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

Lab Record

Software Engineering and Object-Oriented Modeling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

AISHWARYA R

1BM23CS018

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 Analysis and Design(22CS6PCSEO) laboratory** has been carried out by **Aishwarya R(1BM23CS018)** during the 5th Semester August 2025-December 2025

Signature of the Faculty Incharge:

Name of your Batch Incharge and designation: **Megavalli M**
Assistant Professor

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

Table of Contents

1.HOTEL MANAGEMENT SYSTEM

A) Problem Statement

The hospitality industry is constantly evolving with guest expectations rising and competition fiercer than ever. With the rising demand for efficient operations, streamlined booking and personalized guest experiences, selecting the best hotel management software becomes crucial to enhance efficiency and guest satisfaction.

B) IEEE Standard Requirements Document

I. Introduction

a. Purpose of the Document

- Supercharge efficiency, streamline operations from front desk to back office.
- Elevate guest experience, convenience and personalization.
- Reduce errors, save time: automate repetitive tasks.
- Enhance security, data protection and business control.

b. Scope

- Reservation Management
- Room Assignment and Availability
- Billing and Invoicing
- Housekeeping Management
- Inventory Management
- Reporting and Analytics

II. Overview

A HMS acts as the central nervous system of a hotel, streamlining operations by managing all aspects of a business in one place.

III. General Description

Having an efficient HMS allows you to spend more time taking care of the needs of your guests and less time on repetitive administrative tasks. It helps minimize workload, maximize productivity and easily manage your hotel property.

IV. Functional Requirements

- Booking / Reservation
- Room Management
- Food Ordering
- Housekeeping
- Reporting and Analytics

V. Interface Requirements

a. Software Interfaces

- Server side OS: Windows / Linux OS
- Database: MongoDB
- Development tools: Java, HTML, JS, etc.

b. Hardware Interfaces

- Server side: 1 Monitor, RAM 4GB, Disk 100GB
- Client side: 1 Monitor, 1 Mouse, RAM 512 MB to 2GB
- Communication Interface: HTTP/HTTPS

VI. Performance Requirements

- Quick data updation
- Query must execute in seconds
- High user load should be supported
- Responsive reporting engine
- Visually pleasing UI

VII. Design Constraints

- Reduce complexities for staff and guest
- Reliance on Internet for cloud-based software
- System should scale and avoid data risk

VIII. Non-Functional Requirements

- *Security:* Confidentiality of the software
- *Reliability:* System must work without major recovery issues
- *Usability:* Users must be trained to use the system efficiently
- *Compatibility:* Work with different OS and devices

IX. Preliminary Schedule and Budget

a. Schedule

- **Phase 1:** Elicitation and Requirement Gathering (4 weeks)
- **Phase 2:** System Design (1 month)
- **Phase 3:** Development (3 months)
- **Phase 4:** Testing (2 months)
- **Phase 5:** Deployment and Training (2 months)
- **Phase 6:** Maintenance and Support (Ongoing)

b. Budget Estimation

- Functional cost ≈ ₹2 lakh (project manager, developers, UI/UX, system admin)
- System requirement ≈ ₹1.2 lakh
- Implementation & maintenance ≈ ₹50,000

Total estimate ≈ ₹3.7 lakh

Class Diagram:

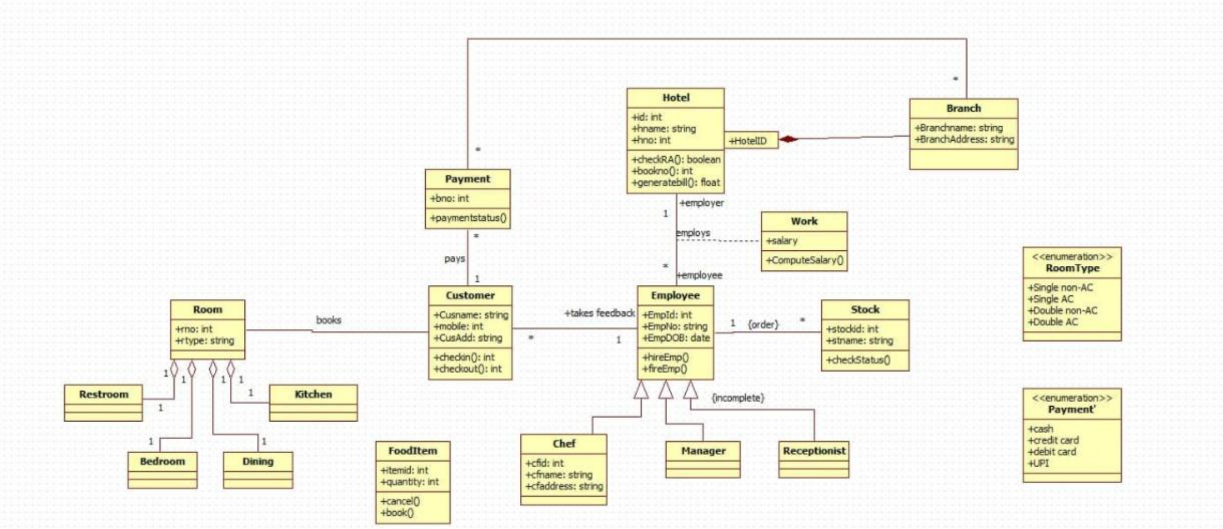

Fig 1.1 Class diagram of hotel management system

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:

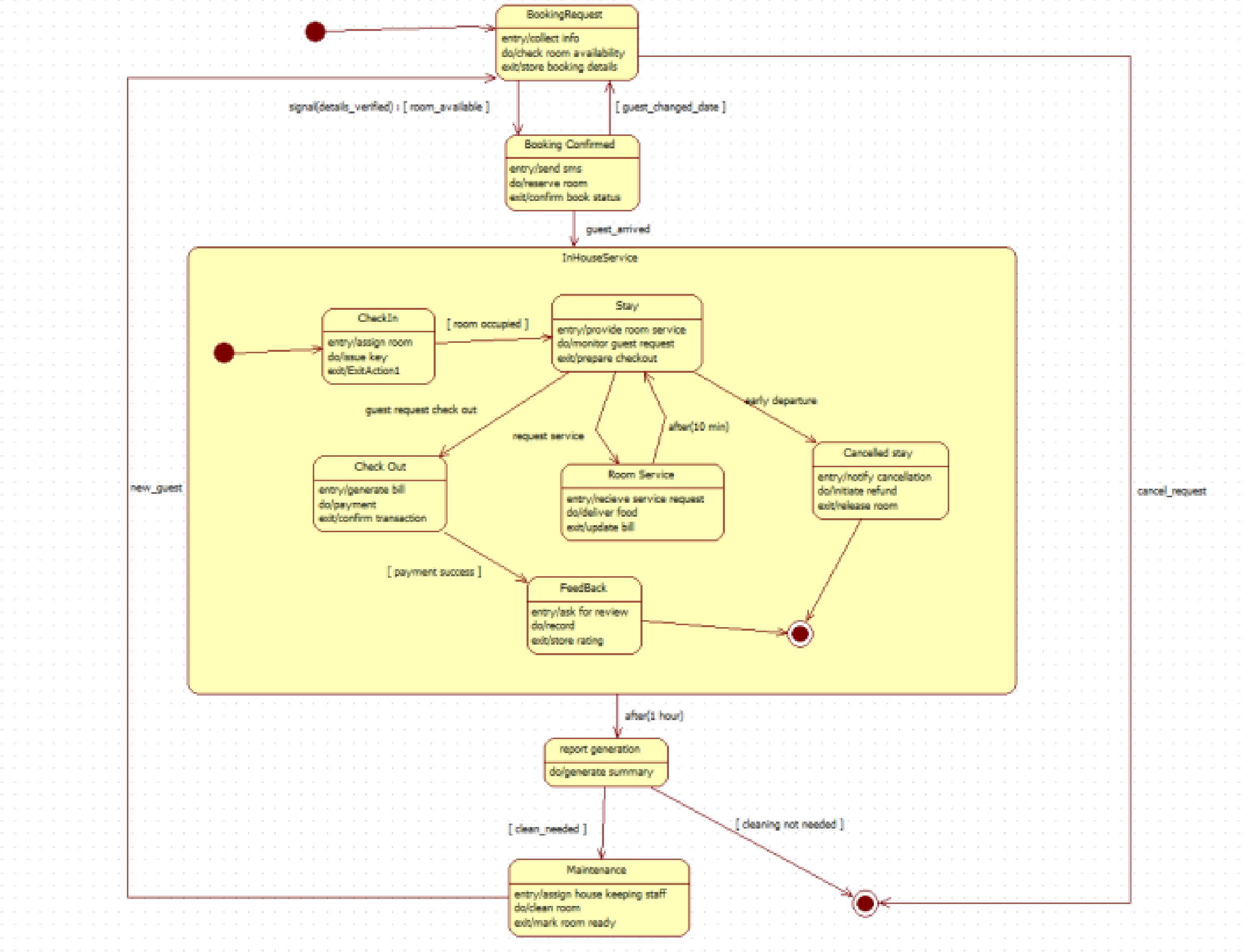

Fig 1.2 State diagram of hotel management system

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:

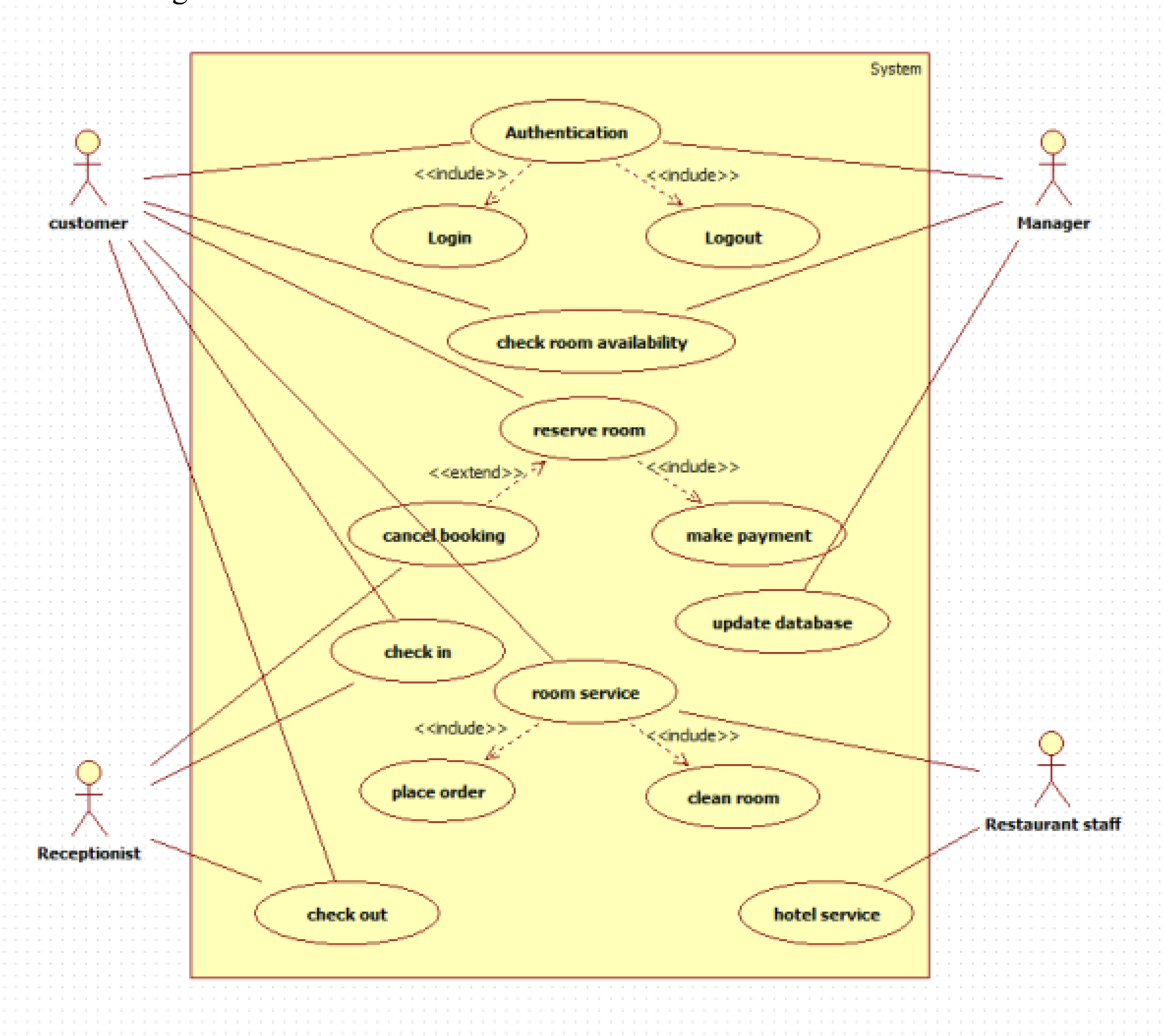

Fig 1.3 Use case diagram of hotel management system

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

Sequence Diagram:

