B.M.S. COLLEGE OF ENGINEERING BENGALURU

Autonomous Institute, Affiliated to VTU

Lab Record

Object-Oriented Modeling and Design(23CS5PCOOM)

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

Chirag S

1BM23CS079

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(23CS5PCOOM)** laboratory has been carried out by **Chirag S (1BM23CS079)** during the 5th Semester August 2025-December 2025

Signature of the Faculty In charge:

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

Table of Contents

1.HOTEL MANAGEMENT SYSTEM

System Requirement Document:

Problem statement :

A hotel needs a computerized system to streamline its daily operations, improve customer experience, and reduce manual workload. Currently, most tasks such as room booking, check-in/check-out, staff allocation, and bill generation are handled manually, which leads to errors, delays, and difficulty in maintaining records.

The Hotel Management System (HMS) should automate all major hotel activities and maintain accurate data in a structured way. The system must allow receptionists, managers, and customers to efficiently perform their respective tasks.

SRS-Software Requirements Specification

1. Introduction

1.1 Purpose

The purpose of this SRS document is to define the complete set of requirements for the Hotel Management System. It ensures a shared understanding between developers, management, and stakeholders about what the system will deliver. This document also establishes a base for design, development, and testing activities throughout the project lifecycle.

1.2 Scope

The system will automate hotel operations such as room reservations, guest check-in/check-out, room status updates, billing, and reporting. It will support hotel staff, administrators, and customers with accurate, real-time information. The scope also includes integration with booking platforms and secure payment methods required for operational efficiency.

1.3 Overview

The Hotel Management System provides a centralized platform for managing room availability, guest records, and financial operations. It is designed to reduce manual errors, improve service quality, ensure faster operations, and maintain all hotel data digitally.

2. General Description

The system will serve receptionists, managers, and administrators by providing a simple interface for routine tasks. It manages multiple room categories, guest profiles, service charges, and reservations. The system ensures smooth workflows by updating all modules in real time. It supports role-based access control, cloud storage, and multi-device access. The solution aims to increase operational efficiency and enhance customer satisfaction.

3. Functional Requirements

3.1 Reservation Management

- Users can create, modify, and cancel reservations.
- The system checks room availability based on date, room type, and occupancy.
- Reservation confirmations are automatically generated and sent via email/SMS.
- Supports advance booking, group booking, and reservation history tracking.

3.2 Room Management

- Maintains room information including type, price, and capacity.
- Updates real-time room status: Available, Occupied, Cleaning, or Maintenance.
- Assigns rooms automatically based on guest preferences and availability.
- Notifies housekeeping when rooms need cleaning or when guests check out.

3.3 Guest Management

- Stores guest personal details, identity proofs, preferences, and booking history.
- Supports fast check-in and check-out with digital verification.
- Maintains guest service usage data for billing purposes.
- Allows storing special notes for VIP or repeat guests.

3.4 Billing & Invoicing
- Generates bills automatically based on stay duration and services used.
- Supports UPI, cash, card, online wallet, and corporate billing.
- Provides printable and downloadable invoices.
- Maintains financial logs and tax calculations.

3.5 Reporting
- Generates daily, weekly, and monthly reservation and revenue reports.
- Provides room occupancy statistics and staff activity logs.

4. Interface Requirements
4.1 User Interface
- Clean and user-friendly dashboard with room status view, booking forms, and billing windows.
- Responsive UI accessible on desktops, tablets, and mobile devices.
- Different dashboards for admin, manager, and receptionist.
- Provides search functionality for guests, bookings, and rooms.

4.2 Integration Interfaces
- Integration with payment gateways for secure transactions.
- API support for integration with booking sites like Booking.com or OYO.
- Can integrate with hotel hardware such as printers, card readers, and POS systems.

5. Performance Requirements
- The system should respond to user actions (booking, billing, search) within 2 seconds.
- Must support at least 1000 concurrent users during peak seasons.
- Real-time updates must reflect room status changes instantly.
- System uptime should be at least 99% to avoid service interruptions.

6. Design Constraints
- Must run on standard hotel computers, printers, POS machines, and mobile devices.
- Requires a relational database such as MySQL for consistent data handling.
- Backend should be built using Java/Spring Boot, Python Django, or equivalent frameworks.
- Should operate smoothly with low-speed internet within hotel premises.

7. Non-Functional Attributes
7.1 Security -Must include authentication, role-based access control, encrypted data storage, and secure payment handling.
7.2 Reliability- The system should provide consistent performance with minimal downtime and support data backup and recovery.
7.3 Scalability -Should support increasing guests, rooms, and concurrent staff without affecting performance.
7.4 Portability -The software must run on multiple operating systems and device types with minimal setup.
7.5 Usability -Easy-to-understand interface with clear navigation for staff of all skill levels.
7.6 Reusability
Uses modular components that allow future upgrades and new features to be added easily.
7.7 Compatibility -Compatible with common web browsers, hotel hardware, and standard network environments.

8. Preliminary Schedule and Budget
The project is estimated to take 6 months including planning, analysis, design, development, testing,

and deployment phases. The approximate budget is $100,000, which includes hardware support, software development, third-party integrations, and maintenance.

Class Diagram:

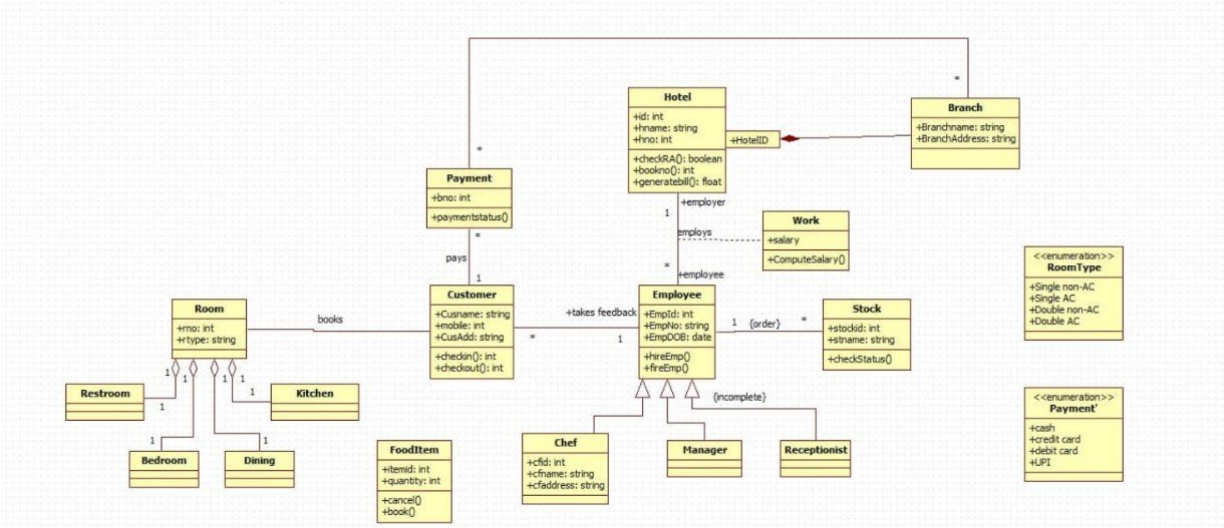

Fig 1.1 Class diagram of hotel management system

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:

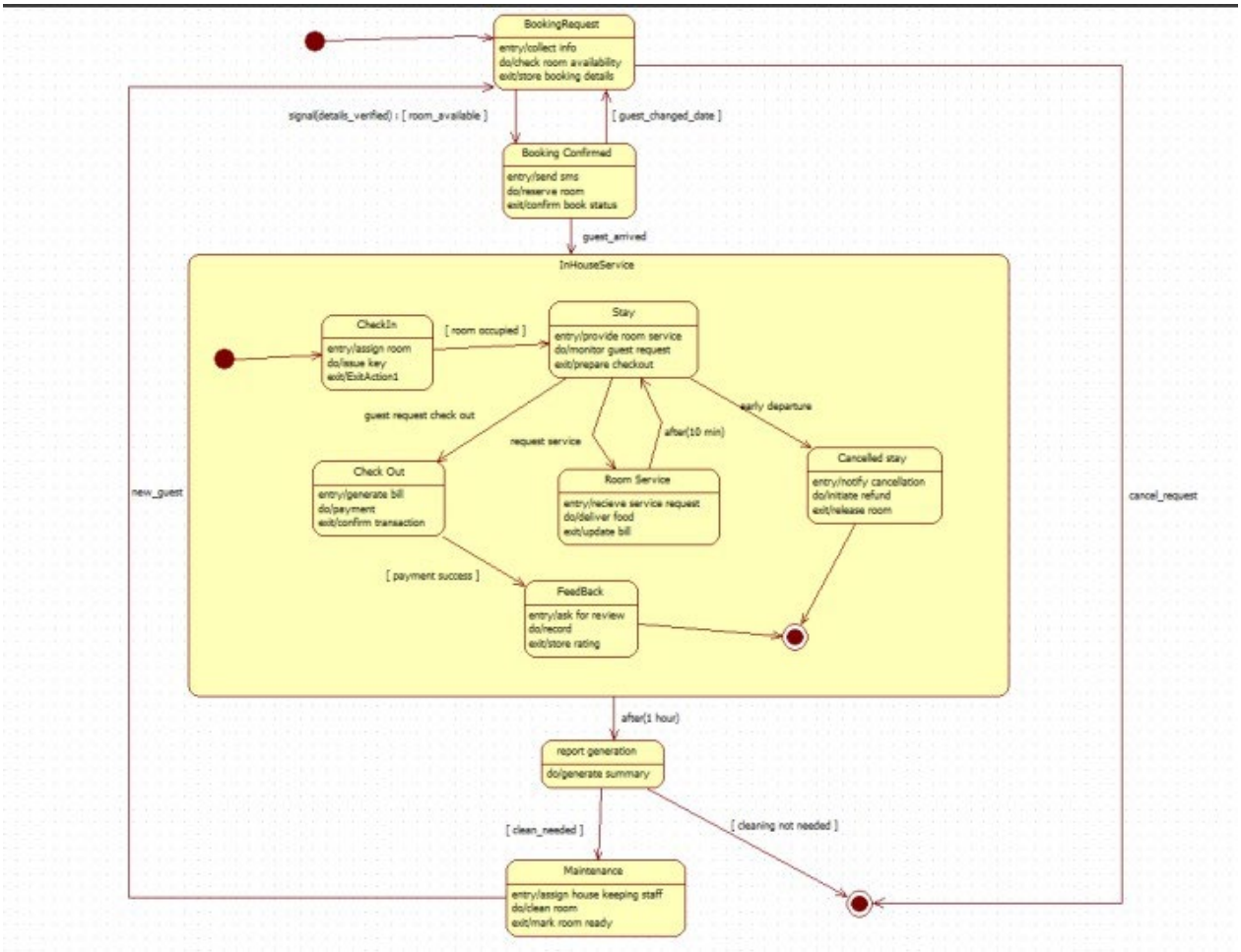

Fig 1.2 State diagram of hotel management system

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:

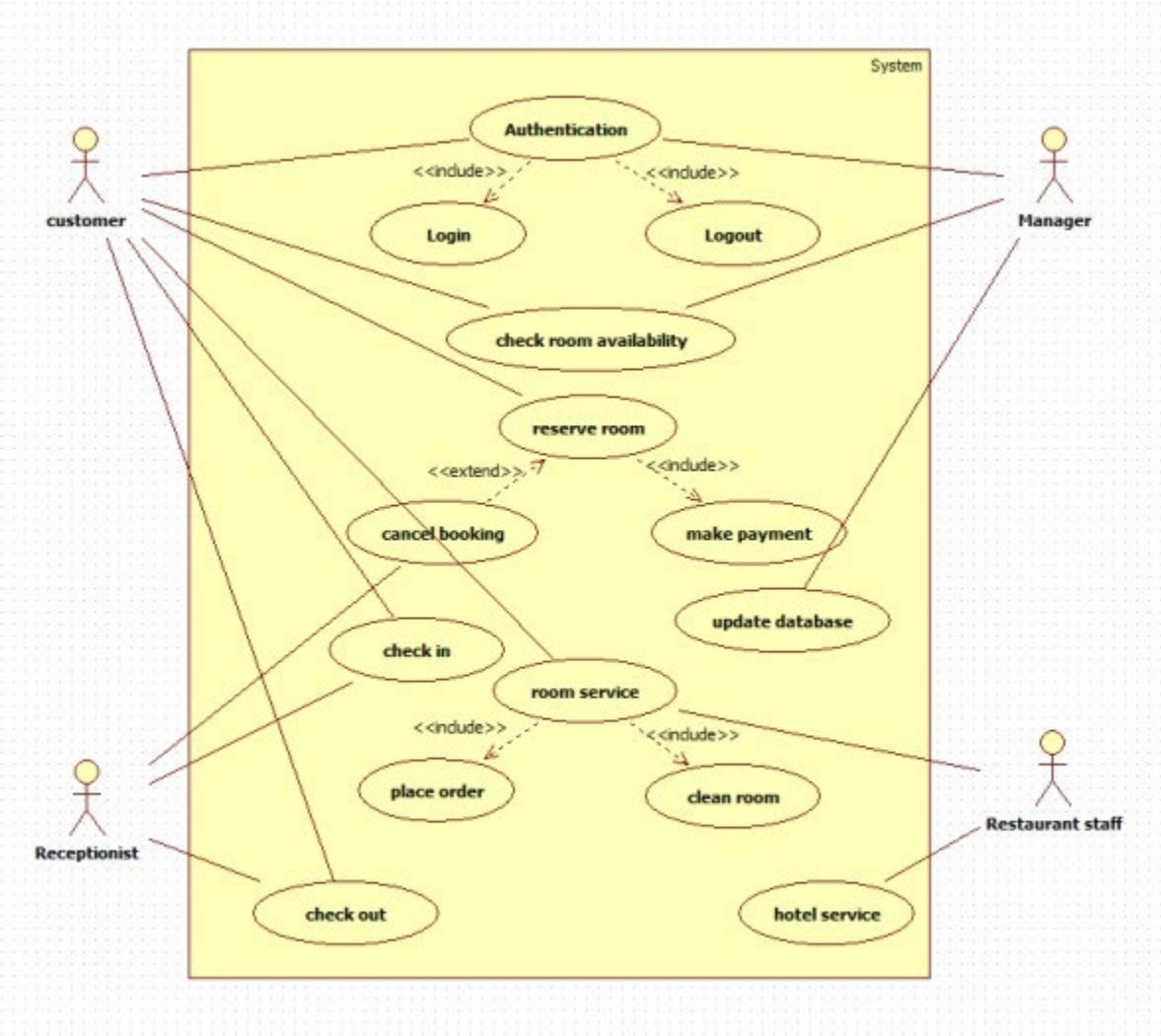

Fig 1.3 Use case diagram of hotel management system

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:

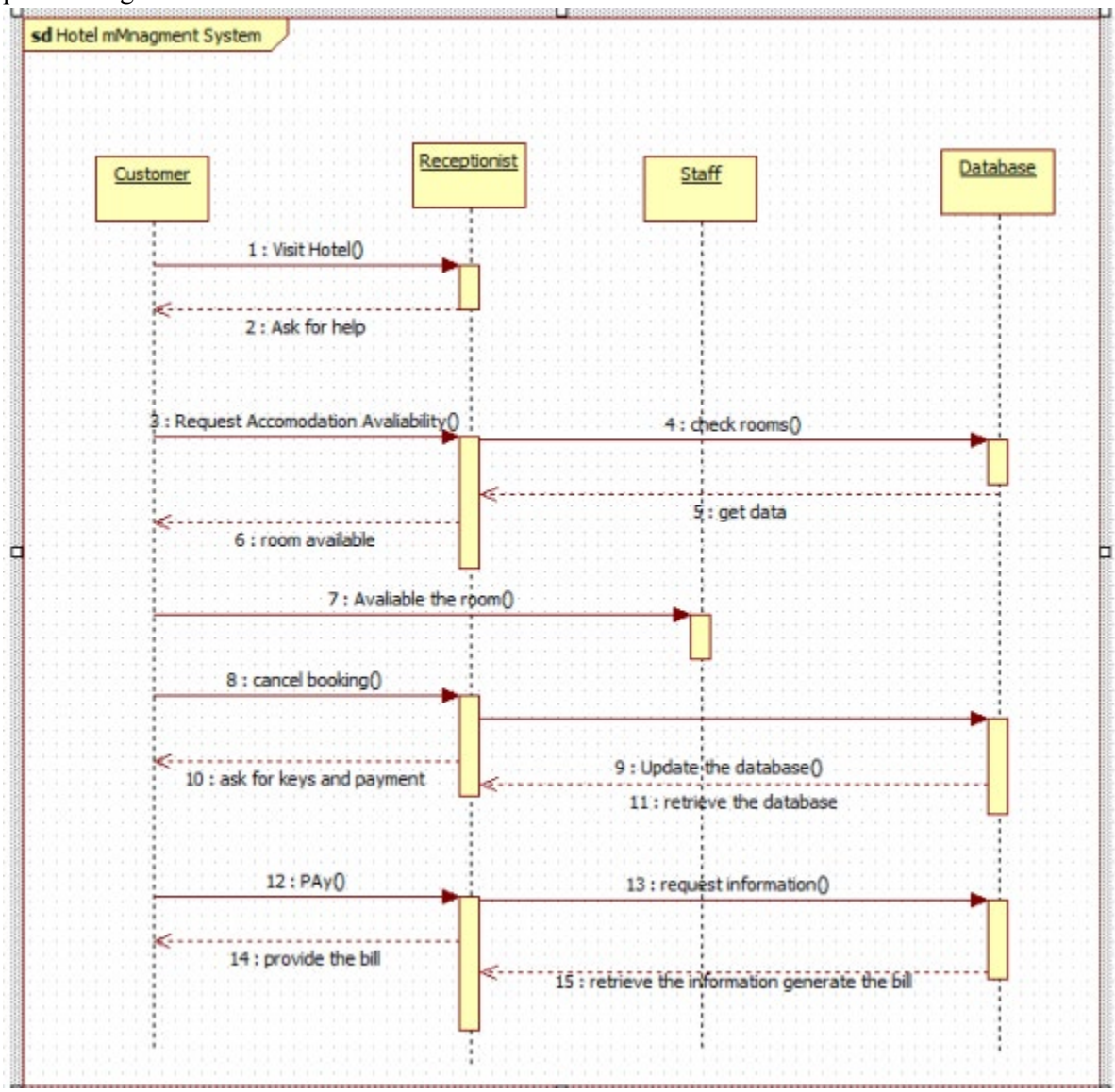

Fig 1.4 Sequence diagram of hotel management system

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:

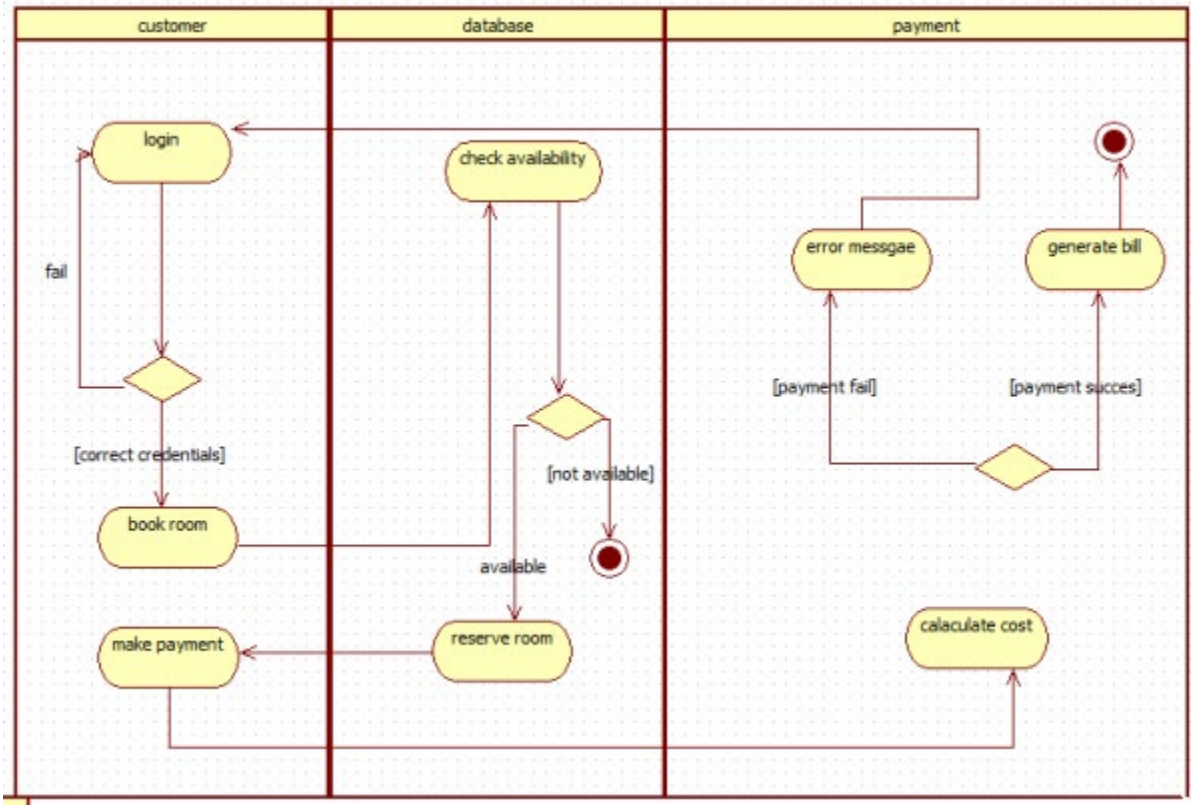

Fig 1.5 Activity diagram of hotel management system

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 MANAGEMENT SYSTEM

System Requirement Document:

Problem statement :

The existing manual and semi-automated credit card processing methods often lead to delays, security risks, and inconsistencies in transaction approvals. Financial institutions and merchants face challenges such as fraud detection, slow authorization response, and lack of real-time monitoring. A secure and efficient Credit Card Processing System is required to automate card verification, authorize transactions instantly, detect fraudulent activities, and ensure compliance with financial security standards. The system should enable smooth communication between merchants, banks, and payment networks to ensure fast, reliable, and secure payment processing.

SRS-Software Requirements Specification

1. Introduction

1.1 Purpose

The purpose of this SRS is to define all requirements for the Credit Card Processing System, ensuring secure card validation, transaction authorization, and fraud prevention. It serves as a foundation for developers, testers, and stakeholders to understand expected system behavior and technical constraints.

1.2 Scope

The system handles card verification, payment authorization, settlement processing, refunds, transaction logs, and fraud alerts. It connects merchants, payment gateways, and banks to facilitate seamless and secure financial transactions.

1.3 Overview

This system ensures quick, reliable, and compliant credit card processing while reducing risks associated with fraud. It provides dashboards for banks and merchants, supports large-scale transactions, and ensures secure communication with external financial networks.

2. General Description

The system acts as an intermediate layer between customer card data, merchant terminals, and banking servers. It validates card details, approves transactions, and records financial activities securely. The system maintains high reliability through redundancy, real-time processing, and auditing. User-friendly dashboards allow merchants and bank officials to monitor transactions, settlement cycles, and fraud alerts.

3. Functional Requirements

3.1 Card Verification

- Validates card number (Luhn check), expiry date, CVV, and cardholder information.
- Confirms card status through issuer bank (active, blocked, stolen, expired).
- Performs optional OTP or two-factor authentication for added security.

3.2 Transaction Processing

- Handles full payments, partial payments, reversals, and refunds in real time.

- Connects to banks using secure APIs to authorize or decline transactions.
- Generates unique transaction IDs and logs all financial records for audits.

3.3 Fraud Detection

- Uses rule-based and pattern analysis to detect suspicious transactions.
- Blocks questionable activities and sends alerts to merchants or banks.
- Maintains blacklists for stolen or compromised cards.

3.4 Merchant Management

- Stores merchant details, business category, and settlement information.
- Provides transaction reports, settlement summaries, and analytics.
- Supports monthly reconciliation to ensure financial accuracy.

4. Interface Requirements

4.1 User Interface

- Merchant dashboard for viewing daily transactions, settlements, refunds, and disputes.
- Bank admin dashboard for monitoring approvals, fraud alerts, and system health.
- Clean layout with charts, filters, and downloadable reports.

4.2 Integration Interfaces

- APIs for connecting with Visa, Mastercard, RuPay, and bank authorization servers.
- Secure HTTPS-based communication for payment gateway integration.
- Supports card readers, POS machines, and mobile payment devices.

5. Performance Requirements

- System must authenticate and process transactions within 1 second.
- Must support more than 10,000 concurrent transactions without failure.
- Fraud detection algorithms should run in real time with minimal delay.
- System uptime must be above 99.9% due to financial sensitivity.

6. Design Constraints

- Must comply with PCI-DSS and banking security standards.
- Requires encrypted communication (HTTPS, SSL/TLS).
- Must integrate with payment networks like Visa, Mastercard, and RuPay via APIs.
- Should run on high-availability cloud or data center infrastructure.

7. Non-Functional Attributes

The system must ensure high encryption, authentication, and PCI-DSS compliance. It should support high availability, load balancing, and fast performance even under heavy transaction volumes. Portability across cloud servers and devices is required. The UI must be intuitive, and modules should be reusable for future upgrades. Compatibility with major browsers and

accurate financial data integrity is essential.

