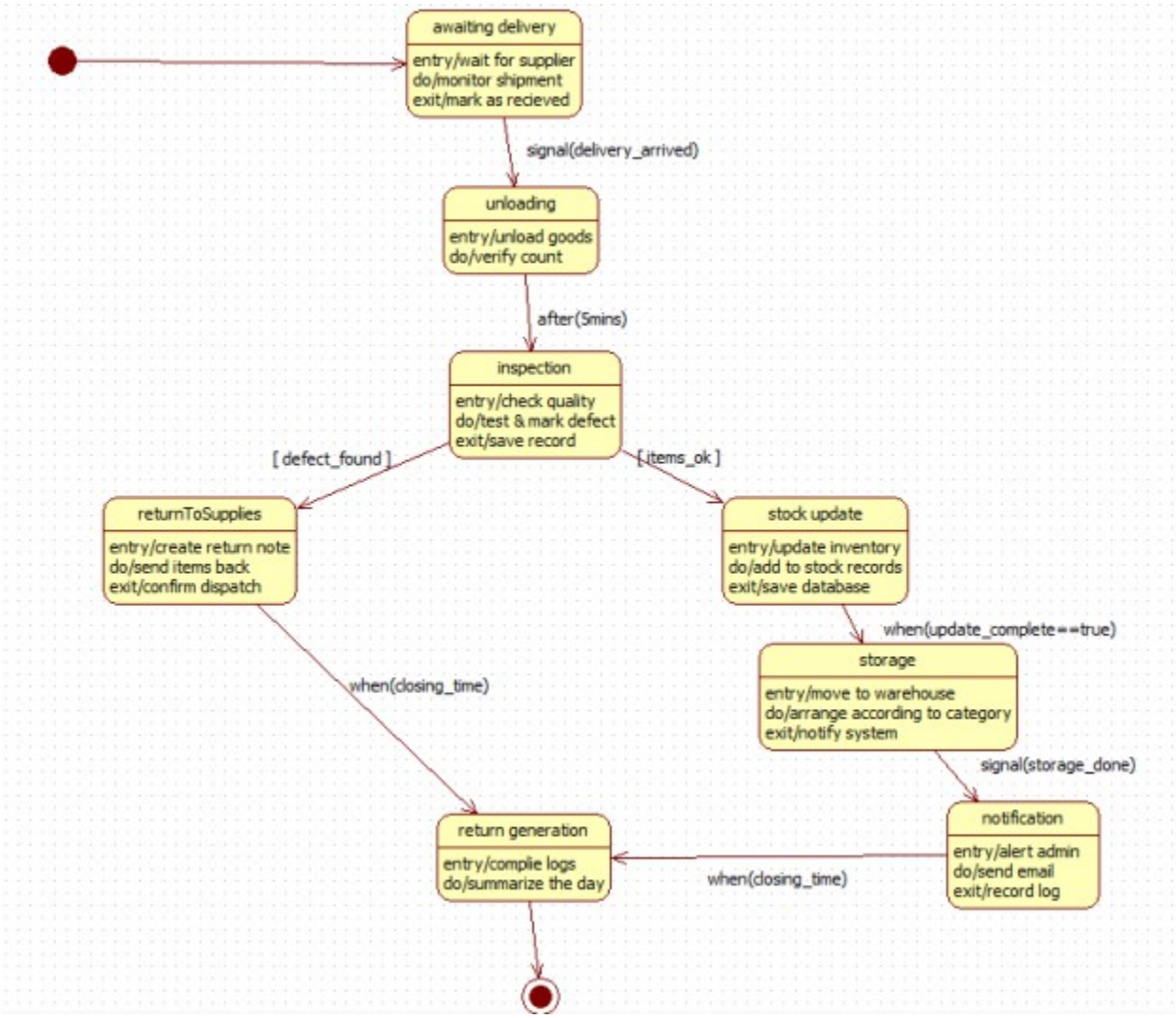


Fig 4.2 State diagram of Stock management system

The state diagram models the workflow of handling incoming goods in an inventory system. The process starts with awaiting delivery, followed by unloading and inspecting the goods. If defects are found, items are returned to the supplier; if the goods are acceptable, the system updates stock records and moves items into storage. After storage, a notification is sent to the administrator. At closing time, the system generates a daily return log, summarizing all activities before ending the process.