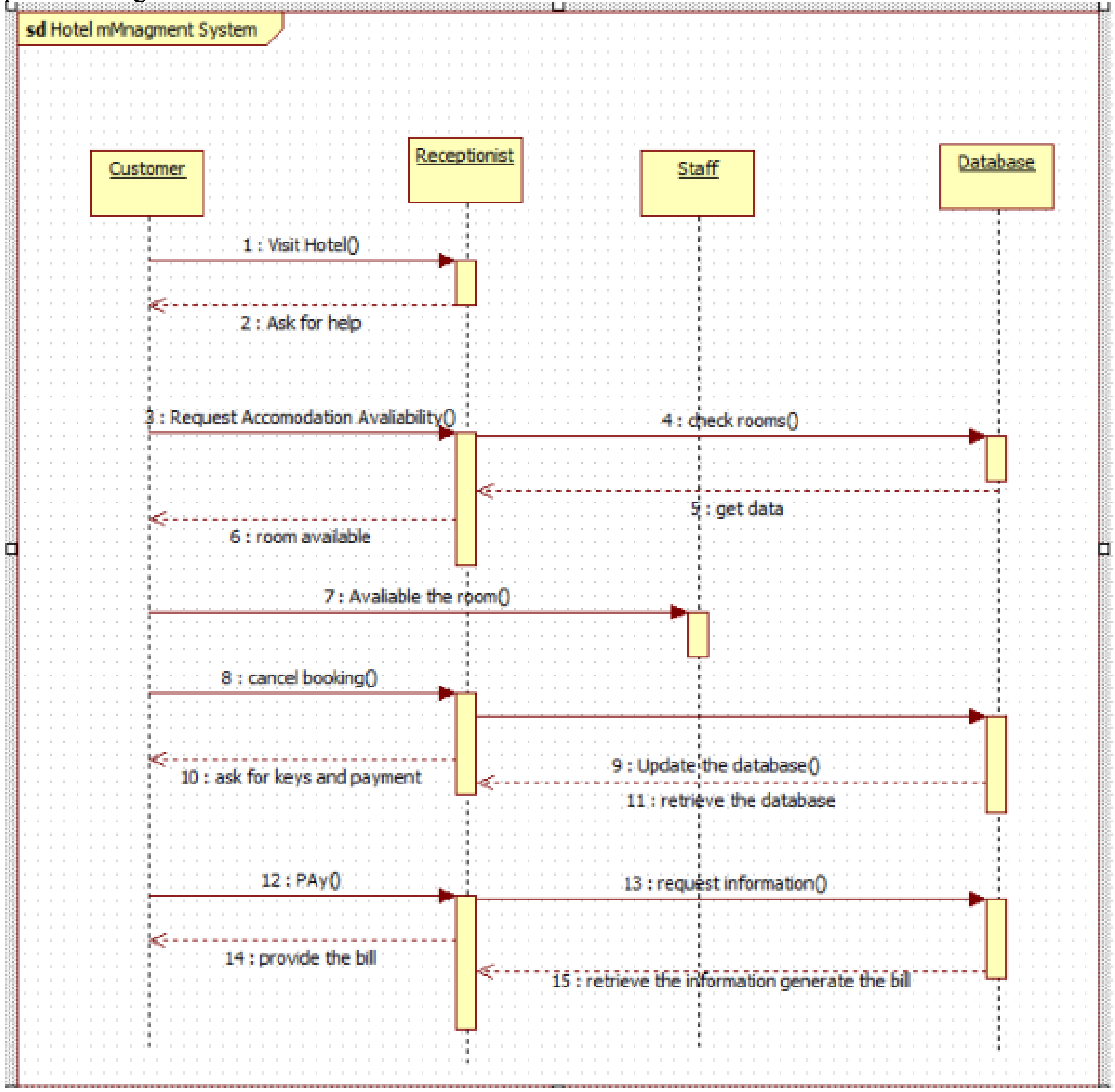
sd Hotel mMnagment System
Customer
Receptionist
Staff
Database
1 : Visit Hotel()
2 : Ask for help
3 : Request Accomodation Avaliability()
4 : check rooms()
5 : get data
6 : room available
7 : Avaliable the room()
8 : cancel booking()
9 : Update the database()
10 : ask for keys and payment
11 : retrieve the database
12 : PAy()
13 : request information()
14 : provide the bill
15 : retrieve the information generate the bill

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:

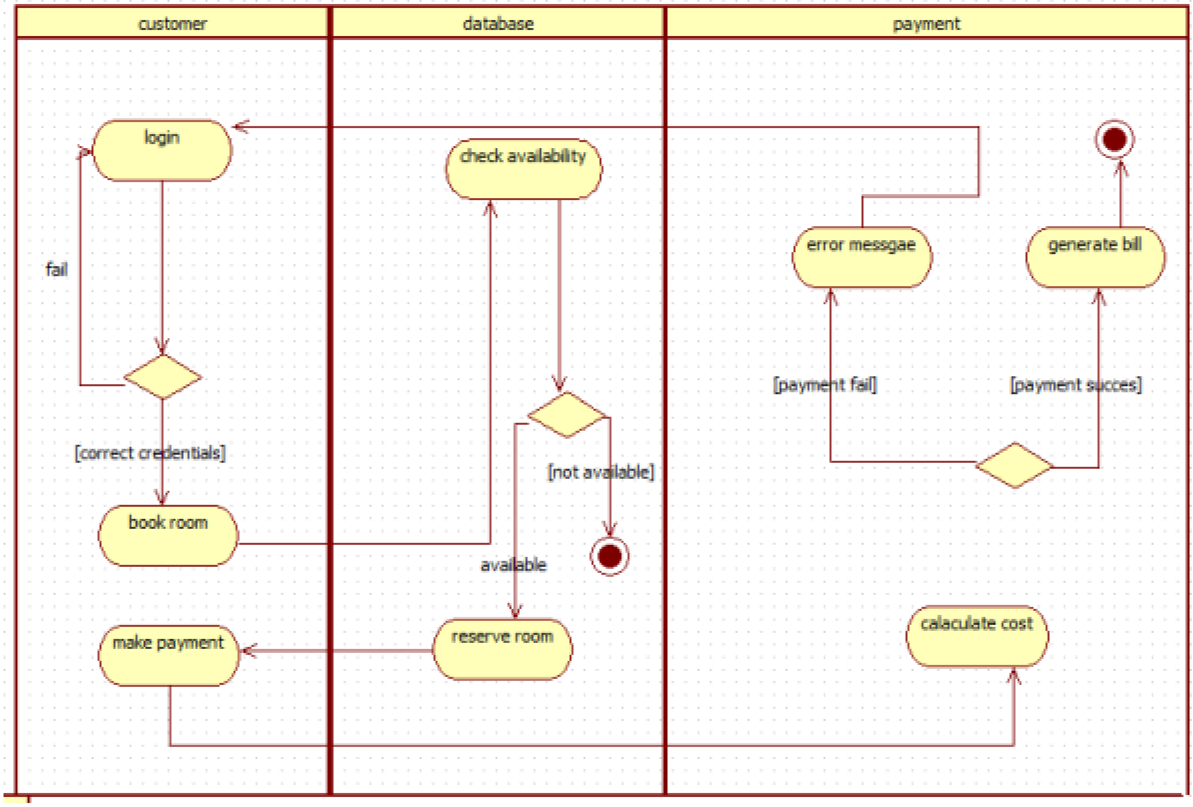

Fig 1.5 Activity diagram of hotel management system

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 PROCESSING SYSTEM

A) Problem Statement

Credit card processing is vital for businesses to accept electronic payments securely and efficiently.

In this digital world, CCP is essential for verification, ensuring funds, easy money transfer.

B) SRS Document

I. Introduction

a. Purpose

Credit card processing is a critical component in receiving transactions, whether online or in-store or over the phone. It is slowly becoming one of the most convenient and acceptable means of online payment.

b. Scope

- Stakeholders – card owners, merchants, banks, card networks
- Transaction process
- Tech infrastructure
- Security standards
- Fee and revenue models

II. General Description

The process involves initiation, card transmission, authorisation, approval, fund transfer.

III. Functional Requirements

- **User data** – validate card details
- **Payment processing** – generate invoice & receipt
- **Transaction management** – invoice generation
- **Security and compliance** – fraud detection, OTP
- **Funding and settlement** – fund transfer & dispute resolution

---

IV. Interface Requirements

- UI
- Payment screen
- Transaction history
- Payment API

V. Design Constraints

- Fraudulent activity
- Multiple payments
- High processing fee

VI. Preliminary Schedule and Budget

a. Schedule

Phase	**Duration**
Requirements gathering	1 month
System design	1.5 months
Core development	3 months
Frontend dev	2 months
API integration	2 months
Testing	2 months
Deployment	1 month

Class Diagram:

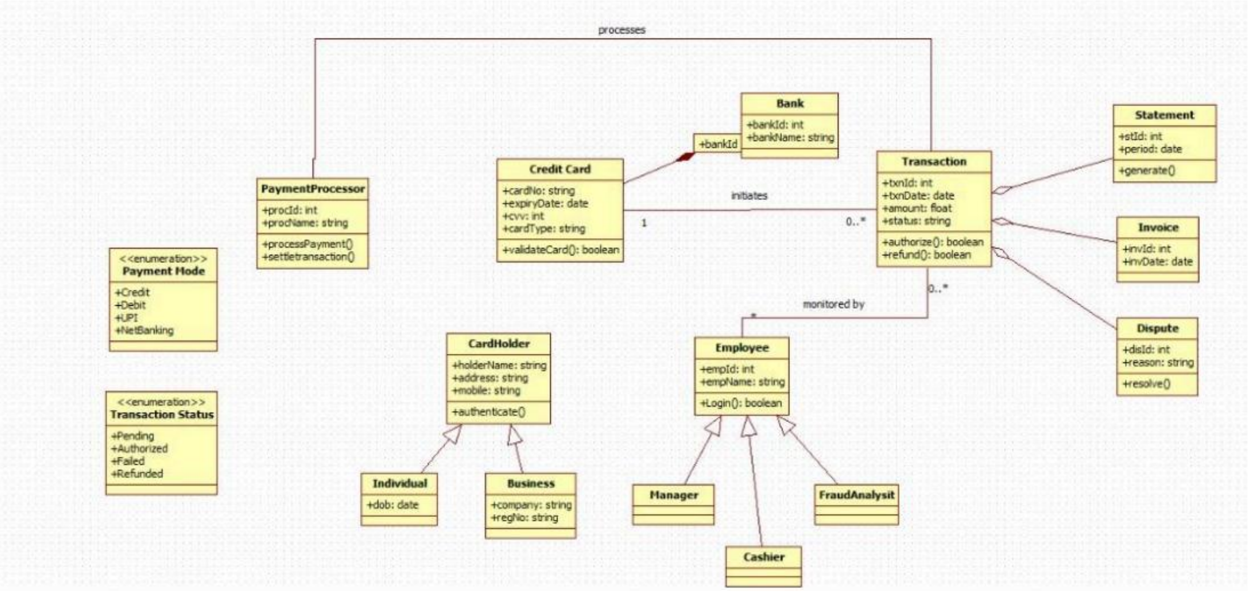

Fig 2.1 Class diagram of Credit card processing system

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:

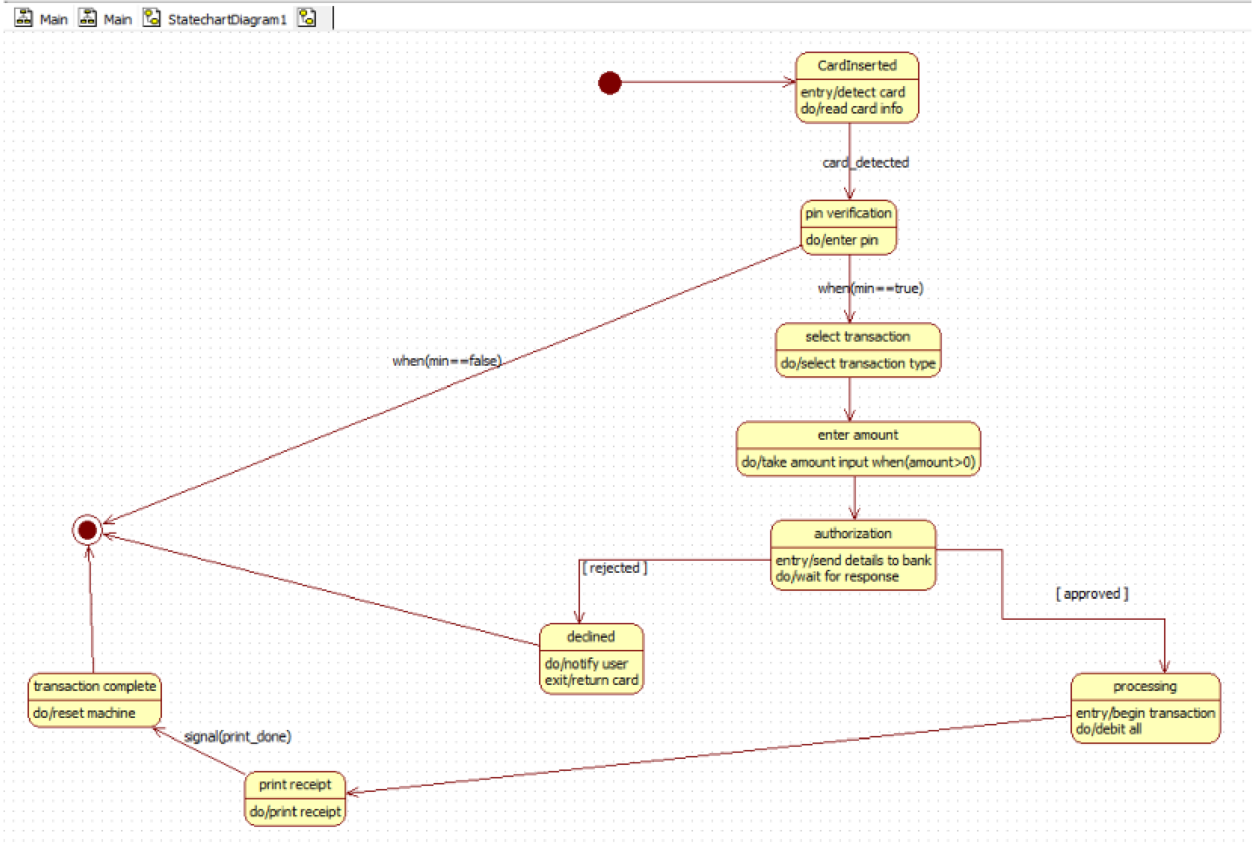

Fig 2.2 State diagram of Credit card processing system

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:

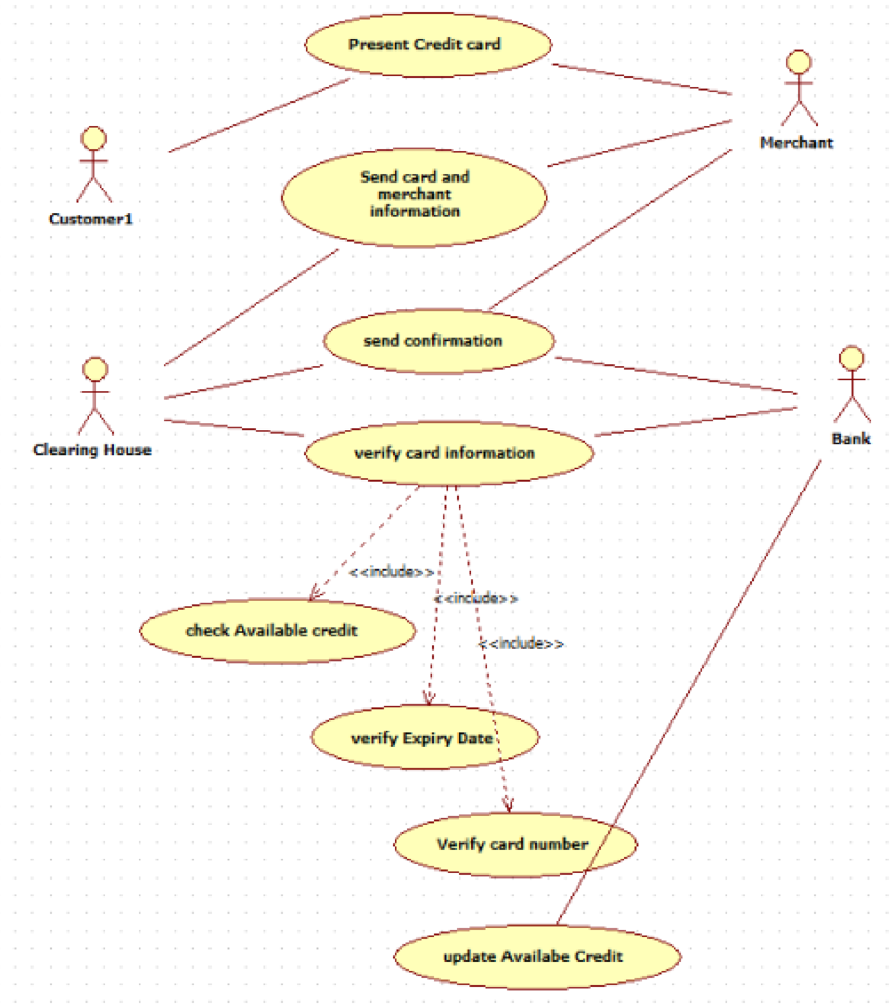

Fig 2.3 Use-case diagram of Credit card processing system

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:

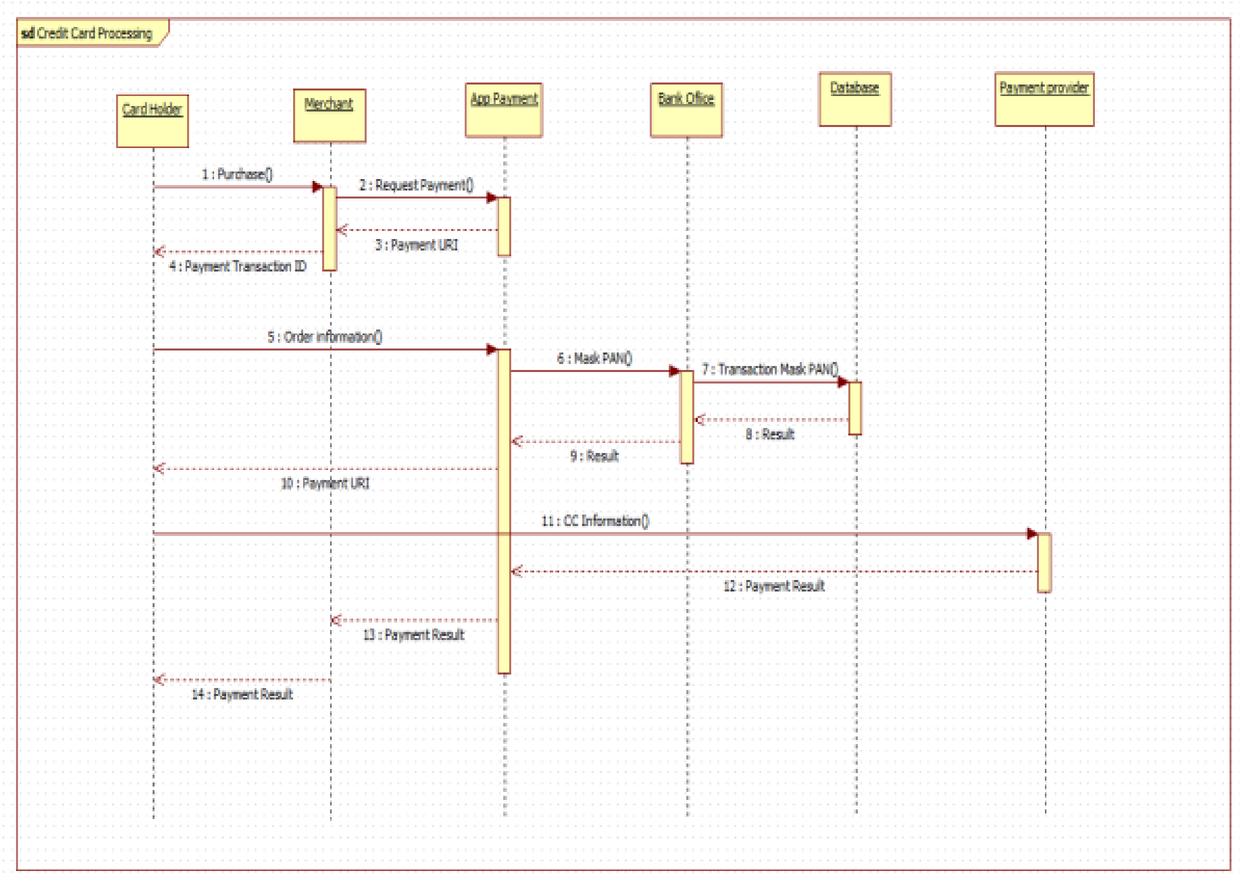

Fig 2.4 Sequence diagram of Credit card processing system

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:

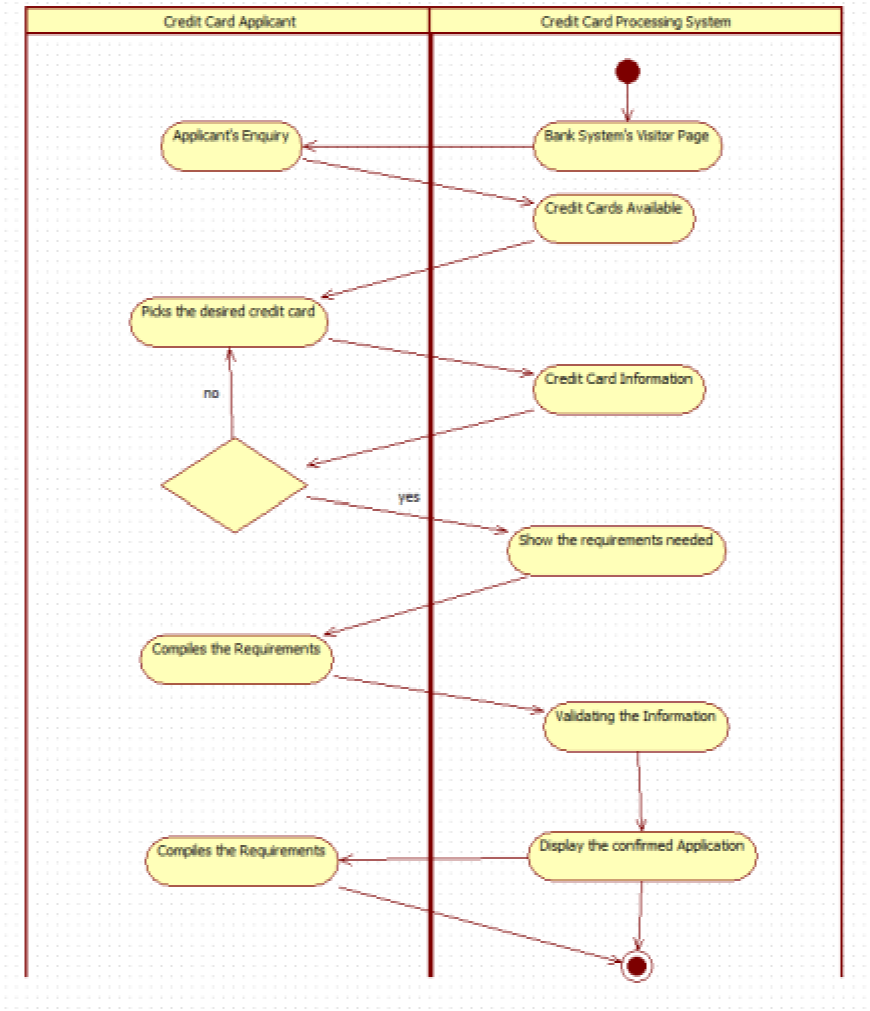
Credit Card Applicant
Credit Card Processing System
Applicant's Enquiry
Bank System's Visitor Page
Credit Cards Available
Picks the desired credit card
Credit Card Information
no
yes
Show the requirements needed
Compiles the Requirements
Validating the Information
Compiles the Requirements
Display the confirmed Application

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

Manual library management is slow, error-prone and inefficient.

Tasks like book search, issuing, manual record keeping, fine calculation become difficult with paper records.

An automated system is needed to manage books and users more efficiently.

PURPOSE

The purpose of this system is to computerize the management of library operations.

It will allow librarians to manage books, members and borrowing records efficiently.

SCOPE

It will provide a digital platform to store book details, member info and transaction history.

It reduces manual work and redundancy and ensures quick information access.

OVERVIEW

The system will maintain a centralized database of books and users.

Librarians can add, update or delete book entries.

Library members can log in and search books.

FUNCTIONAL REQUIREMENTS

- To add, update, delete book records
- Register and manage member accounts
- Searching by book title, category, or author
- Track fine & borrowing history and the status of all items

INTERFACE REQUIREMENTS

- **User interface:** simple web-based interface with login page, search bar & dashboard
- **Admin interface:** for librarians for book and user management
- Use of **SQL database (RDBMS)** for storing & retrieving data

PERFORMANCE REQUIREMENTS

- System should support at least **200 concurrent users**
- Search results should have **quick load time**
- Database should handle **10,000+ records** efficiently

DESIGN CONSTRAINTS

- Built using open-source technologies (Java / Python)
- Works on common browsers (Chrome, Edge, Firefox)
- Secure login system with password protection

NON-FUNCTIONAL REQUIREMENTS

- **Reliability:** must ensure 99% uptime
- **Security:** password-based system with encryption, role-based access
- **Usability:** interface must be simple for users
- **Maintainability:** easy to update records & add new features

SCHEDULE

- **Week 1–2:** Requirements gathering and analysis
- **Week 3–4:** Database design and setup
- **Week 5–7:** Implementation of core modules
- **Week 8–9:** Integration, testing
- **Week 10:** Deployment

COST ESTIMATION

- **Software development:** ~ ₹50,000
- **Maintenance:** ~ ₹10,000
- **Hardware & hosting:** ~ ₹10,000

- **Total estimated cost:** ~ ₹70,000

Class diagram:

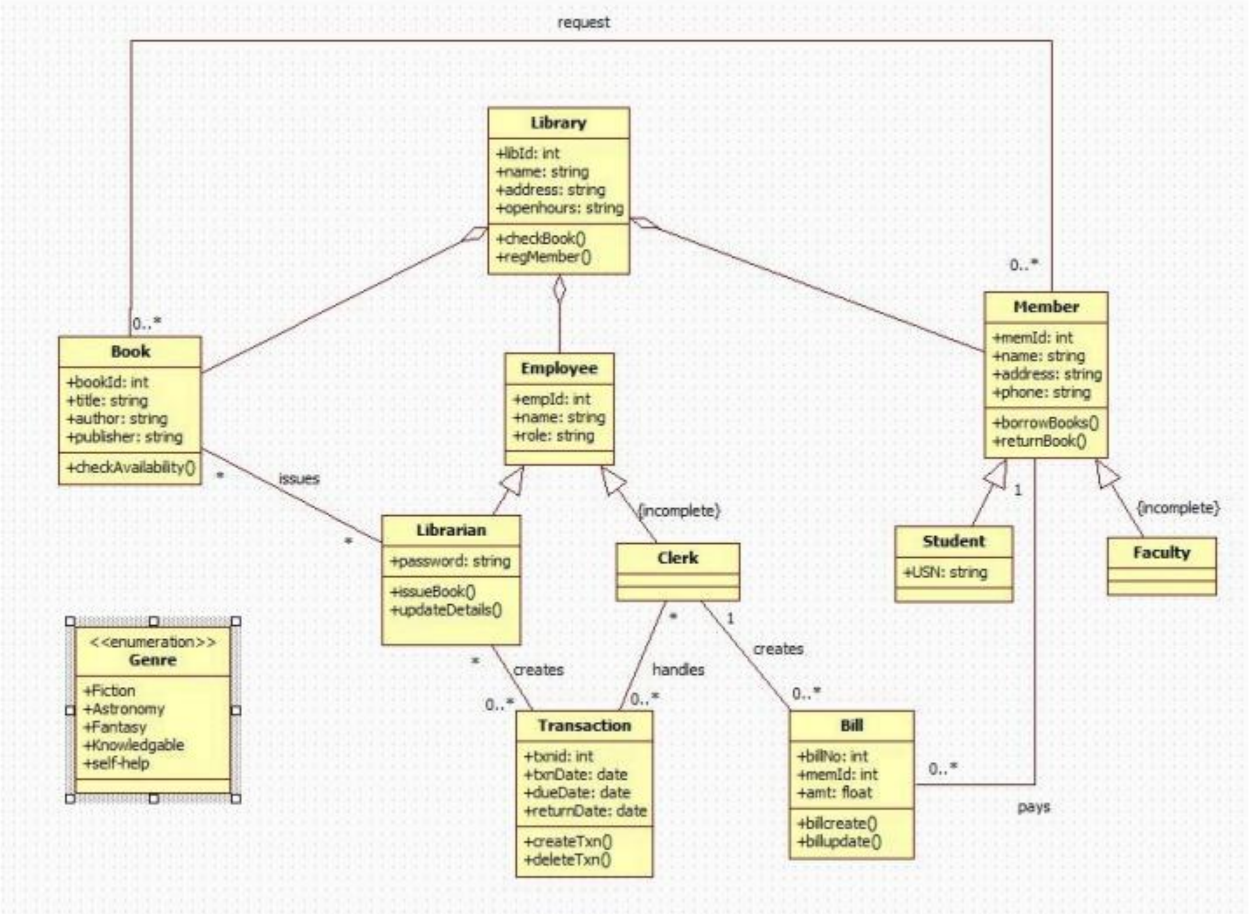

Fig 3.1 Class diagram of Library management system

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:

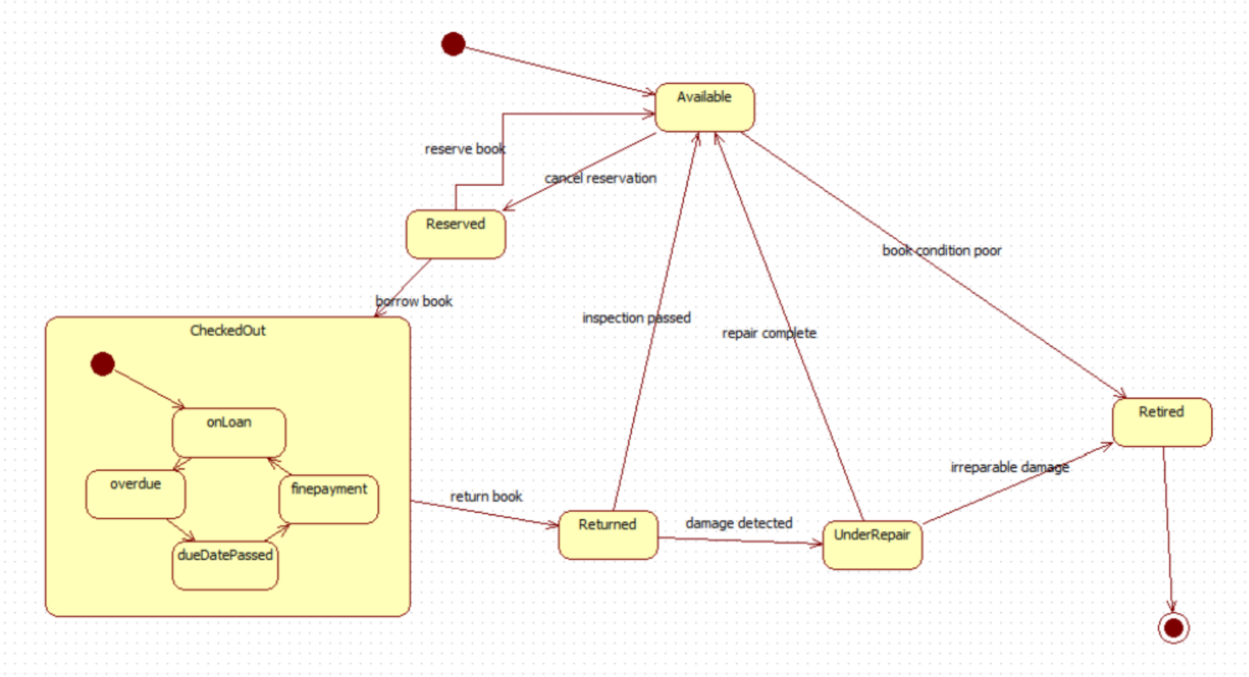

Fig 3.2 State diagram of Library management system

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:

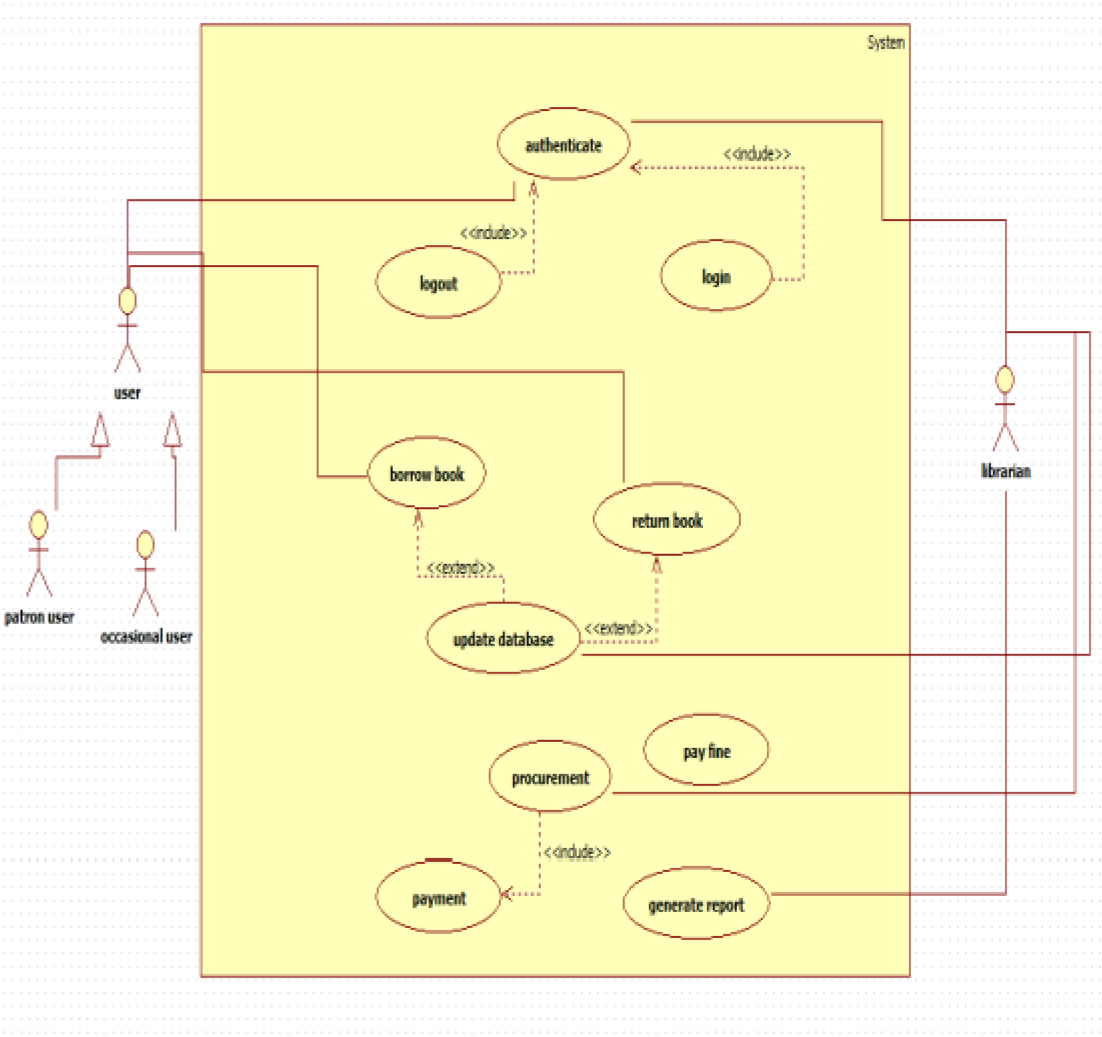

Fig 3.3 Use case diagram of Library management system

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:

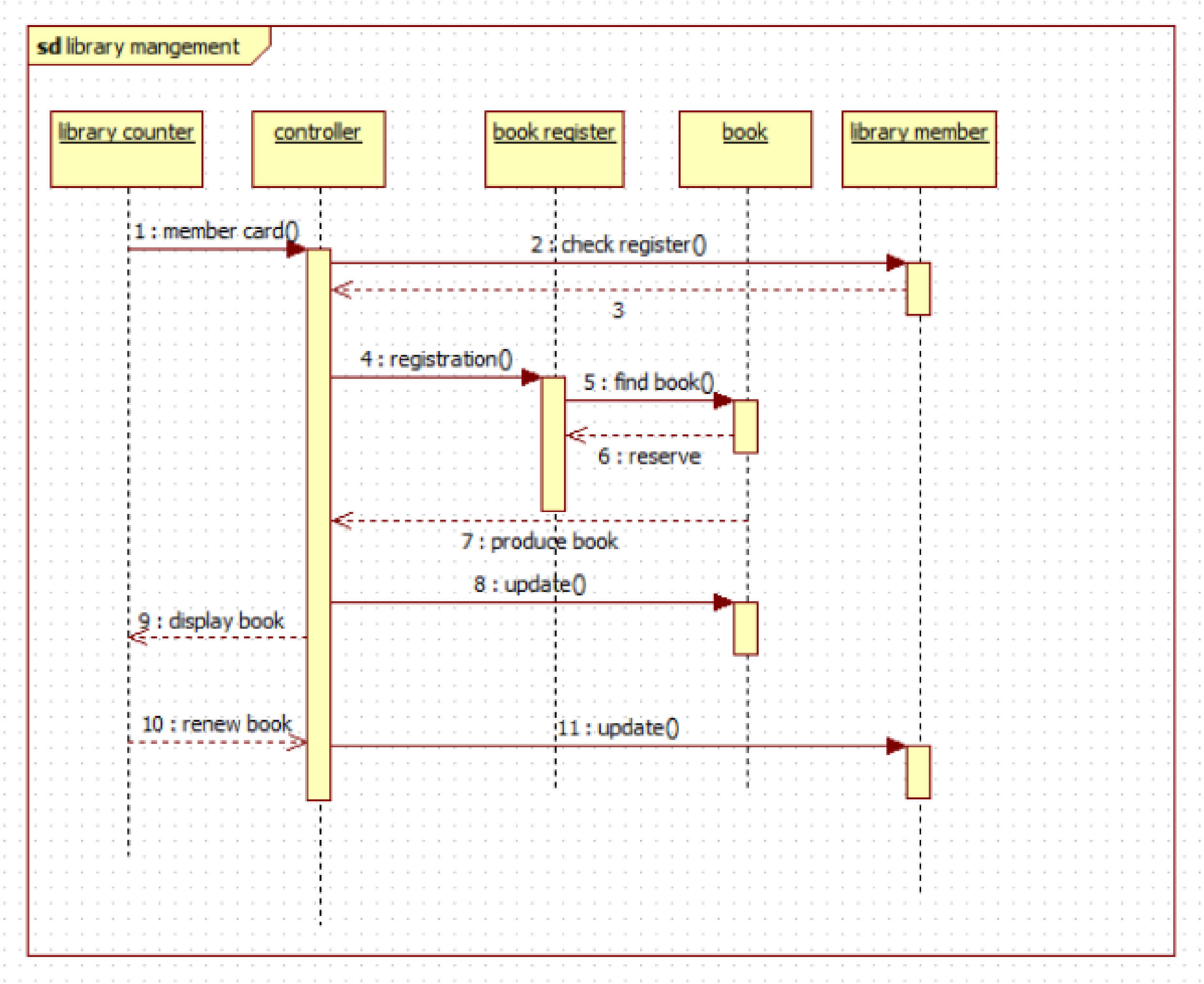

Fig 3.4 Sequence diagram of Library management system

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:

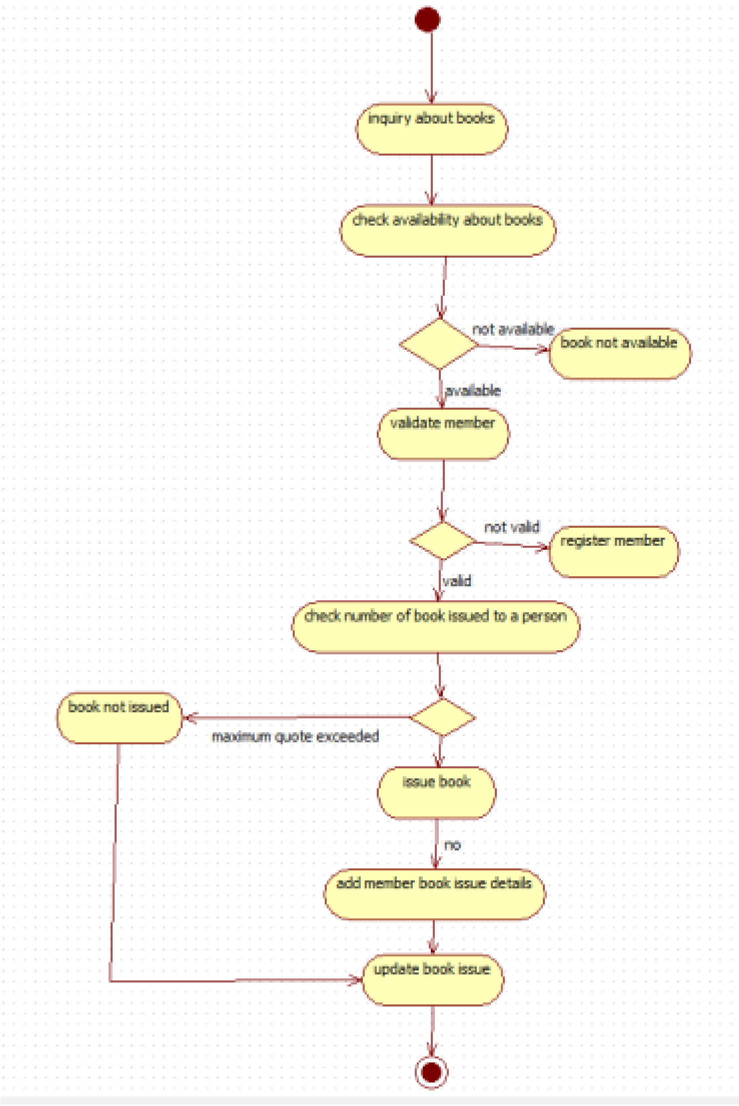

Fig 3.5 Activity diagram of Library management system

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 MANAGEMENT SYSTEM

Problem Statement

Manual stock management in warehouses leads to errors such as incorrect inventory counts, delayed stock updates and difficulty in tracking product availability.
This causes inefficiency and overstocking.
A digital solution is hence needed to streamline inventory tracking and reporting.

Purpose

The purpose of this system is to provide a reliable and efficient method for maintaining stock levels, monitoring item movement and generating reports.

Scope

- To maintain item details (ID, name, price, etc.)
- Track inflow and outflow of stock
- Generation of reports (daily/weekly/monthly)
- Secure access for admin and staff

Overview

The Stock Maintenance System will be a software application.
It records transactions and real-time update tasks.
It supports role-based access control.