8. Schedule and Budget

The project development is estimated to take 7–8 months including analysis, security compliance, integration, testing, and deployment. The total budget is approximately $150,000, covering secure API development, fraud detection modules, cloud infrastructure, and maintenance.

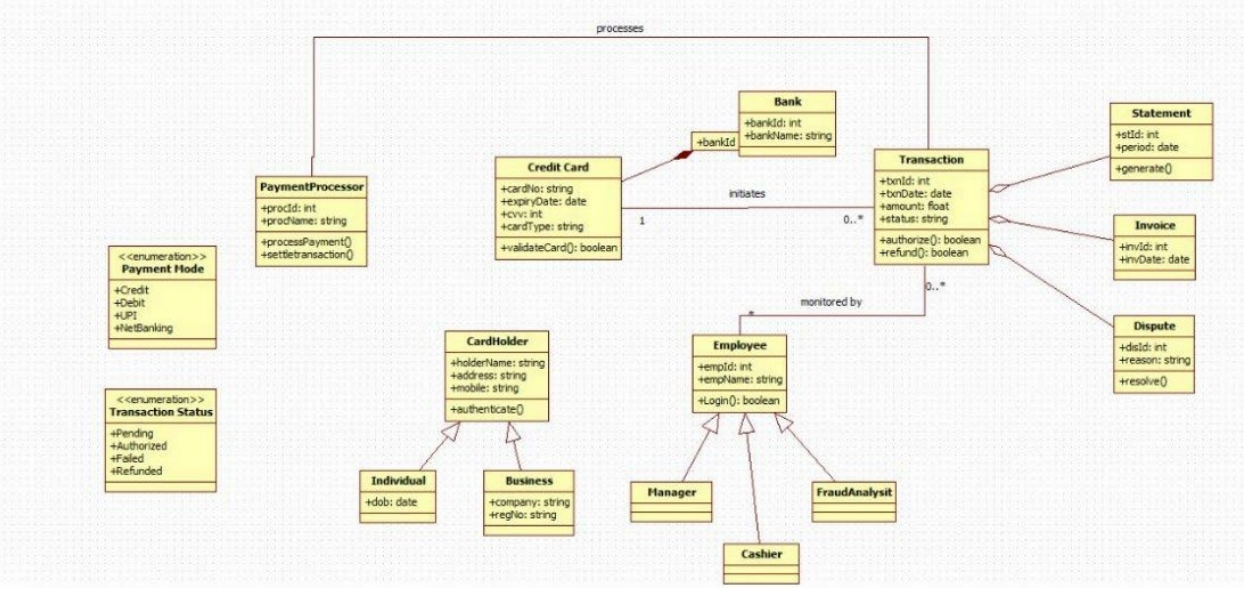

Fig 2.1 Class diagram of Credit card processing system

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:

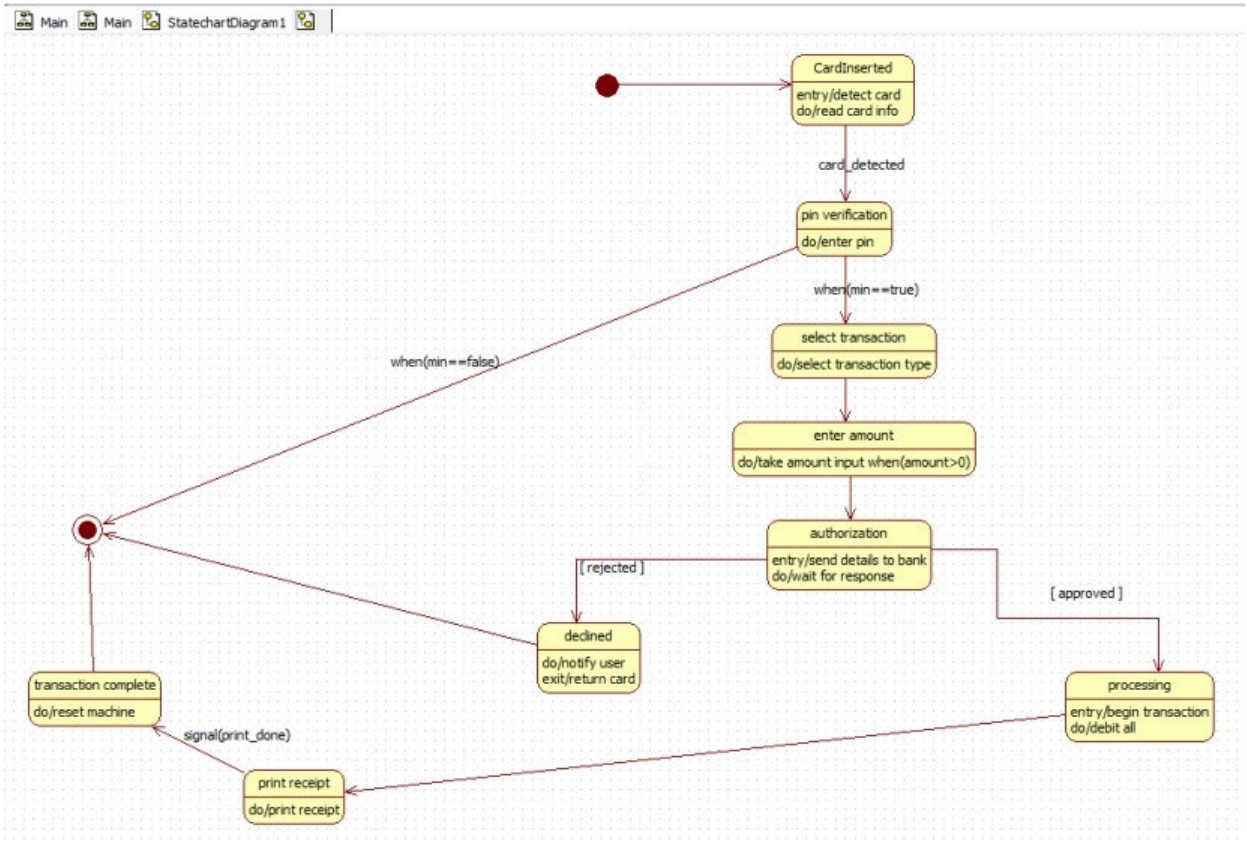

Fig 2.2 State diagram of Credit card processing system

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:

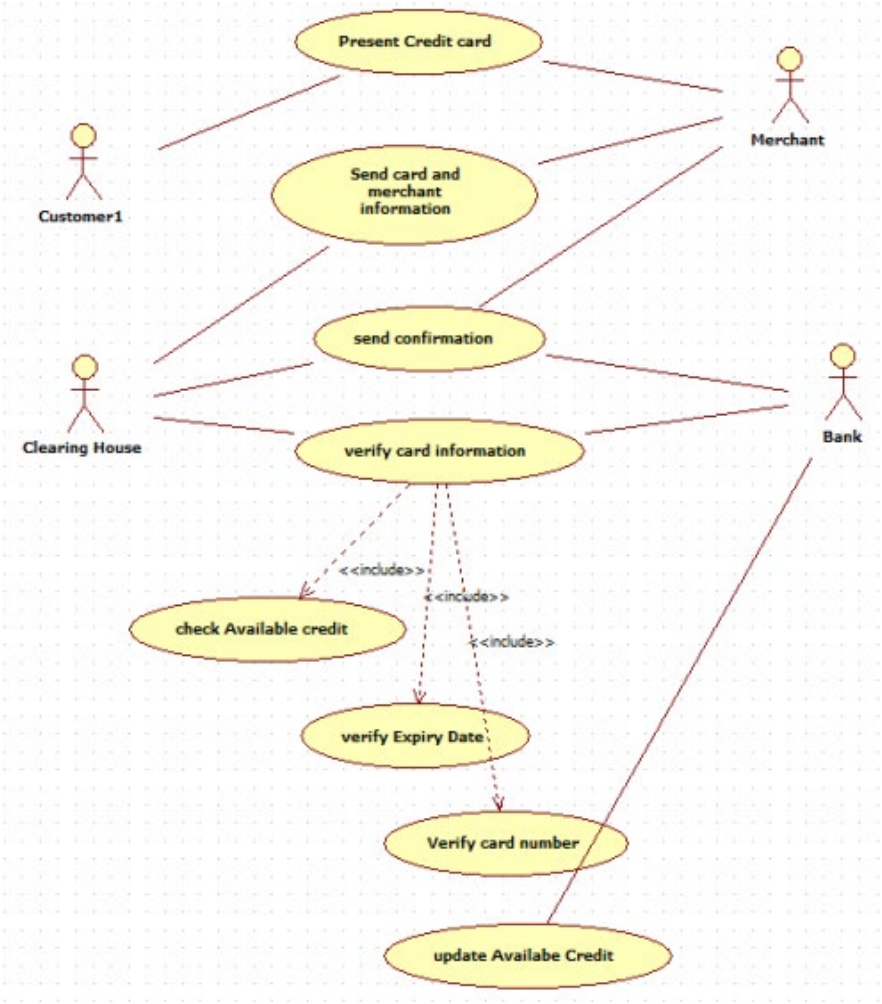

Fig 2.3 Use-case diagram of Credit card processing system

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:

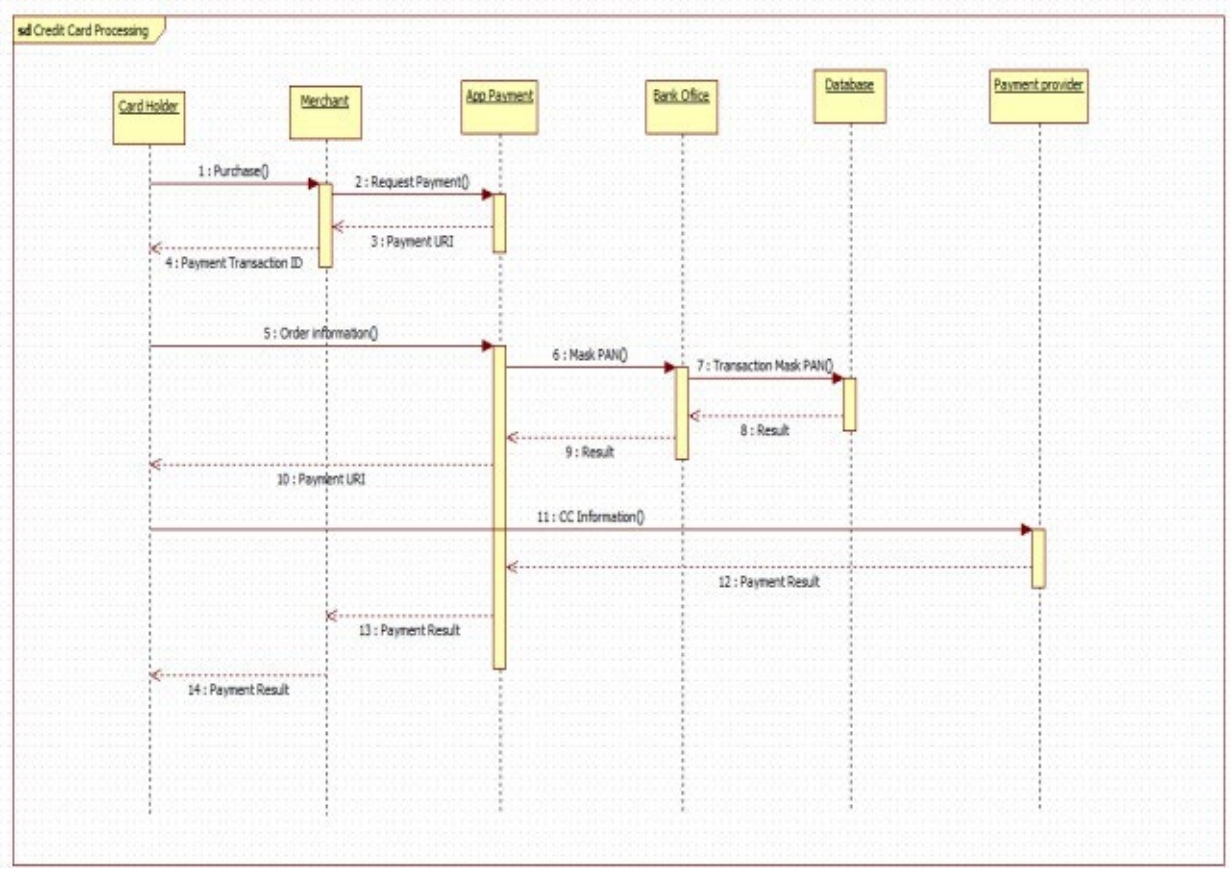

Fig 2.4 Sequence diagram of Credit card processing system

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:

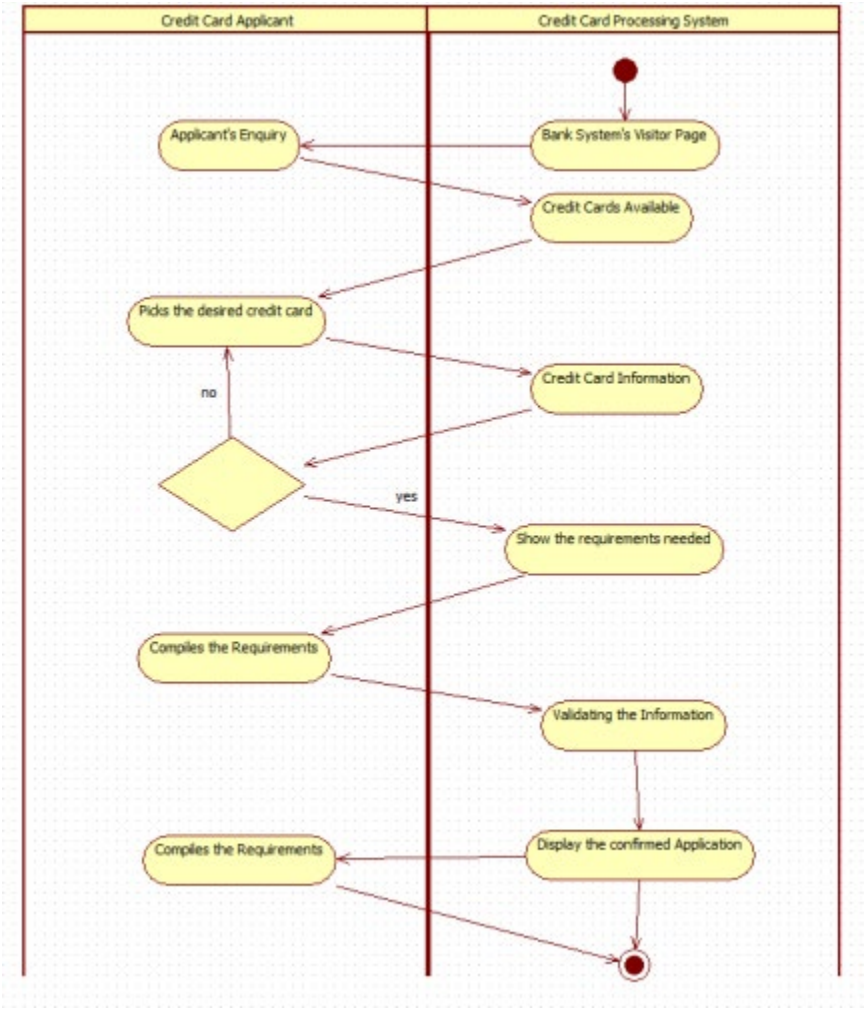

Fig 2.5 Activity diagram of Credit card processing system

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

System Requirement Document:

Problem statement :

Traditional library management relies heavily on manual book issuance, record keeping, and fine calculation, which leads to errors, loss of data, and inefficiency during peak hours. Students and staff often face difficulty in finding books, checking availability, and tracking due dates. Therefore, an automated Library Management System is needed to streamline book cataloging, member registration, lending/return operations, and fine management. The system should provide quick search capabilities, maintain accurate inventory records, and improve overall library operations with reduced manual workload.

SRS-Software Requirements Specification

1. Introduction

1.1 Purpose

This document defines the functional and technical requirements for the Library Management System. It provides clear guidance for developers and stakeholders regarding the system's expected features and constraints.

1.2 Scope

The system covers book cataloging, member registration, lending/return operations, fine calculation, searching, and reporting. It aims to digitalize library tasks and improve accuracy and efficiency.

1.3 Overview

The Library Management System replaces manual processes with automated workflows, enabling easy management of books, users, transactions, and inventory.

2. General Description

The system supports librarians, students, and administrators. It maintains a database of books, authors, categories, and member profiles. Features include borrowing rules, fine calculation, notifications, and inventory tracking. The system ensures quick searching and reduces workload.

3. Functional Requirements

3.1 Book Catalog Management

- Add, edit, delete, and categorize books.
- Maintain information such as title, author, ISBN, and availability.
- Track lost, damaged, or reserved books.

3.2 Member Management

- Register new members and store personal details.
- Track borrowing history and membership validity.
- Generate membership cards if needed.

3.3 Lending & Return Management

- Issue and return books with date tracking.
- Auto-calculate late fees and generate payment receipts.
- Display due date reminders to members.

3.4 Search and Reporting

- Search books by title, author, category, or ISBN.
- Generate daily transaction reports, inventory summaries, and user activity logs.

4. Interface Requirements

4.1 User Interface

- Clean dashboard for librarians with book inventory and member lists.
- Member portal for viewing borrowed books and due dates.
- Designed to be simple and user-friendly.

4.2 Integration Interfaces

- Barcode scanner integration for fast book issuing.
- Optional integration with online library portals.

5. Performance Requirements

- Searching books or members should take less than 2 seconds.
- Must support up to 500 users at the same time, including students and staff.
- Issuing and returning transactions should update instantly across modules.
- System should operate without lag even during semester peak days.

6. Design Constraints

- Should run on existing library computers with typical hardware specs.
- Requires a relational database like MySQL/PostgreSQL to store book records.
- Must support barcode scanners and optional RFID technology.
- Should function offline with local database access if internet is slow.

7. Non-Functional Attributes

Includes secure access, reliability for continuous operations, scalable database, easy UI navigation, cross-platform compatibility, modular design for reuse, and consistent data accuracy across modules.

8. Schedule and Budget

The project requires approximately 4–5 months with a budget of $40,000 for software development, database setup, and testing.

Class Diagram:

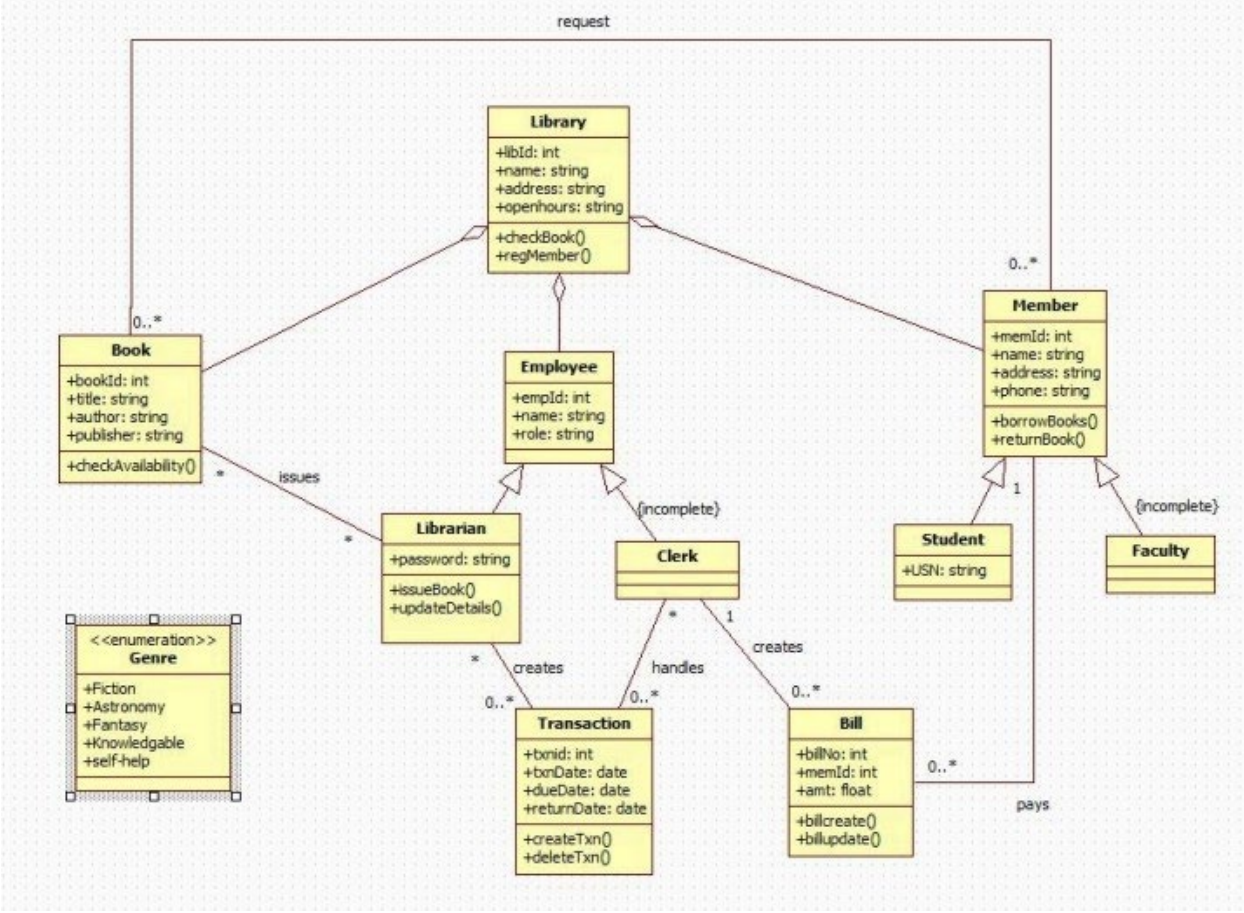

Fig 3.1 Class diagram of Library management system

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:

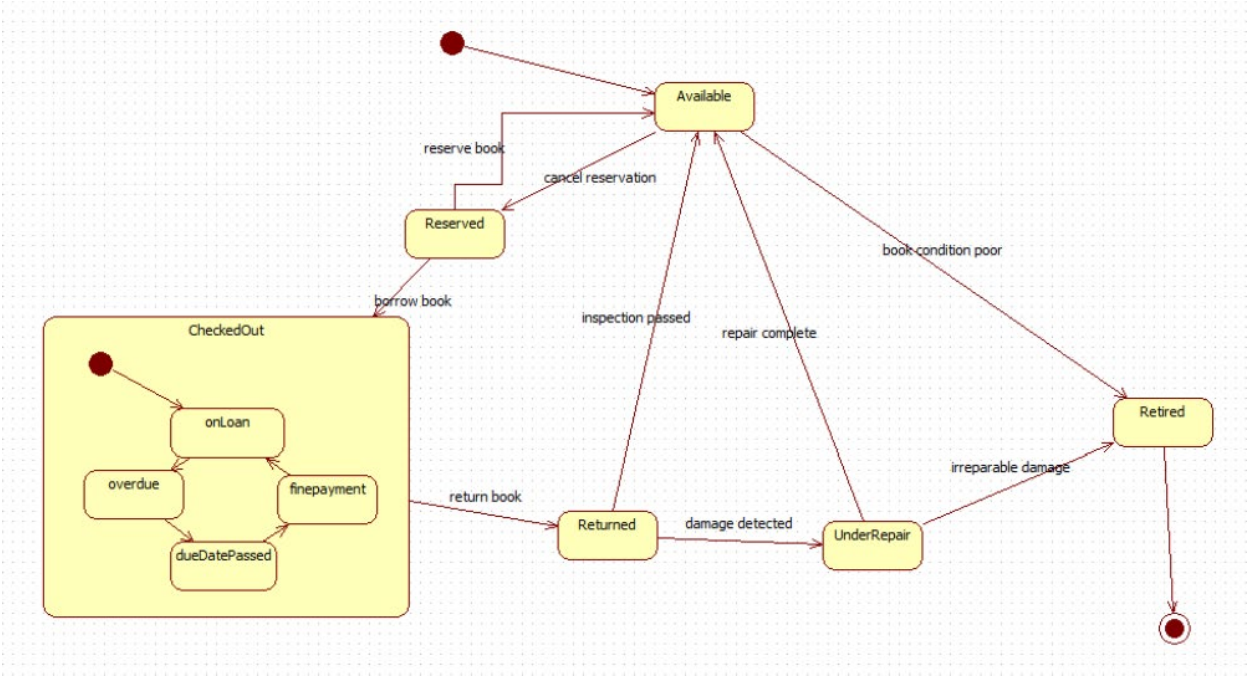

Fig 3.2 State diagram of Library management system

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:

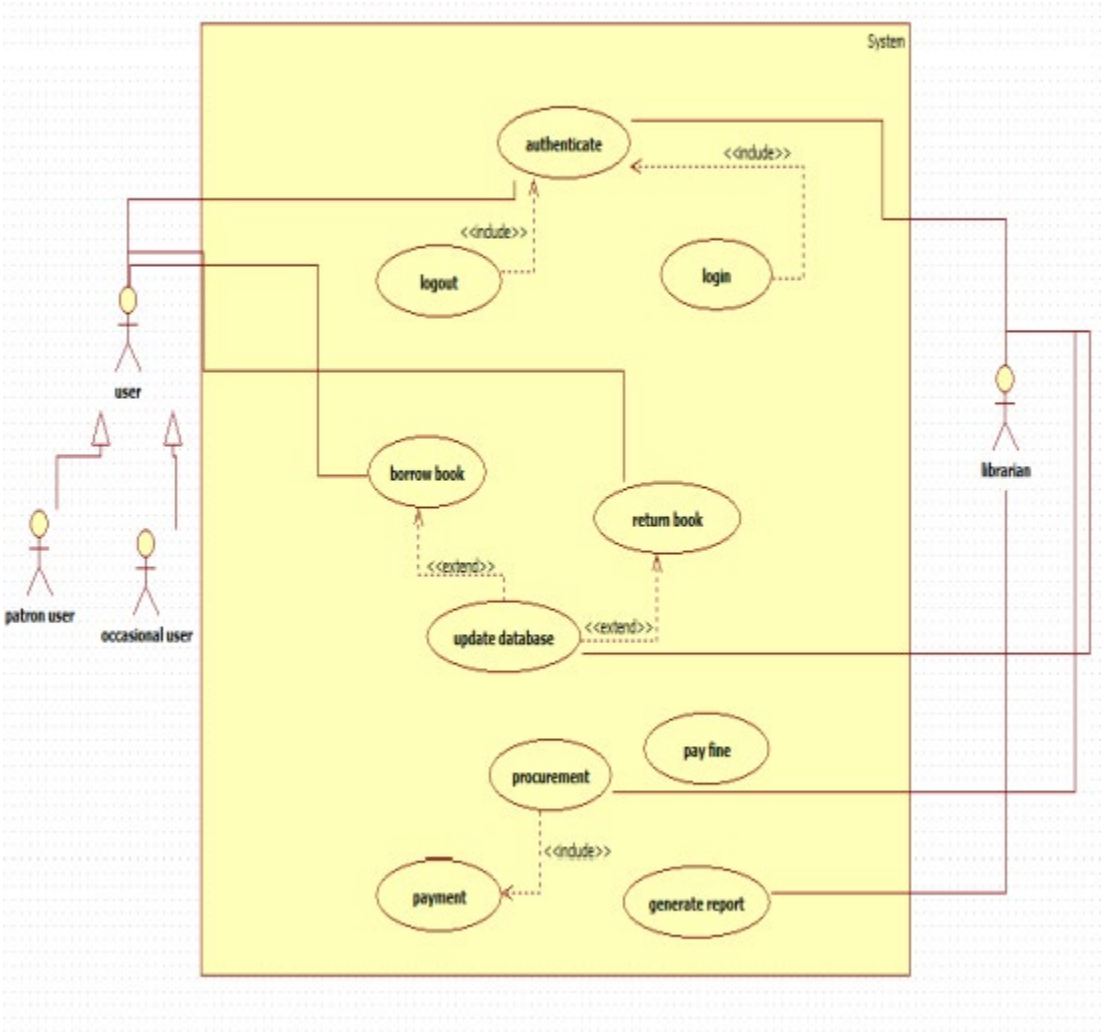

Fig 3.3 Use case diagram of Library management system

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:

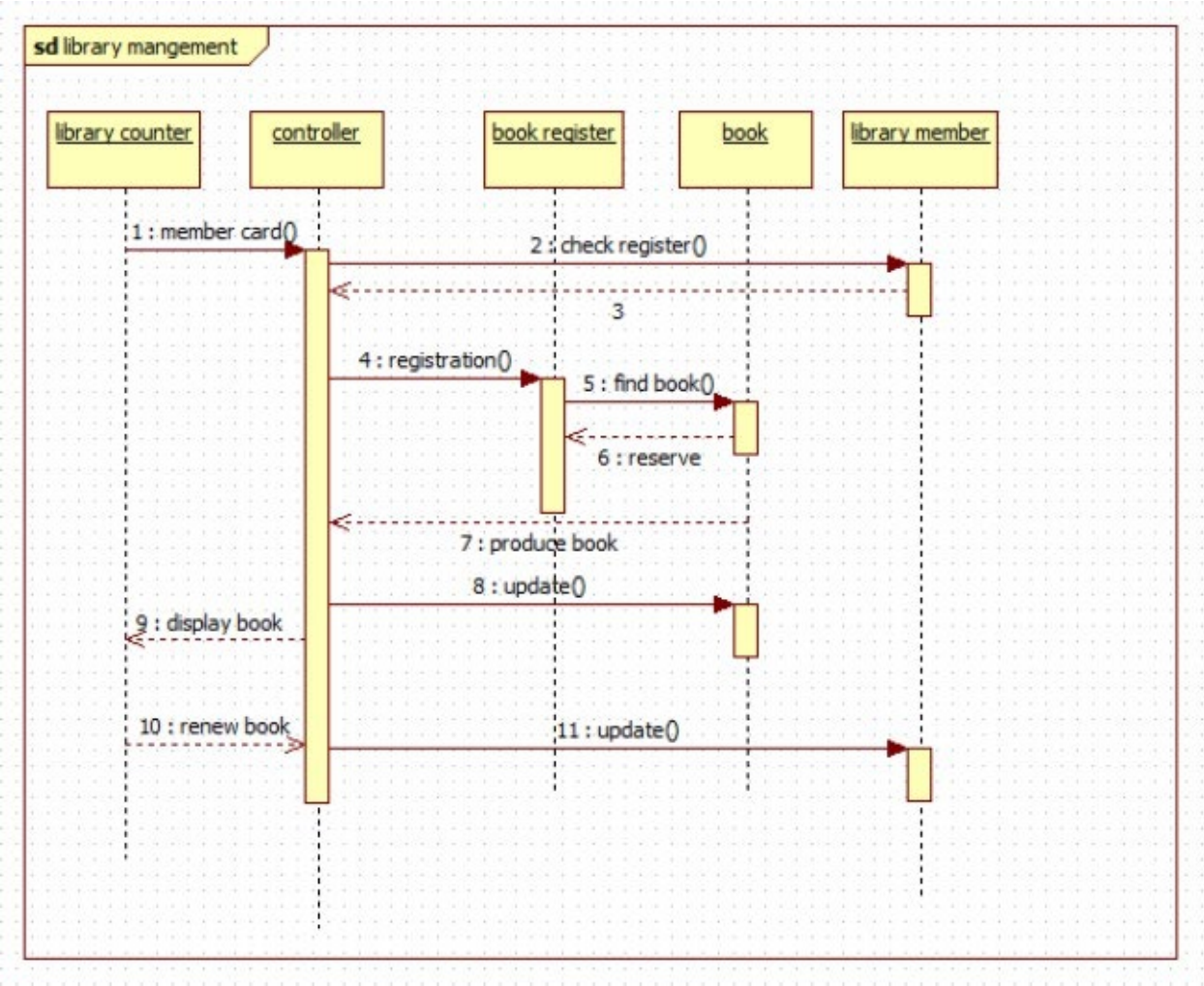

Fig 3.4 Sequence diagram of Library management system

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:

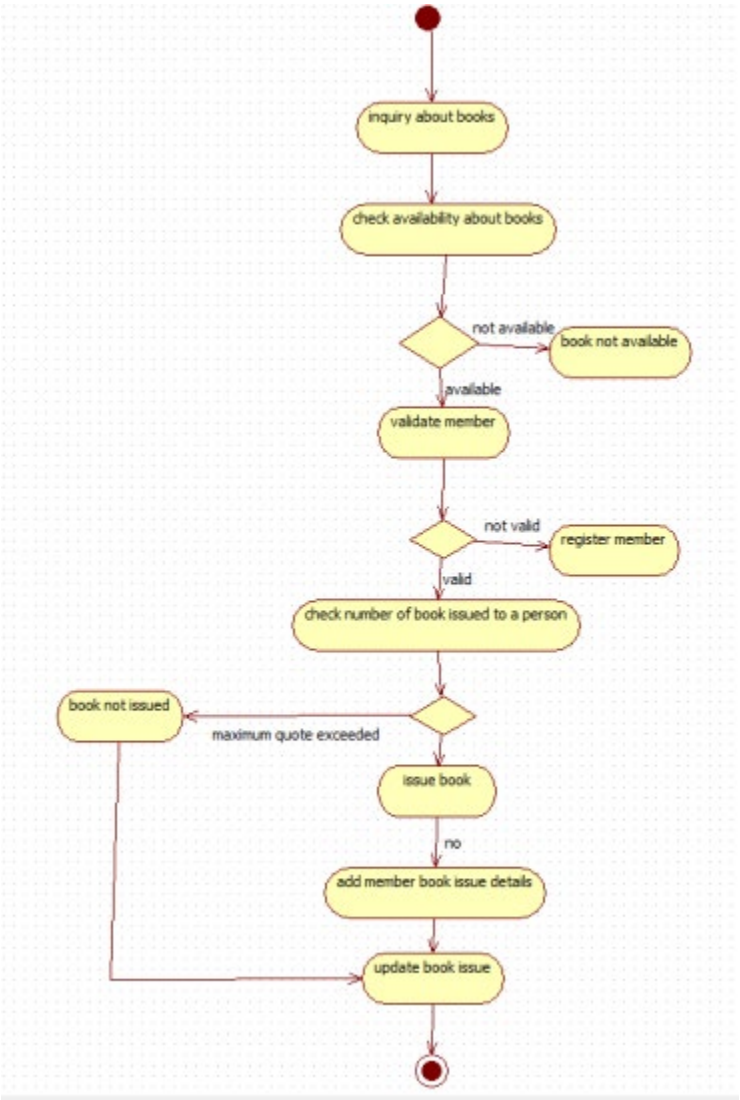

Fig 3.5 Activity diagram of Library management system

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

System Requirement Document:

Problem statement :

Businesses often struggle to maintain accurate stock details, record stock movements, track damaged goods, and ensure timely replenishment due to manual monitoring methods. This leads to inconsistencies, loss of inventory, and delays in decision-making. A Stock Maintenance System is required to efficiently record stock entries, monitor stock levels, track item conditions, update stock usage, and generate maintenance logs. The system should help staff maintain accurate inventory records, reduce manual errors, and support smooth operational flow.

SRS-Software Requirements Specification

1. Introduction

1.1 Purpose

The purpose of this SRS is to define the functional and non-functional requirements of the Stock Maintenance System. It ensures that developers, testers, and stakeholders understand the system goals, behavior, and technical expectations.

1.2 Scope

The system covers stock entry management, updating stock usage, tracking damaged/expired goods, generating stock maintenance reports, and sending low-stock alerts. It supports warehouse staff, store managers, and administrators.

1.3 Overview

The Stock Maintenance System helps businesses keep accurate stock records by automating the monitoring and updating of inventory details. It reduces manual errors, improves stock visibility, and ensures timely maintenance actions.

2. General Description

The system is used by warehouse workers, store managers, and admin staff to manage stock quantities, condition status, and movement history. It stores item details, consumption patterns, stock condition reports, and maintenance logs. The system ensures up-to-date stock information and provides alerts for low or damaged items to help prevent shortages or wastage.

3. Functional Requirements

3.1 Stock Entry Management

- Allows adding new stock items with details like name, category, quantity, and condition.
- Supports editing and deleting stock entries.
- Maintains purchase date and supplier information.

3.2 Stock Usage Updating

- Reduces quantities when items are issued or consumed.
- Maintains usage logs for tracking consumption history.

- Allows returned or unused stock updates.

3.3 Damage & Expiry Tracking

- Marks items as damaged, expired, or unavailable.
- Stores reason codes and timestamps for all damaged entries.
- Supports separate listing for unusable stock.

3.4 Low-Stock Alerts

- Sends alerts when stock falls below minimum threshold.
- Helps managers plan timely reordering.

3.5 Reporting

- Generates daily, weekly, or monthly stock maintenance reports.
- Provides summaries for damaged, consumed, and available stock.

4. Interface Requirements

4.1 User Interface

- Simple dashboard displaying total stock, low-stock items, and damaged items.
- Easy-to-use forms for adding or updating stock.
- Supports search and filter options for fast navigation.

4.2 Integration Interfaces

- Supports barcode scanners for item entry.
- Can integrate with accounting or purchase order systems if required.

5. Performance Requirements

- Updates to stock quantities must reflect in real time.
- Should handle 200+ stock updates per hour without delay.
- System responses (search, update, add) should be under 2 seconds.
- Must maintain accuracy even with large stock volumes (10,000+ items).

6. Design Constraints

- Must support existing store hardware like barcode readers and standard computers.
- Requires a stable relational or NoSQL database for large inventory sets.
- Must run on commonly used OS (Windows/Linux) and support major browsers.
- Should support periodic backup to prevent data loss.