Use-case diagram:
Class diagram:

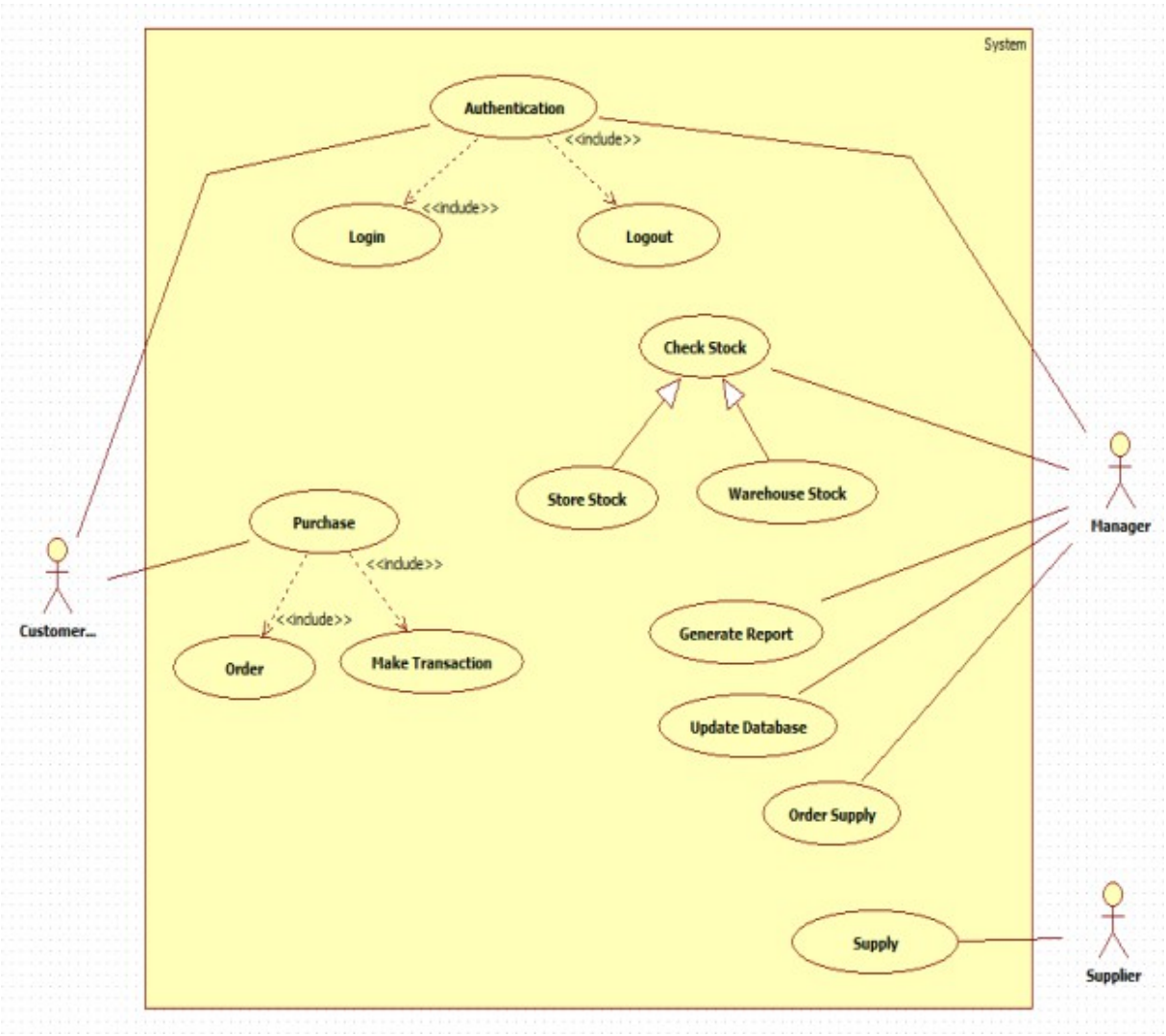


Fig 4.3 Use-case diagram of Stock management system

The use case diagram illustrates an inventory and order management system involving three actors: Customer, Manager, and Supplier. Customers can authenticate, log in, make purchases, place orders, and complete transactions. Managers oversee stock by checking warehouse and store quantities, generating reports, updating the database, and ordering supplies. Suppliers provide stock based on the manager's supply orders. Authentication supports all major activities, ensuring secure access to system functions.

Sequence diagram:

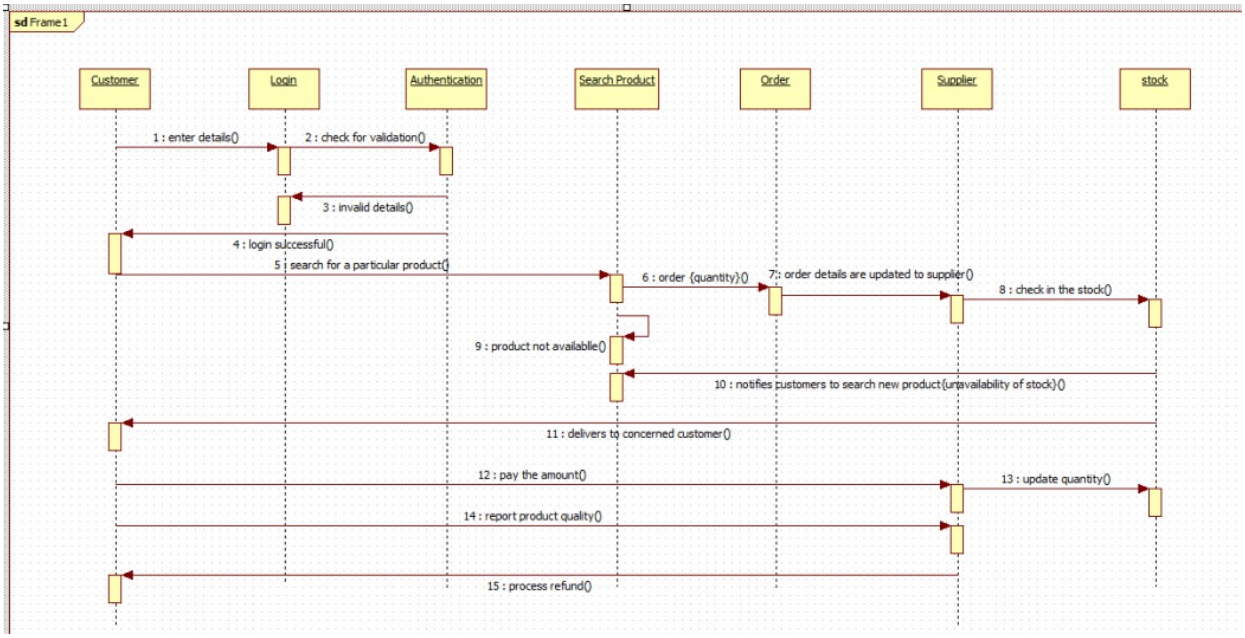


Fig 4.4 Sequence diagram of Stock management system

The sequence diagram illustrates the workflow of an online product ordering system. A customer logs in, searches for a product, and places an order. The system forwards order details to the supplier, who checks stock availability. If the product is unavailable, the system notifies the customer. If available, the supplier delivers the product, the customer makes the payment, and stock levels are updated. The customer may also report product quality, after which the system processes a refund if required.

Activity diagram:

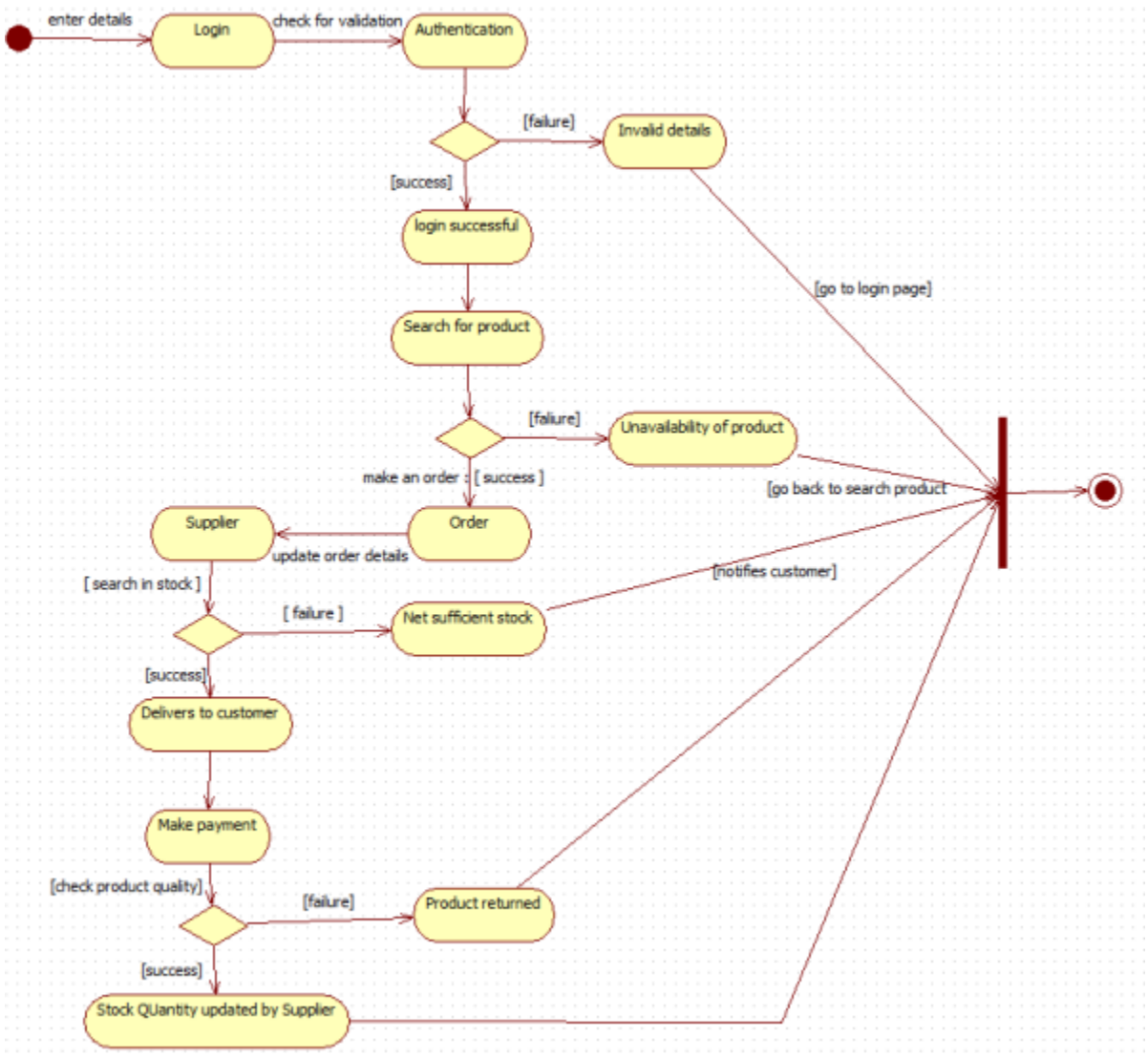


Fig 4.5 Activity diagram of Stock management system

The activity diagram shows the workflow of an online product purchase process. The customer logs in and is authenticated; if valid, they search for a product and place an order. The supplier checks stock availability and updates order details. If stock is sufficient, the product is delivered, payment is made, and product quality is checked. If the product fails the quality check, it is returned; if successful, stock quantity is updated. Invalid login or unavailable products redirect the user back to the appropriate steps.

5. Passport Automation System

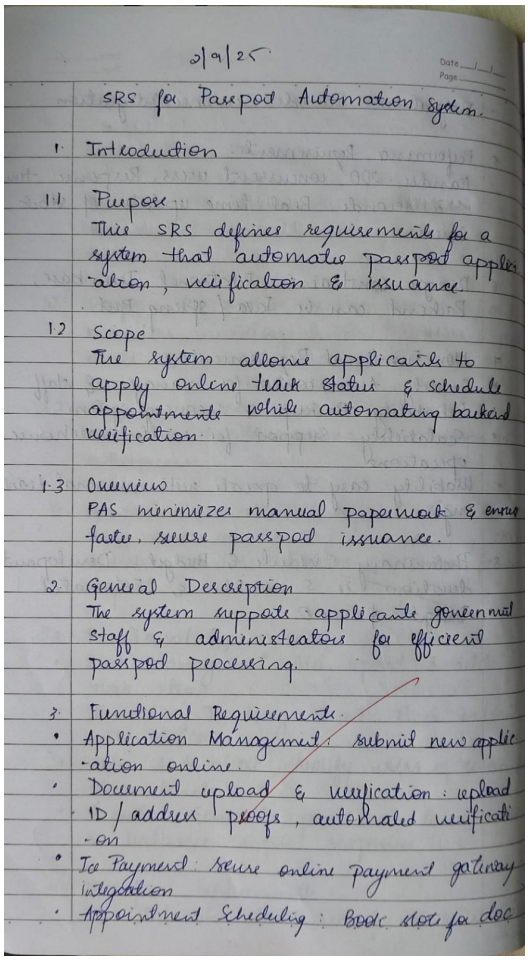
Activity diagram:

Problem Statement

Traditional passport application and verification processes involve multiple manual steps, long queues, and paperwork, leading to delays, errors, and inefficiencies. Tracking application status, verifying documents, scheduling appointments, and issuing passports become increasingly difficult as the number of applicants grows.

A Passport Automation System is needed to streamline application submission, document verification, appointment scheduling, status tracking, and passport issuance. The goal is to reduce processing time, minimize human errors, improve transparency, and provide a faster, user-friendly experience for both applicants and officials through a centralized digital platform.

Software Requirements Specification (SRS)



2/9/25

SRS for Passpod Automation System.

1. Introduction

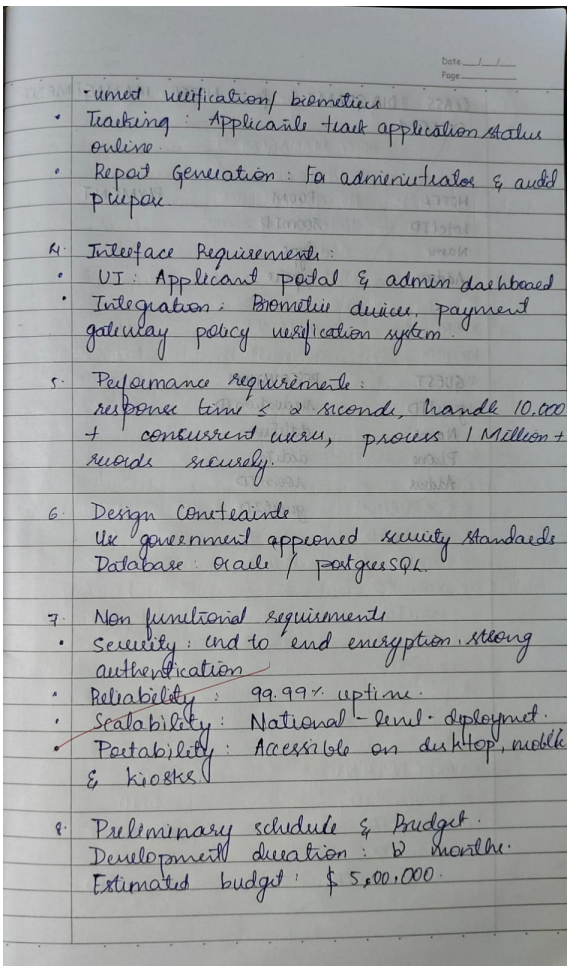
1.1 Purpose
This SRS defines requirements for a system that automates passpod applic-ation, verification & issuance.