Functional Requirements

- User login/logout with authentication
- Add, update, delete stock details
- Record transactions
- Generate inventory reports

INTERFACE REQUIREMENTS

- Windows/Linux
- Database (SQL, MongoDB)
- Data security and backups

NON-FUNCTIONAL REQUIREMENTS

- **Performance:** should handle at least 10k product details efficiently
- **Security:** role-based access, password auth, secure transactions
- **Usability:** easy to learn for non-technical staff
- **Reliability:** backup and recovery features

SCHEDULE

- **Week 1–2:** Requirements gathering & analysis
- **Week 3–4:** System design
- **Week 5–8:** Development of core modules

- **Week 9–10:** Testing & integration
- **Week 11:** Deployment and training

COST / BUDGET

Software development:	~ ₹50,000 – ₹80,000
Database & hosting:	~ ₹10,000 / yr
Hardware:	~ ₹5,000
Maintenance:	~ ₹10,000 / yr

Estimated total initial cost: ~ **₹70,000**

Class diagram:

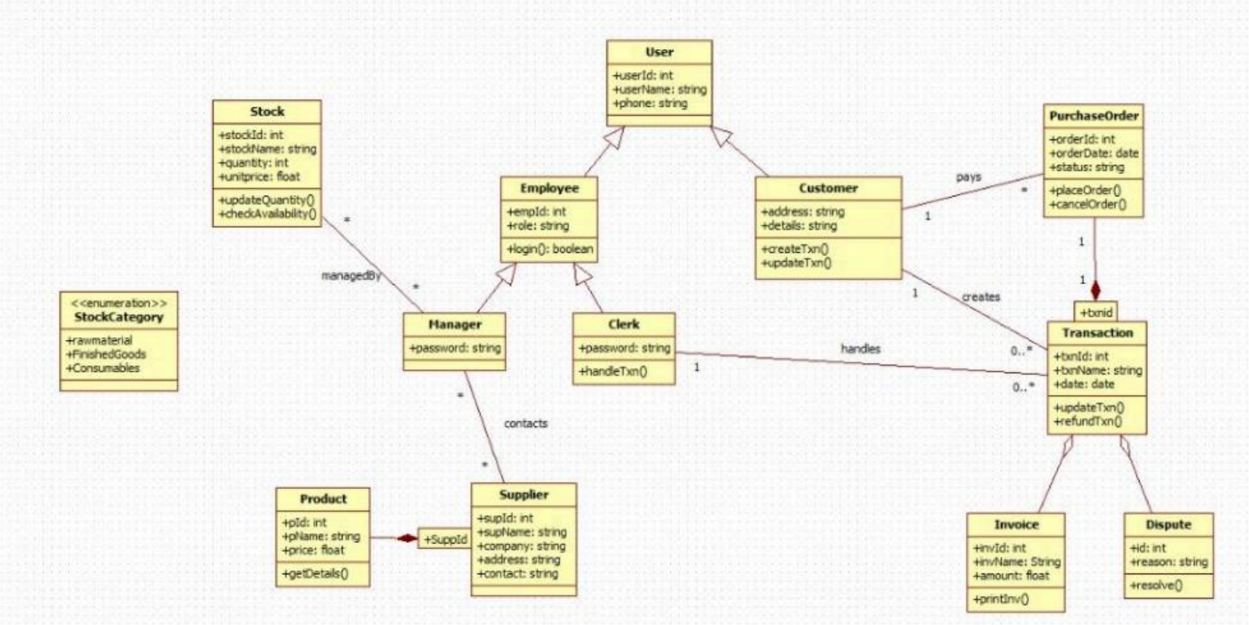

Fig 4.1 Class diagram of Stock management system

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:

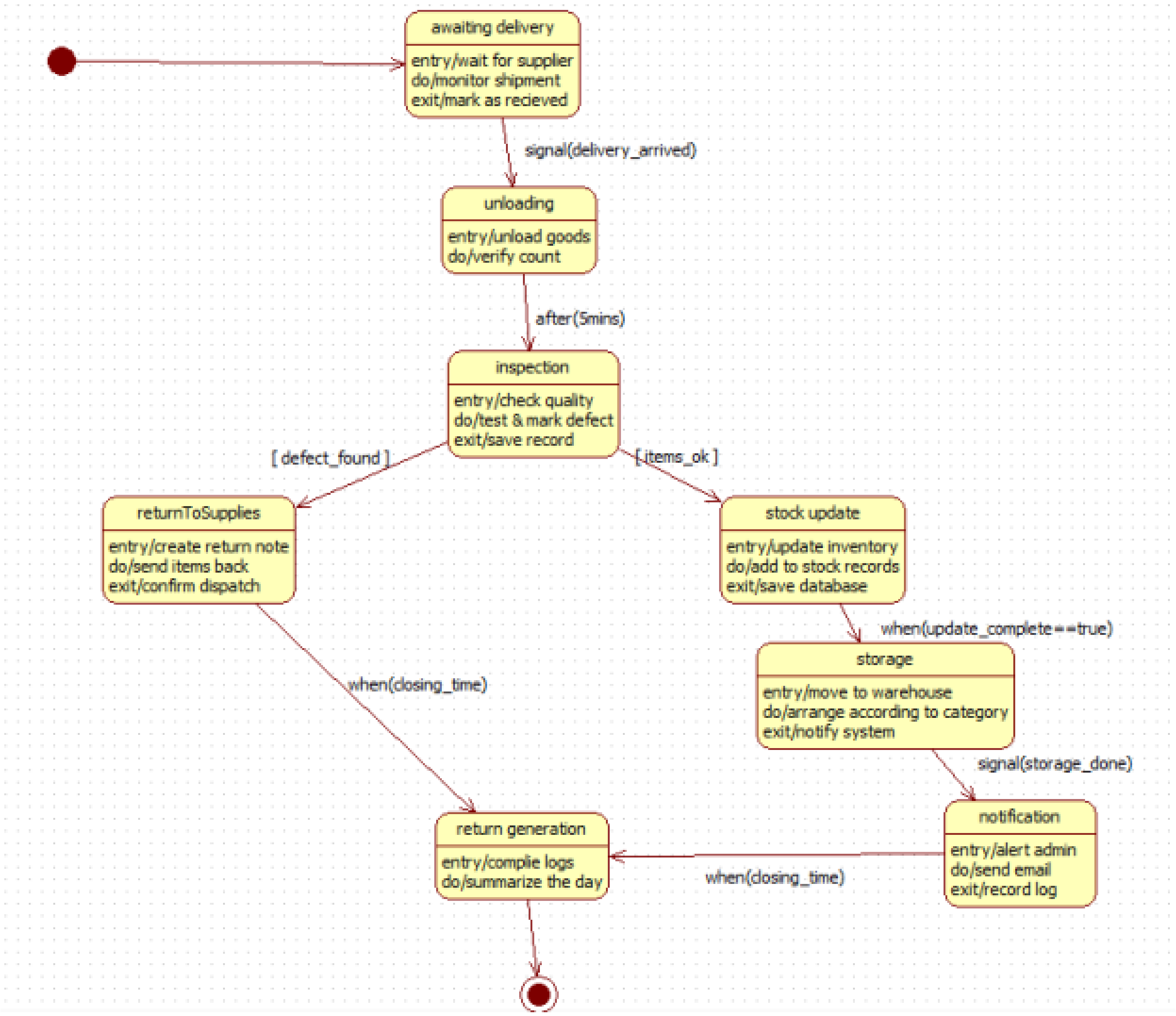

Fig 4.2 State diagram of Stock management system

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:

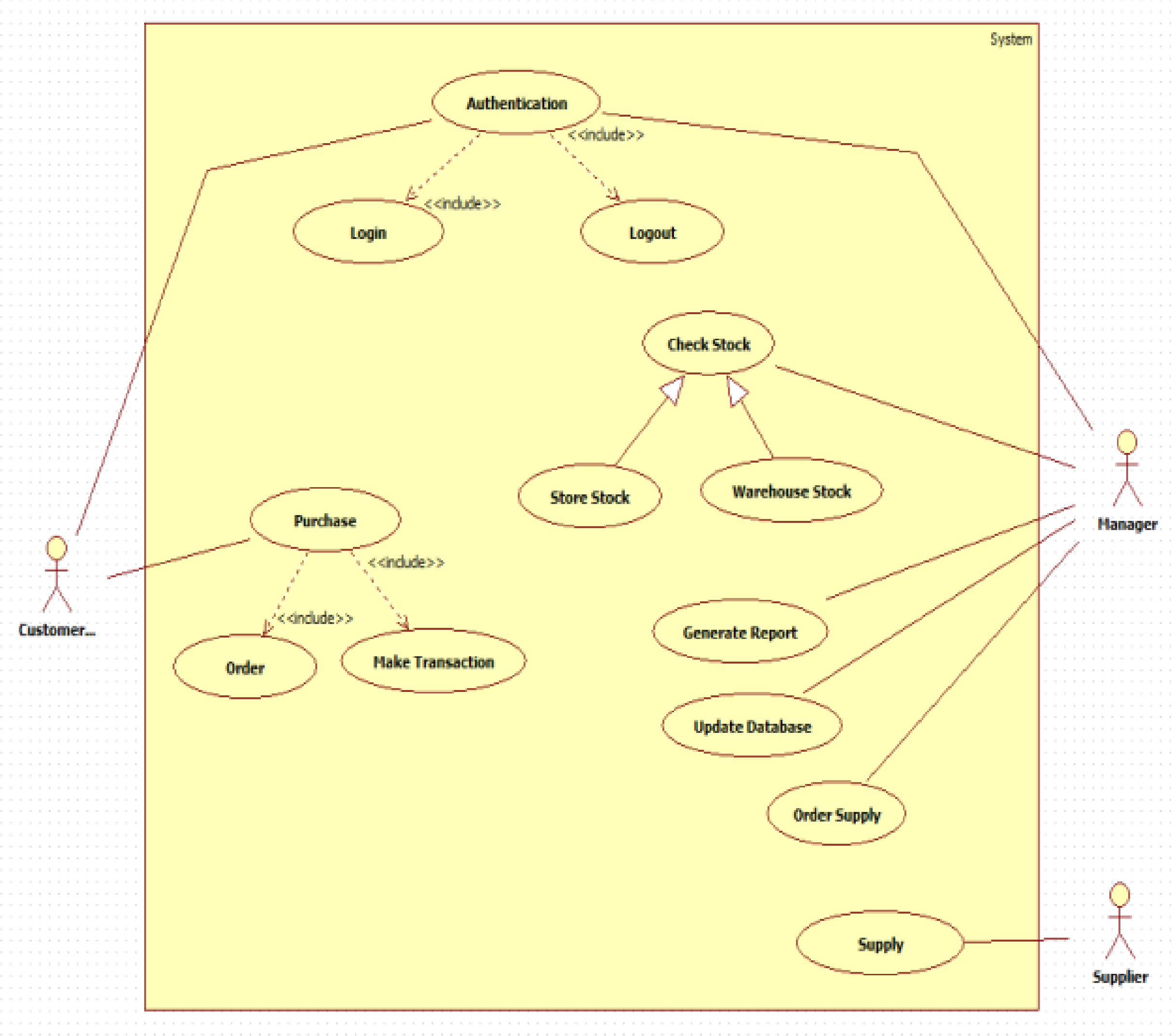

Fig 4.3 Use-case diagram of Stock management system

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:

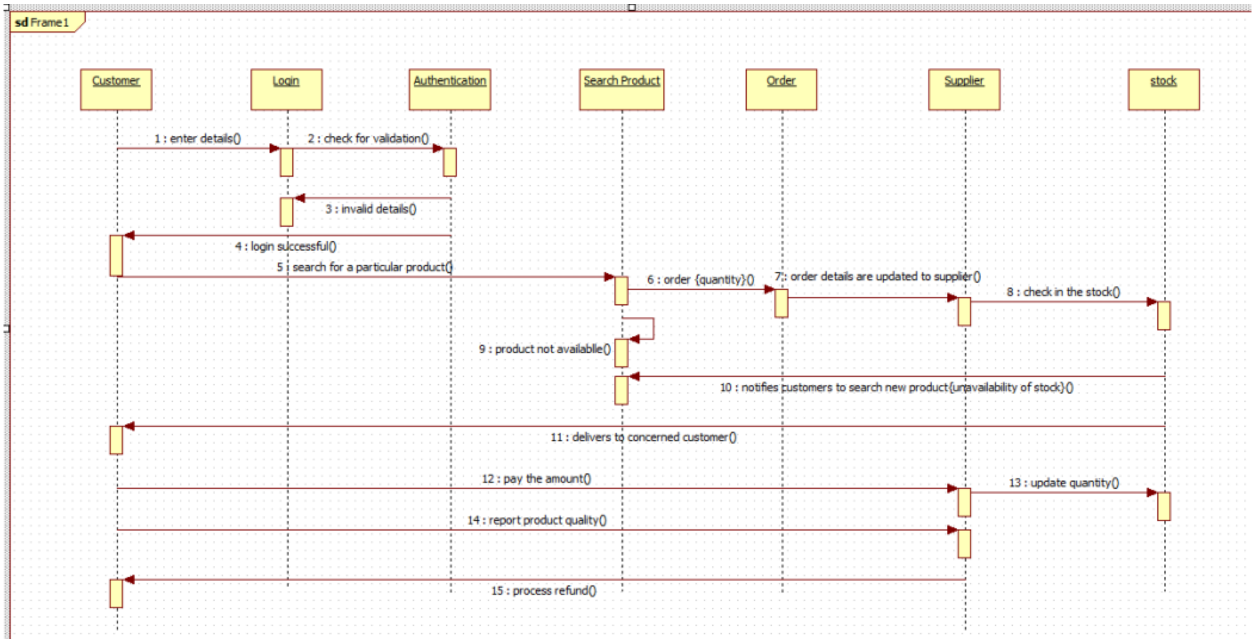
sd Frame1
Customer
Login
Authentication
Search Product
Order
Supplier
stock
1 : enter details()
2 : check for validation()
3 : invalid details()
4 : login successful()
5 : search for a particular product()
6 : order {quantity}()
7 : order details are updated to supplier()
8 : check in the stock()
9 : product not availablle()
10 : notifies customers to search new product{unavailability of stock}()
11 : delivers to concerned customer()
12 : pay the amount()
13 : update quantity()
14 : report product quality()
15 : process refund()