7. Non-Functional Attributes

- **Security:** Role-based access for staff and admin.
- **Reliability:** Should operate continuously with minimal downtime.
- **Scalability:** Able to manage increasing stock items in the future.
- **Usability:** Interfaces must be simple for fast entry operations.

- **Portability:** Should run on desktops, laptops, and web-based systems.
- **Reusability:** Modular design for adding new inventory features.
- **Compatibility:** Works with barcode scanners, POS, and web browsers.
- **Data Integrity:** Ensures correct and consistent stock data at all times.

8. Schedule and Budget

Estimated development time is 5–6 months including design, implementation, testing, and deployment. The expected budget is $70,000, covering database design, UI development, system integration, testing, and maintenance support.

Class Diagram:

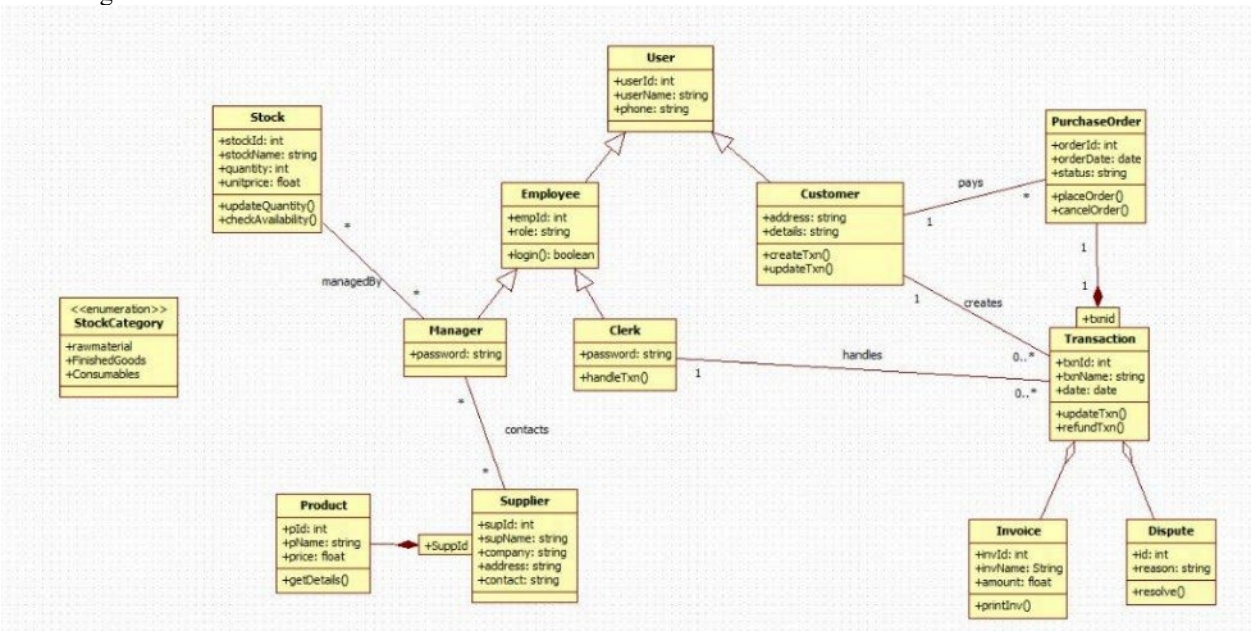

Fig 4.1 Class diagram of Stock management system

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:

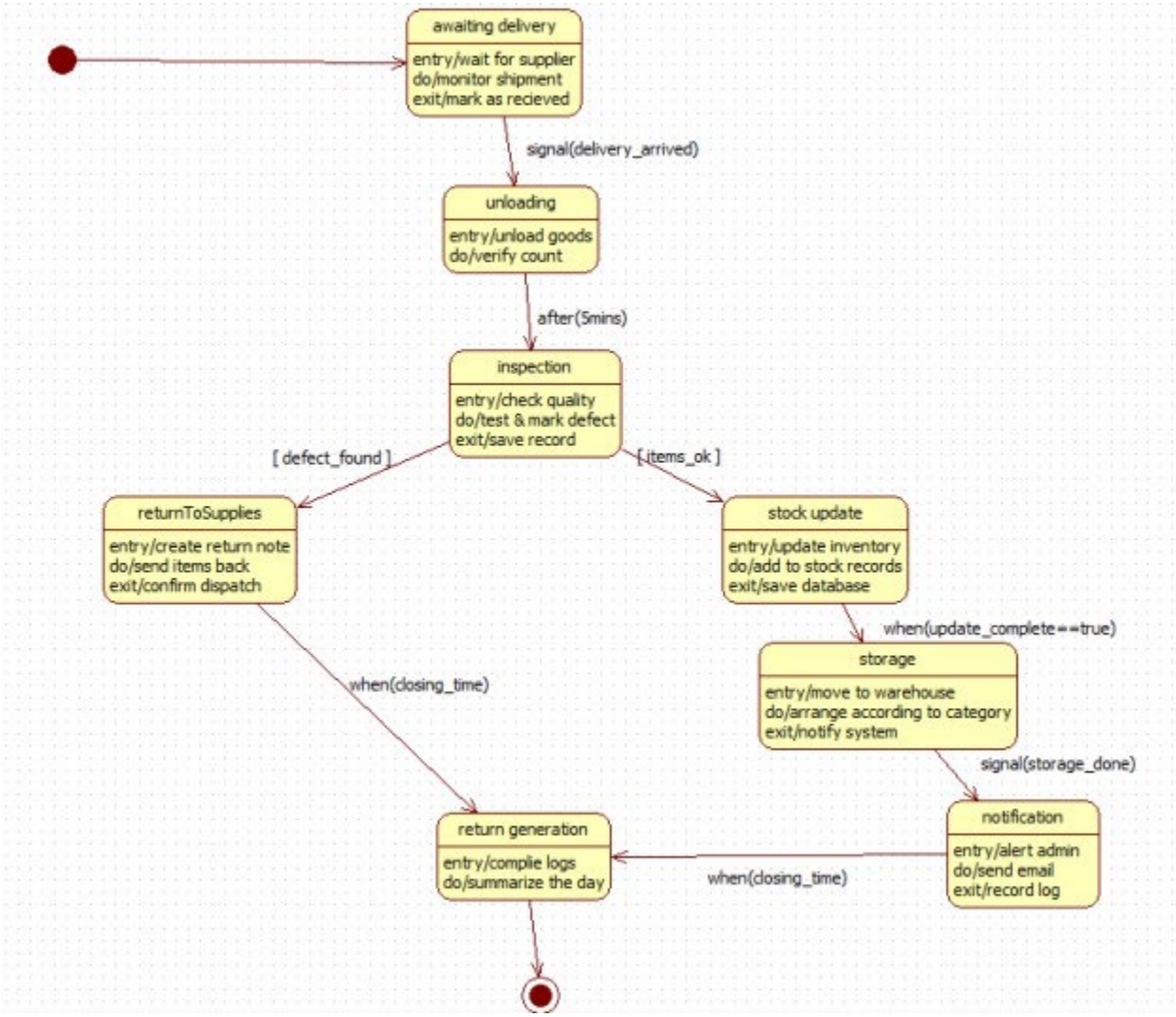

Fig 4.2 State diagram of Stock management system

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:

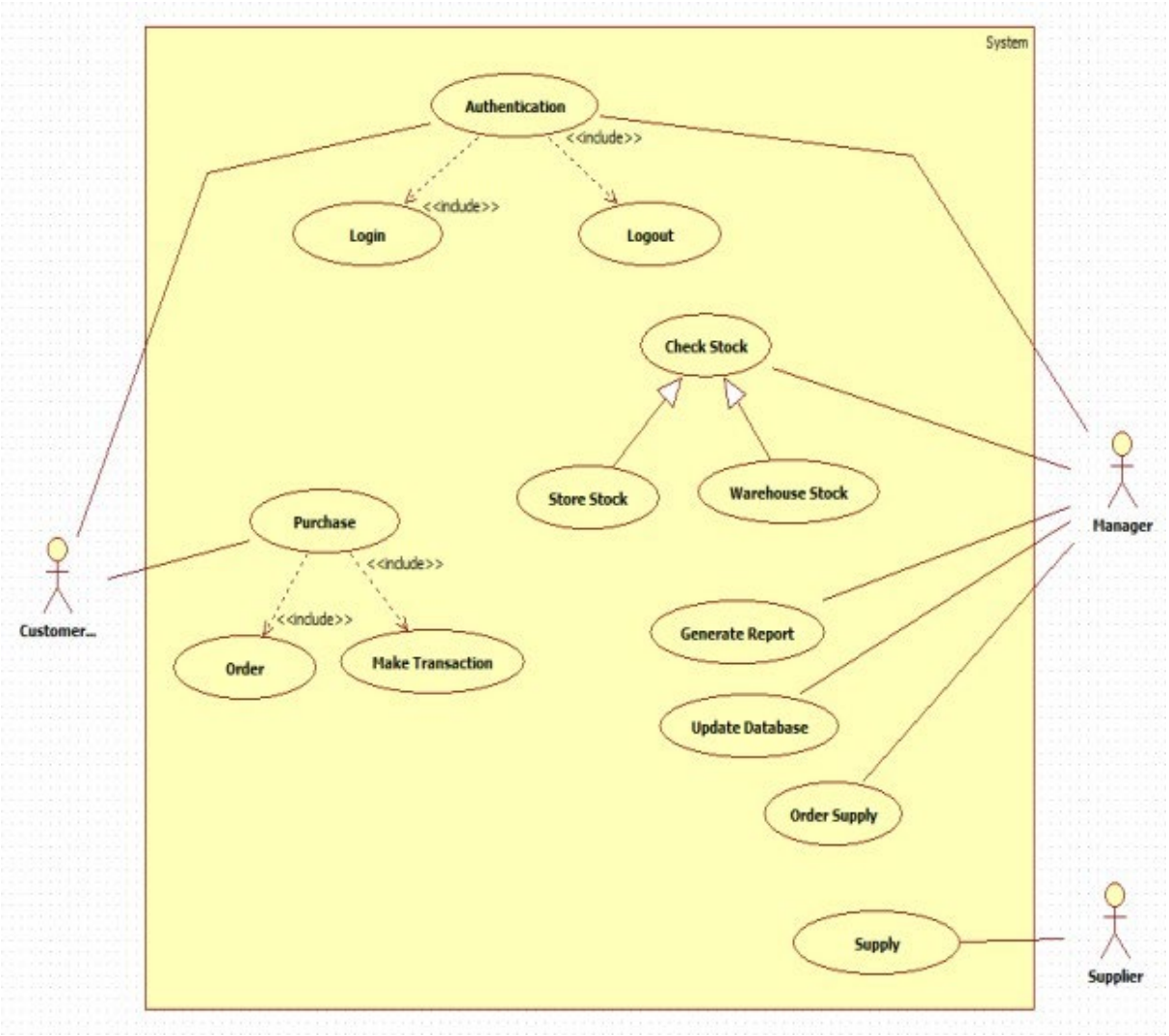

Fig 4.3 Use-case diagram of Stock management system

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:

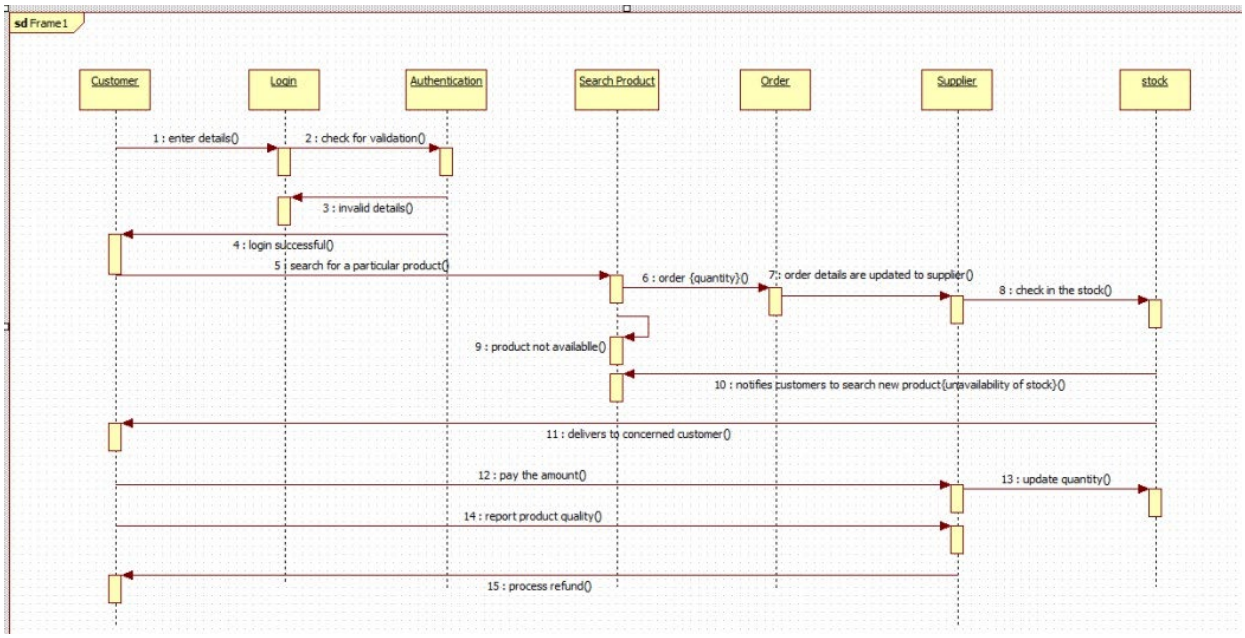

Fig 4.4 Sequence diagram of Stock management system

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:

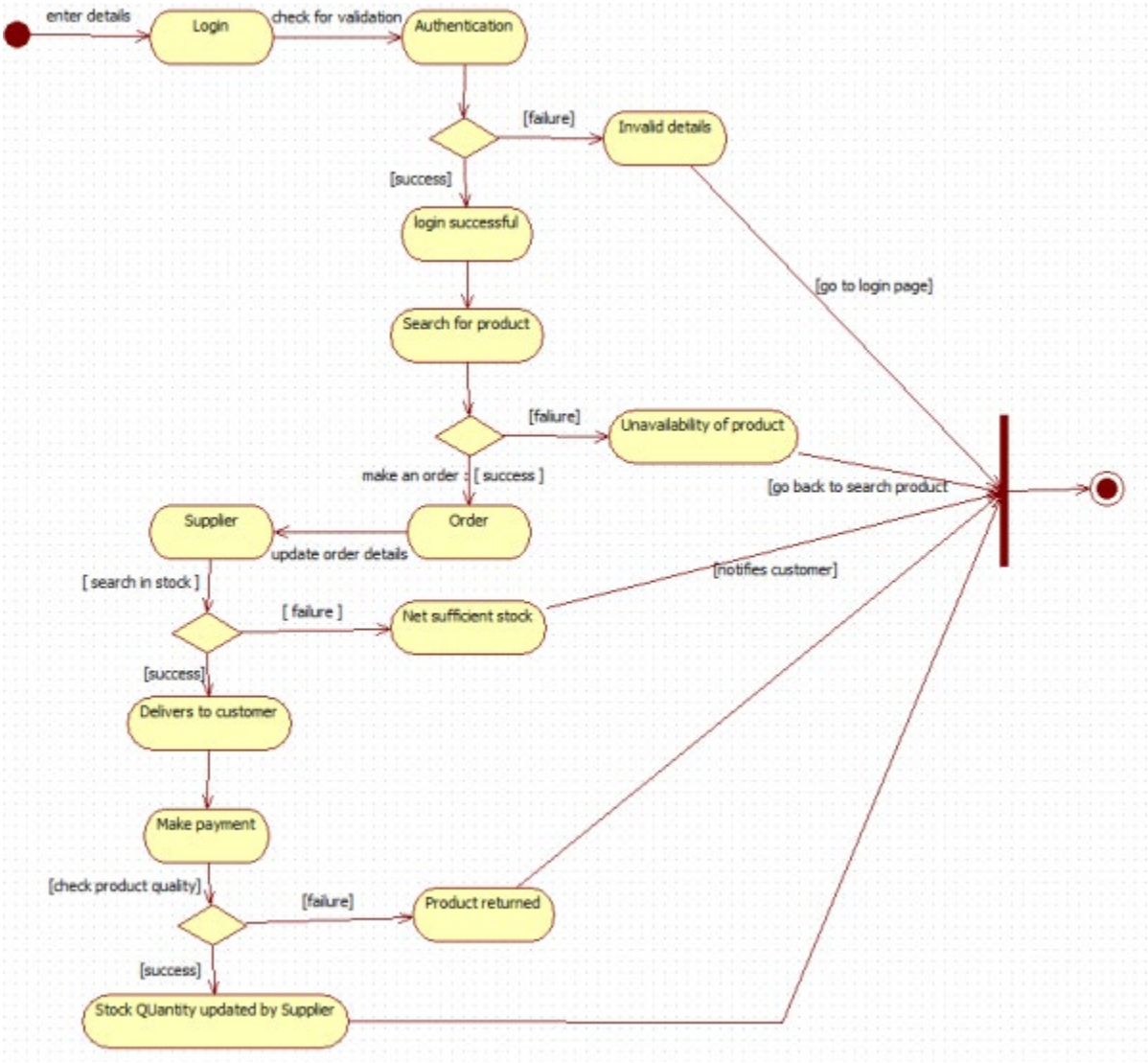

Fig 4.5 Activity diagram of Stock management system

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

System Requirement Document:

Problem statement :

Manual passport application and processing procedures are often time-consuming, prone to human errors, and require applicants to repeatedly visit passport offices for verification and documentation. This results in delays, inefficient document handling, and difficulty in tracking application status. Therefore, a Passport Automation System is required to streamline passport application submission, document verification, appointment scheduling, fee payment, and status tracking. The system should automate workflow steps, reduce manual intervention, improve accuracy, and provide applicants with a transparent and user-friendly experience.

SRS-Software Requirements Specification

1. Introduction

1.1 Purpose

This SRS document defines the requirements for a Passport Automation System that simplifies the passport application process. It ensures a clear understanding among developers and stakeholders regarding system functions such as application submission, appointment booking, verification, payment, and status tracking.

1.2 Scope

The system enables applicants to submit applications online, upload supporting documents, schedule verification appointments, make online fee payments, and track passport status. It also supports government officials in verifying documents, updating application status, and managing approvals.

1.3 Overview

The Passport Automation System improves efficiency by replacing manual paperwork with a centralized digital platform. The system reduces processing time, minimizes human error, and provides greater transparency throughout the application lifecycle.

2. General Description

The system is intended for applicants, passport office staff, and administrators. Applicants can fill forms online, upload documents, pay fees, and check status updates. Officers can verify applications, approve or reject submissions, and schedule processing tasks. The system ensures smooth coordination between different passport processing stages and maintains accurate records for auditing.

3. Functional Requirements

3.1 Application Submission

- Allows applicants to fill in online passport forms.
- Uploads identity proofs, address proofs, and photographs.
- Validates mandatory fields and document formats.

3.2 Appointment Scheduling

- Enables applicants to choose a preferred date and time for verification.
- Automatically assigns passport office counters based on availability.
- Sends SMS/email appointment confirmations.

3.3 Document Verification

- Passport officers review uploaded documents.
- Officers update status as “Approved,” “Pending,” or “Rejected.”
- Supports internal comments for follow-up or clarification.

3.4 Fee Payment

- Supports online payments through UPI, credit/debit cards, and net banking.
- Generates transaction receipts and billing records.

3.5 Status Tracking

- Applicants can track real-time status: Submitted → Under Verification → Police Verification → Printing → Dispatched.
- Sends notifications when status changes.

3.6 Reporting

- Officers and admins can generate reports for daily applications, approvals, rejections, and pending verifications.

4. Interface Requirements

4.1 User Interface

- Simple, accessible interface for applicants with clear navigation for forms and uploads.
- Officer dashboard displaying pending applications, verifications, and status updates.
- Mobile-compatible interface for easy access.

4.2 Integration Interfaces

- Integration with payment gateways for fee processing.
- Integration with police verification system for background checks.
- Integration with national ID databases (Aadhaar, PAN) for identity matching.

5. Performance Requirements

- The system must handle high traffic, especially during seasonal application periods.
- Application form submission should respond within 3 seconds.
- Should support at least 3000 concurrent users.
- Status updates should reflect instantly across modules.

6. Design Constraints

- Must comply with national data privacy and security regulations.
- Should run on government servers and support restricted network environments.
- Requires OCR support for reading uploaded IDs (optional).
- Must use a reliable relational database for storing application data.

7. Non-Functional Attributes

- Security: Encrypted data, secure logins, role-based access.
- Reliability: Continuous operation with backup and recovery options.
- Scalability: Should support growing applicant numbers.
- Usability: User-friendly forms and document upload interface.
- Portability: Web-based, accessible from browsers and mobile devices.
- Reusability: Modular components for future passport services.
- Compatibility: Works across standard browsers and government systems.
- Data Integrity: Ensures accurate and consistent record management.

8. Schedule and Budget

The project development timeline is estimated at 6–7 months including analysis, design, development, testing, and deployment. The budget is approximately $120,000 covering software development, payment integration, database setup, security modules, and maintenance support.

Class diagram:

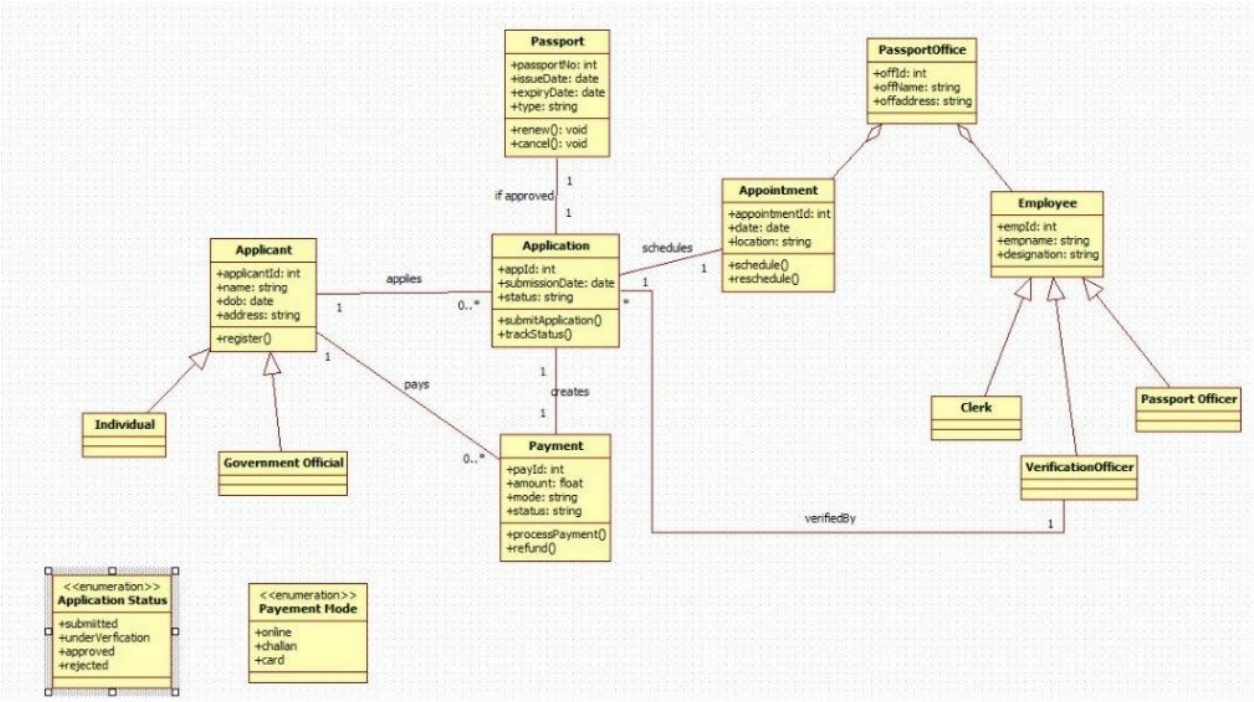

Fig 5.1 Class diagram of Passport authentication system

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:

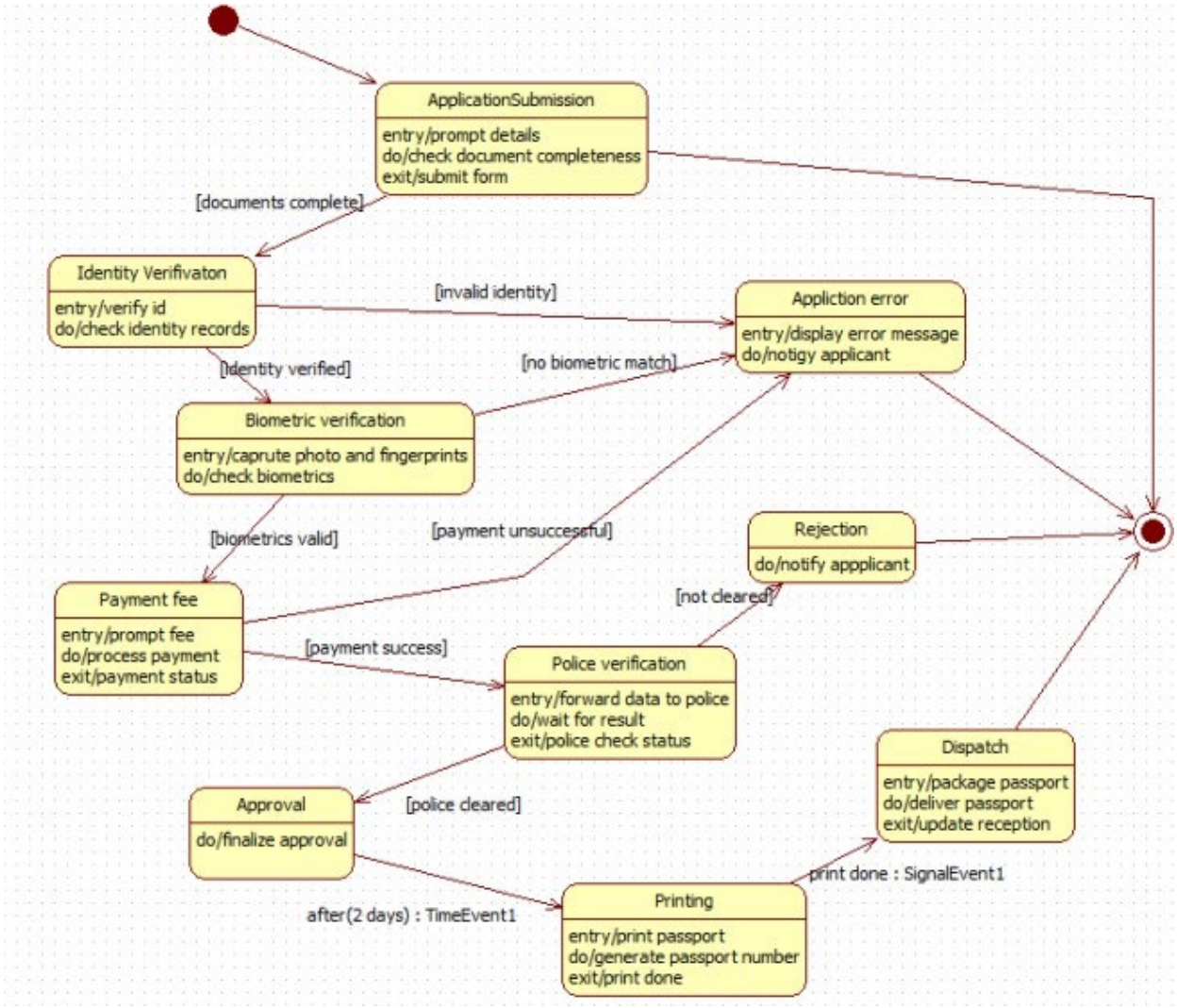

Fig 5.2 State diagram of Passport authentication system

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:

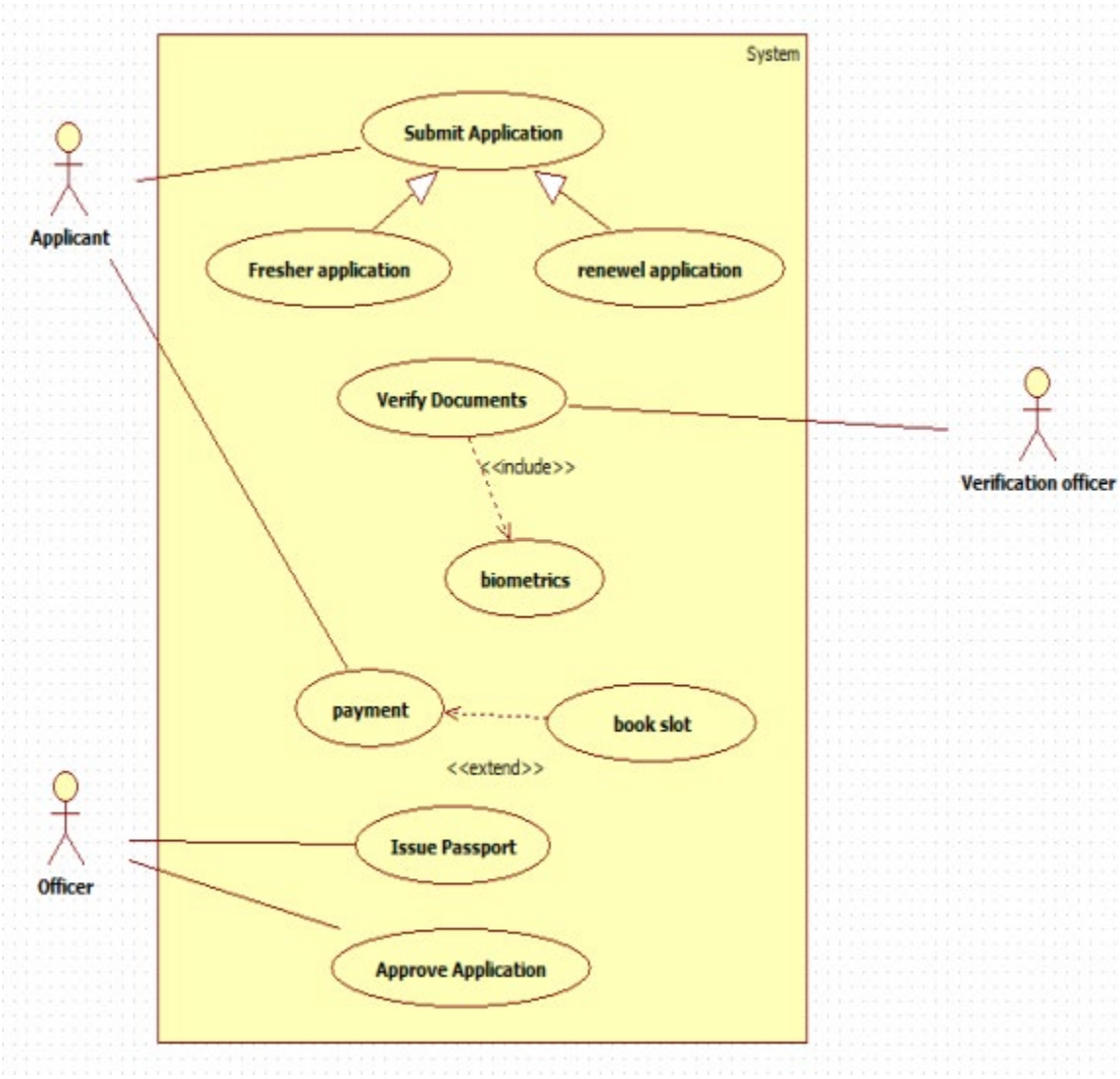

Fig 5.3 Use-case diagram of Passport authentication system

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:

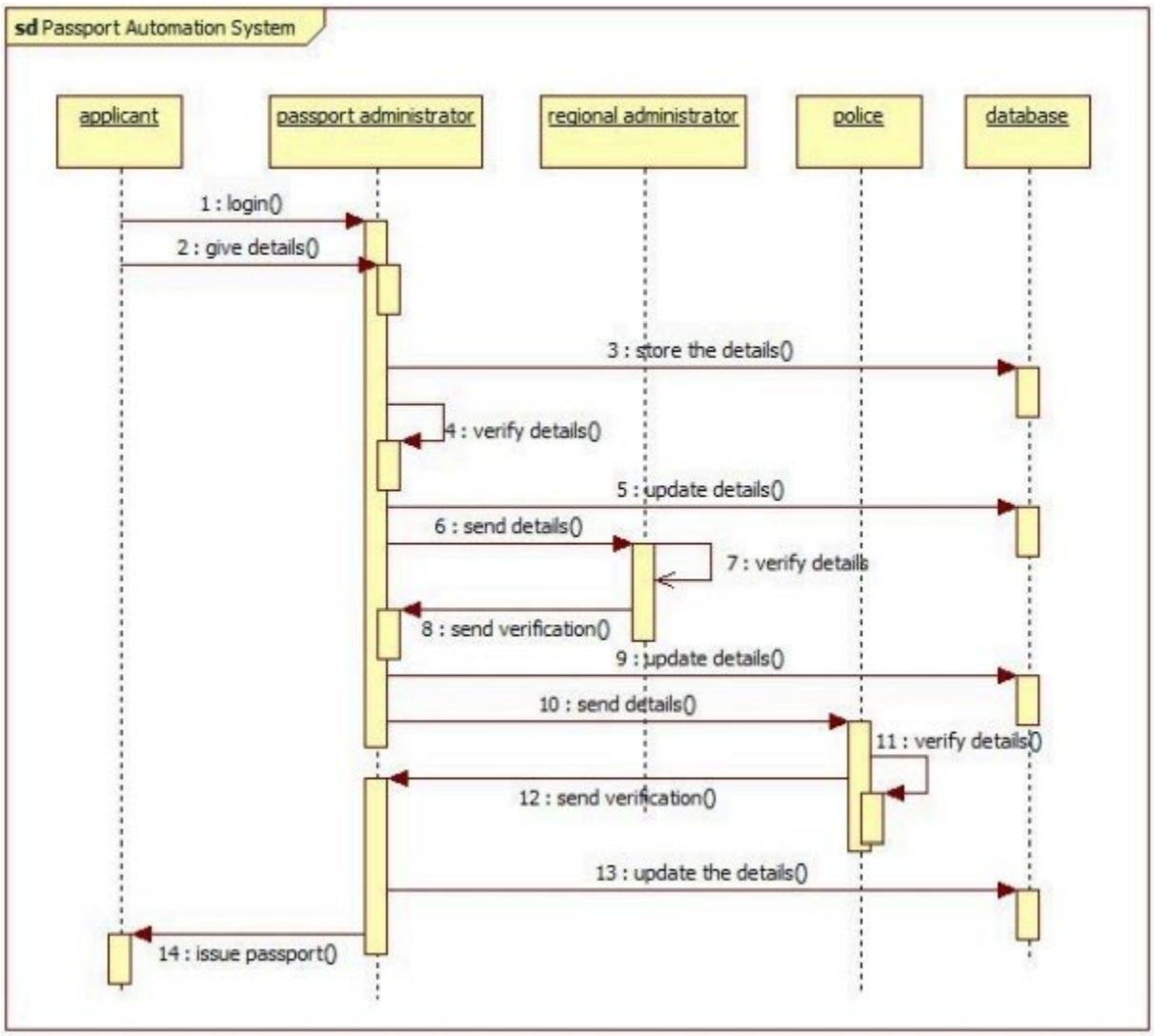

Fig 5.4 Sequence diagram of Passport authentication system

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:

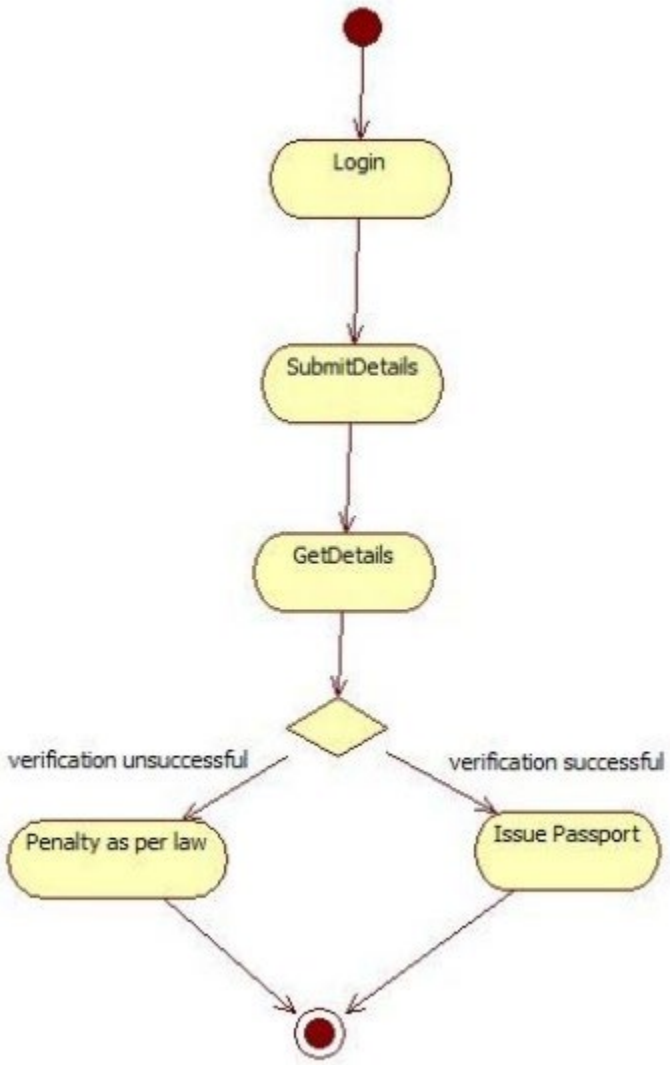

Fig 5.5 Activity diagram of Passport authentication system

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.