1.2 Scope
The system allows applicants to apply online, track status & schedule appointments while automating backend verification.

1.3 Overview
PAS minimizes manual paperwork & errors, faster, secure passpod issuance.

2. General Description
The system supports applicants, government staff & administrators for efficient passpod processing.

3. Functional Requirements
- Application Management: submit new applic-ation online.
- Document upload & verification: upload ID / address proofs, automated verificati-on
- Fee Payment: secure online payment gateway integration
- Appointment Scheduling: Book slots for doc



- ument verification/ biometrics
- Tracking: Applicants track application status online.
- Report Generation: For administrators & audit purpose

4. Interface Requirements:
- UI: Applicant portal & admin dashboard
- Integration: Biometric devices, payment gateway, policy verification system.

5. Performance requirements:
response time ≤ 2 seconds, handle 10,000 + concurrent users, process 1 Million + records securely.

6. Design Constraints
Use government approved security standards
Database: Oracle / postgreSQL.

7. Non functional requirements
- Security: end to end encryption, strong authentication
- Reliability: 99.99% uptime.
- Scalability: National - level deployment.
- Portability: Accessible on desktop, mobile & kiosks

8. Preliminary schedule & Budget.
Development duration: 12 months.
Estimated budget: $ 5,00,000.

Class diagram:

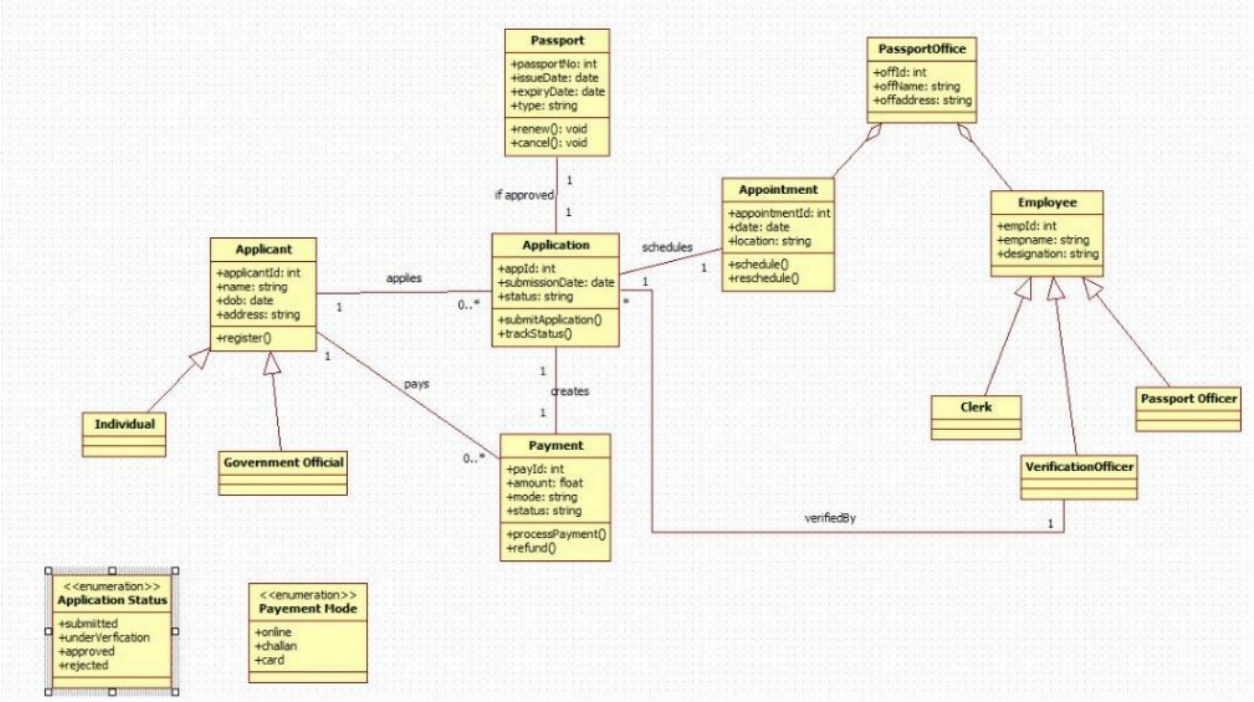


Fig 5.1 Class diagram of Passport Automation System

The class diagram represents a passport application management system. An Applicant—either an individual or government official—submits an Application and makes the required Payment. The Passport Office manages appointments and employs different staff roles, including clerks, passport officers, and verification officers, who verify applications. Upon approval, a Passport is issued or renewed. The diagram also includes application statuses and payment modes, showing how the system processes applications from submission to approval.