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:

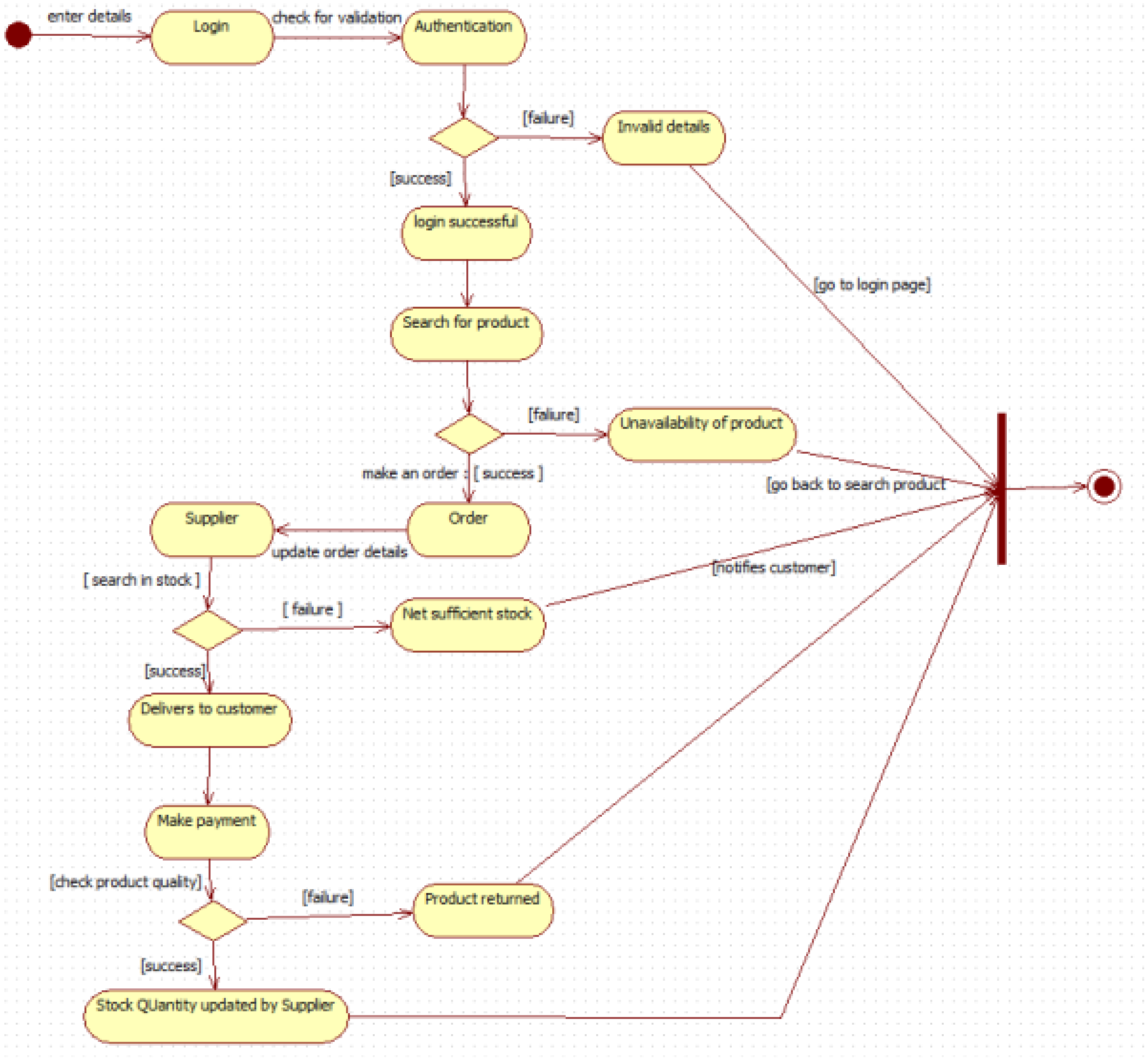

Fig 4.5 Activity diagram of Stock management system

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 AUTHENTICATION SYSTEM

Problem Statement

Manual passport application process is time-consuming, error-prone and requires significant paperwork.
With offices handling huge volumes of applications and lack of real-time tracking, errors and inefficiencies for both applicants and authorities occur.
An automated system is required to simplify passport application and verification.

Purpose

The passport system aims to streamline the process of application, verification, approval and delivery.
It allows users to apply online and track the status, and ensures secure handling of personal data.

Scope

- Online registration and application submission
- Upload necessary documents
- Automated verification workflow (police, govt)
- Tracking of application process
- Notifications via email/SMS

System Overview

The system will be a web-based platform integrated with government and police authorities.
Citizens can apply online, submit docs, schedule appointments, track status.
Officials can verify details and approve/reject issues.

System maintains a centralized database for records and reporting.

Functional Requirements

- User login / registration with authentication
- Online passport application submission
- Document upload and validation
- Appointment scheduling for biometrics / document verification
- Application status tracking

Interface Requirements

- **UI:** Web portal with user dashboard
- **Input:** Uploaded documents, application form
- **Output:** Receipt, status
- **Admin panel:** Government dashboards, database

Design Constraints

- Must comply with govt. security & privacy policies
- Should handle large volumes of applications concurrently

- Multi-language interface

Non-Functional Requirements

- **Security:** Encrypted data, secure authentication, role-based access
- **Usability:** Simple navigation for new users
- **Reliability:** Backup & recovery
- **Scalability:** Capable of expansion

Schedule

- **Week 1–3:** Requirements gathering and analysis
- **Week 4–7:** System design
- **Week 7–12:** Development of modules
- **Week 13–15:** Integration and security testing
- **Week 16:** Deployment and maintenance

Preliminary Budget

- **Software development:** ₹75,000
- **Hardware & Hosting:** ₹1,00,000
- **Database & Networking:** ₹30,000
- **Maintenance:** ₹10,000
- **Total initial cost estimation ≈ ₹2,00,000**

Class diagram:

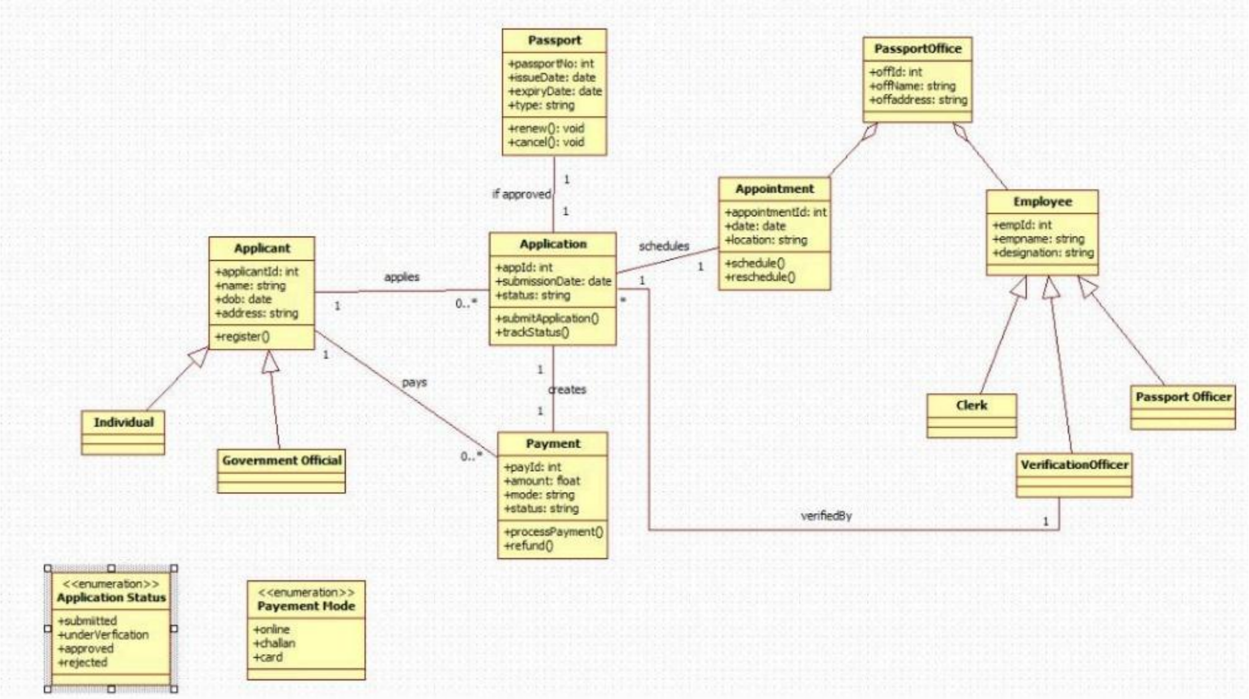

Fig 5.1 Class diagram of Passport authentication system

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:

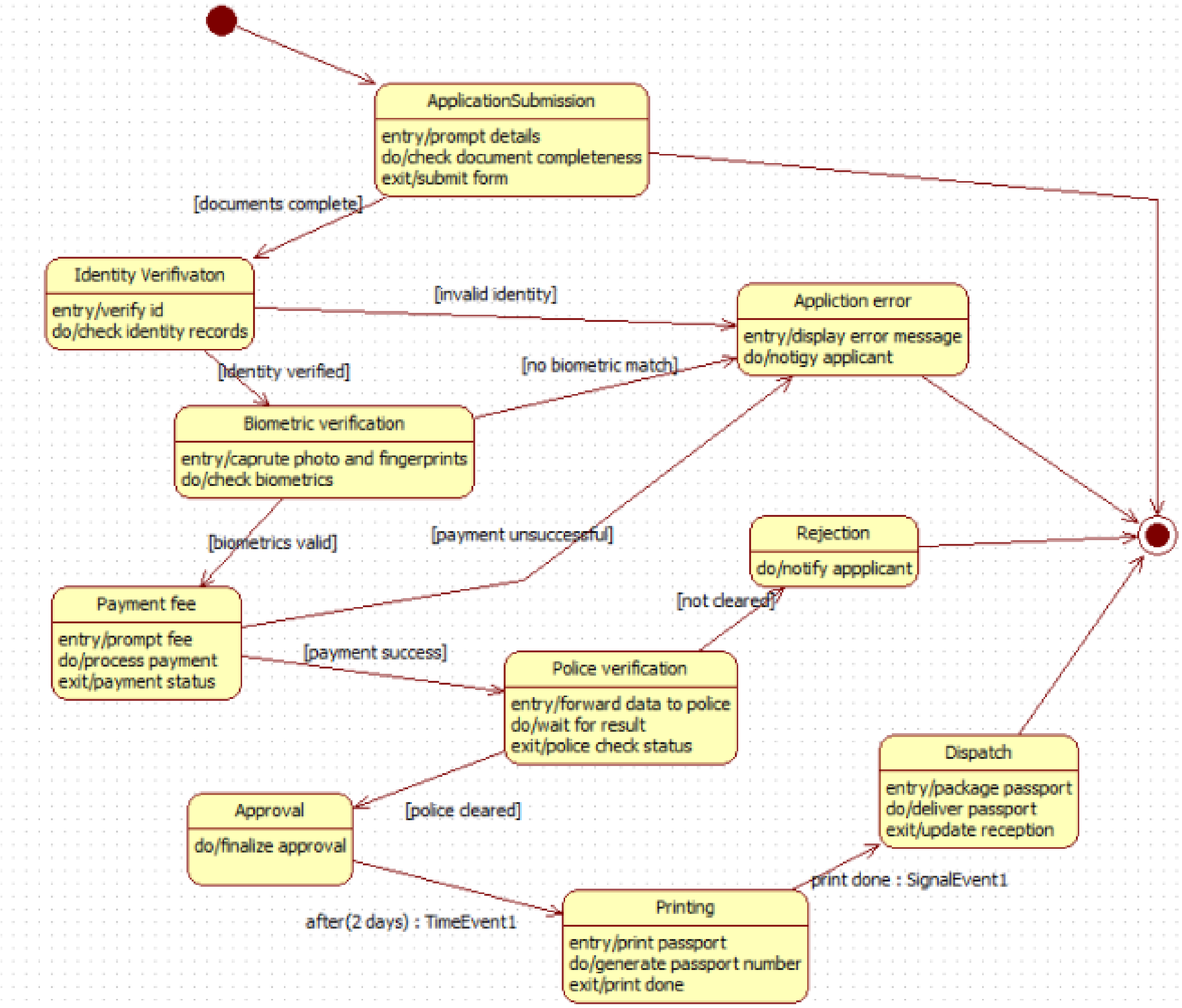

Fig 5.2 State diagram of Passport authentication system

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:

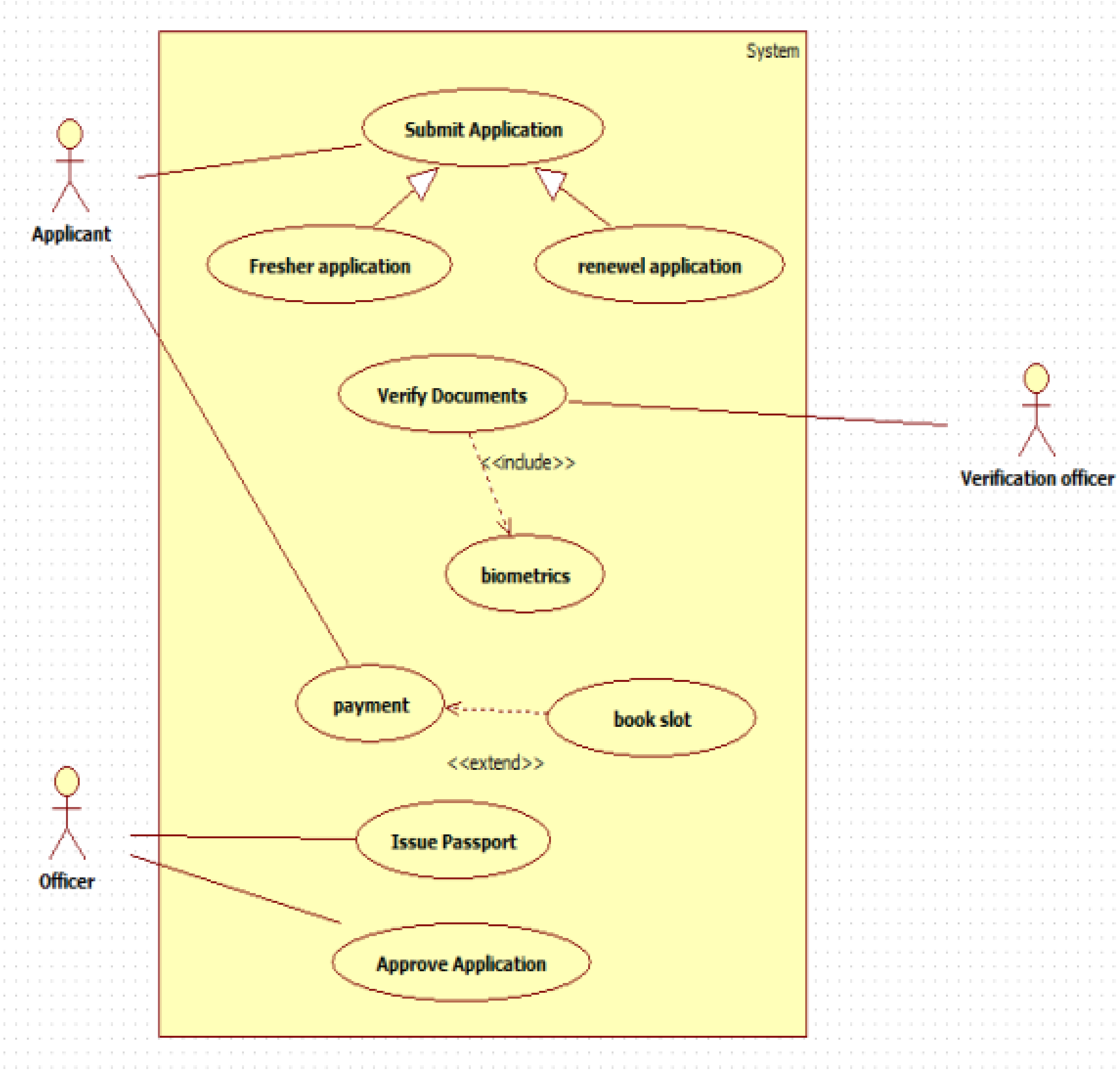

Fig 5.3 Use-case diagram of Passport authentication system

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:

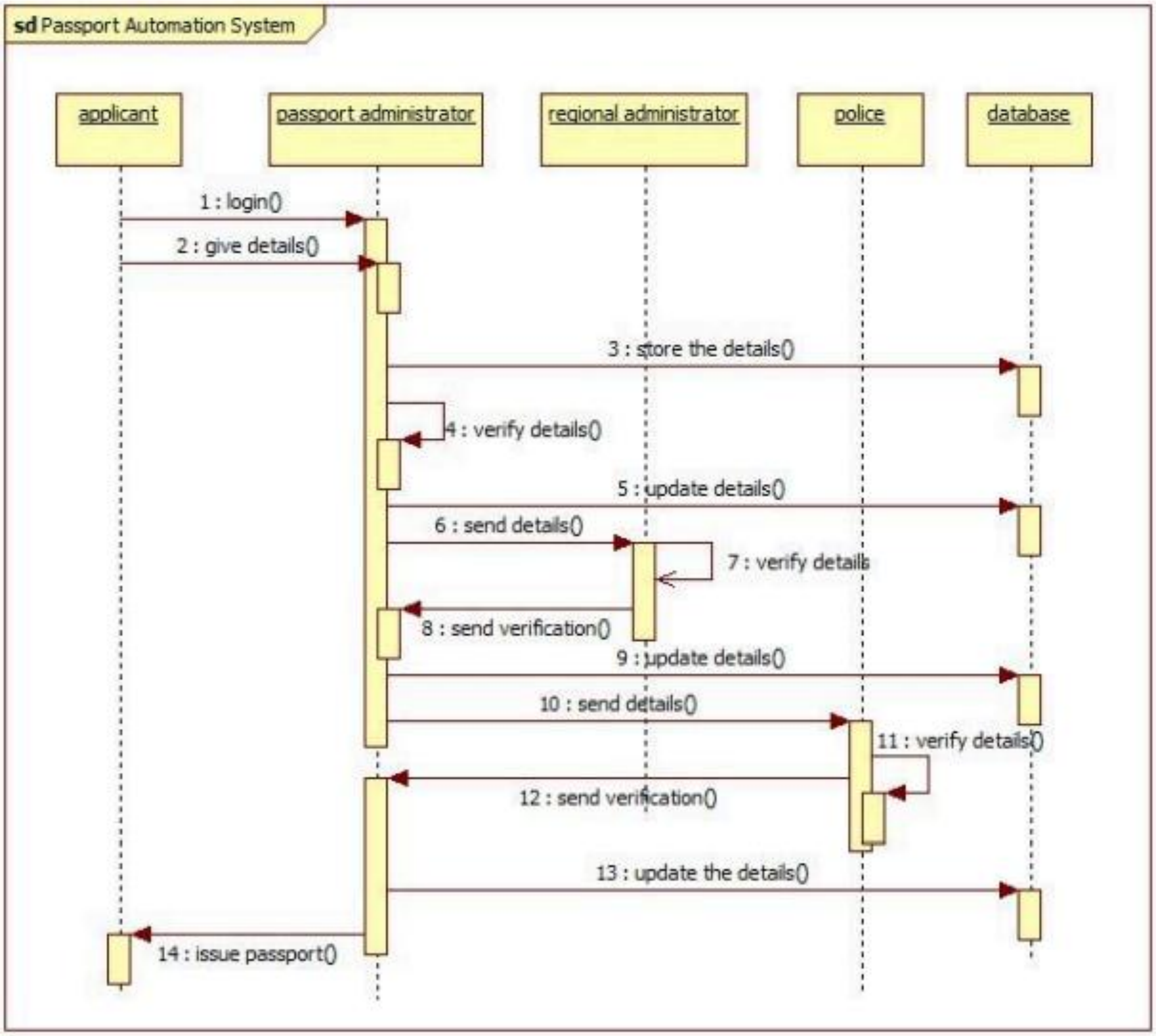

Fig 5.4 Sequence diagram of Passport authentication system

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:

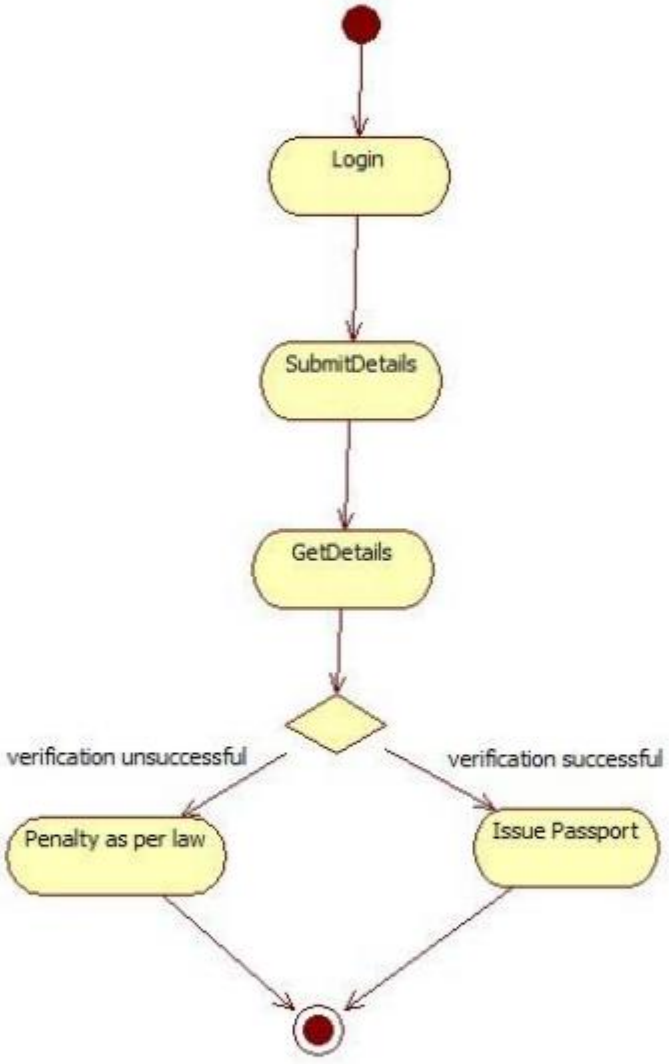

Fig 5.5 Activity diagram of Passport authentication system

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.