State diagram:

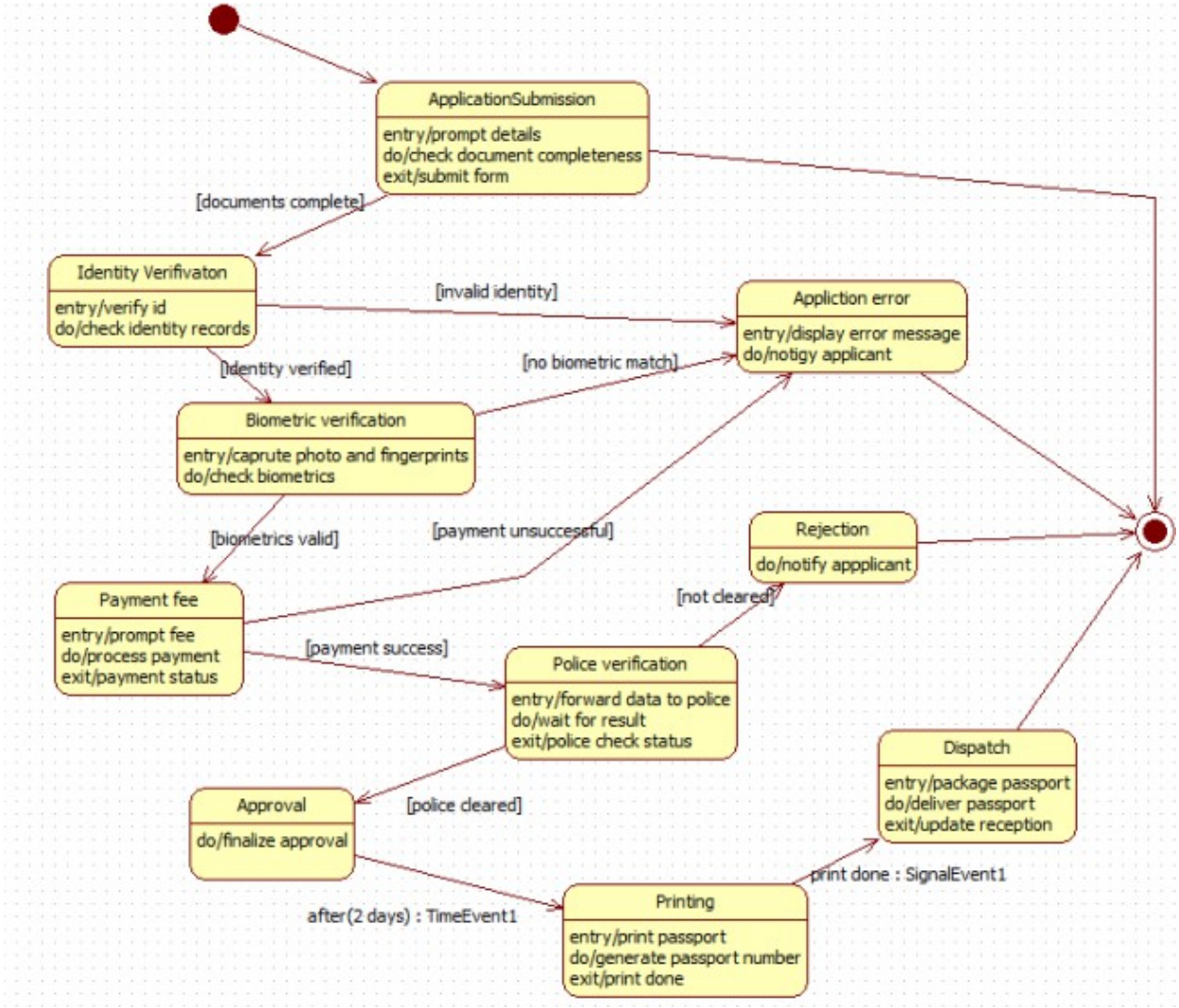


Fig 5.2 State diagram of Passport Automation System

The state diagram illustrates the complete workflow of a passport application process. It begins with application submission, followed by identity and biometric verification. If verification fails, the system moves to an error or rejection state. Upon successful verification, the applicant pays the required fee, and the application proceeds to police verification. Once cleared, the passport is approved, printed, and dispatched to the applicant. Any failure at intermediate steps leads to rejection.

Use-case diagram:

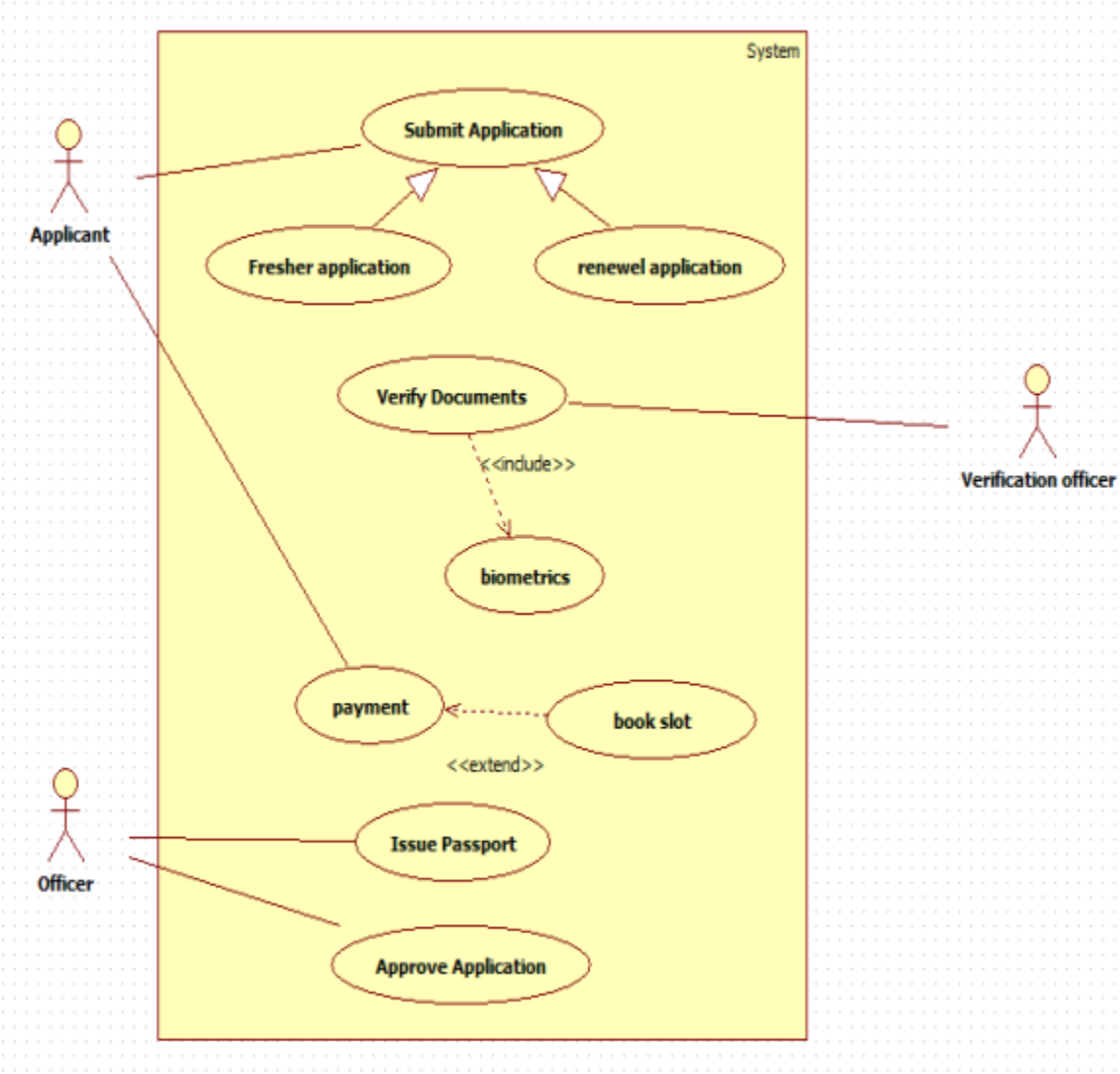


Fig 5.3 Use-case diagram of Passport Automation System

The use case diagram represents a passport application system involving Applicants, Officers, and Verification Officers. Applicants submit either a fresh or renewal application, after which the system verifies documents and captures biometrics. Applicants then make payments and may book a slot if required. Officers handle the approval and issuance of the passport, completing the application process.

Sequence diagram:

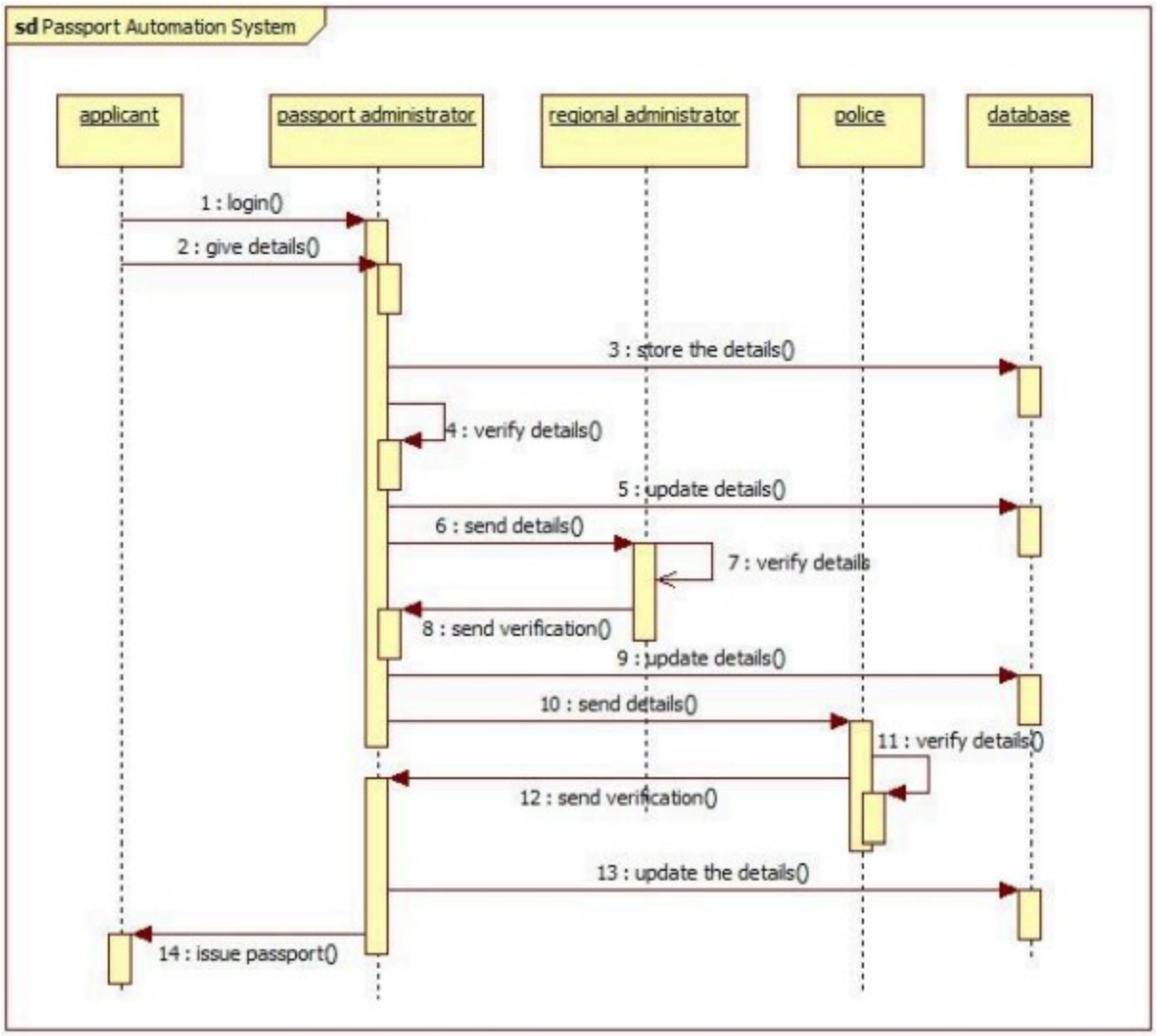


Fig 5.4 Sequence diagram of Passport Automation System

The sequence diagram shows the workflow of a passport automation system. The applicant logs in and submits details, which the passport administrator stores in the database. The regional administrator and police department sequentially verify the submitted information and update the records. After receiving verification from both authorities, the passport administrator issues the passport to the applicant.

Activity diagram:

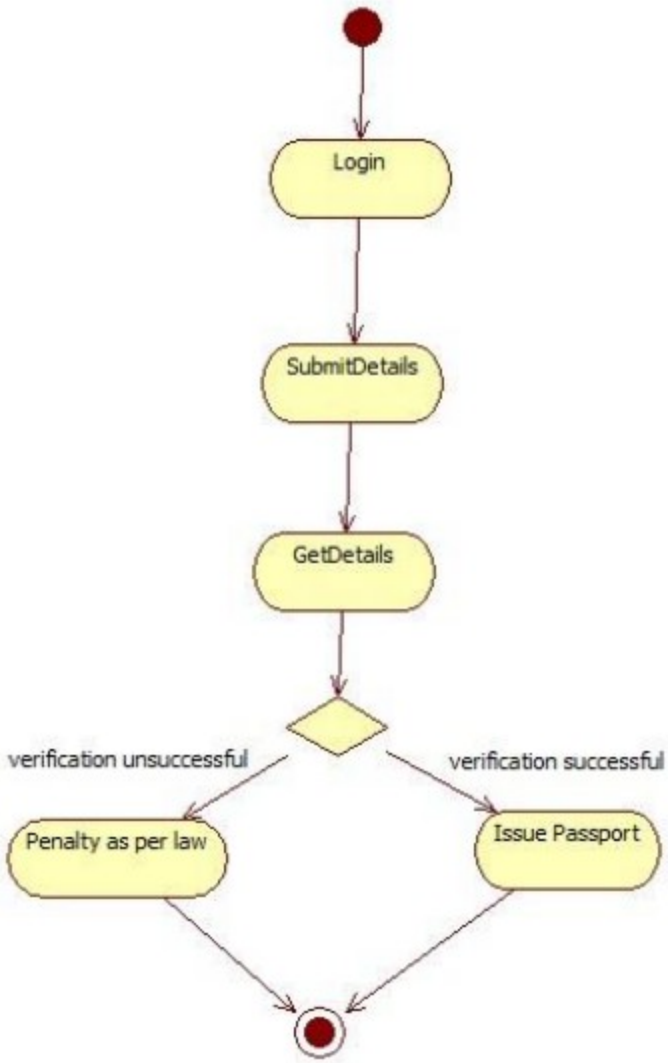


Fig 5.5 Activity diagram of Passport Automation System

The activity diagram shows a simple passport verification process. A user logs in, submits their details, and the system retrieves and checks the information. If verification is successful, a passport is issued; if verification fails, a legal penalty is applied. The process then ends.