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

Lab Record

Object Oriented Analysis and Design

Submitted in partial fulfilment for the 5th Semester Laboratory

Bachelor of Technology
in
Computer Science and Engineering

Submitted by:

B Prabhanjan

1BM23CS060

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
August-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 B Prabhanjan (1BM23CS060) during the 5th Semester August-December 2025

Signature of the Faculty Incharge:

NAME OF THE FACULTY: Vikranth B.M.

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

Table of Contents

1. Hotel Management System

1.1. Problem Statement

The existing operations of the hotel are handled manually, resulting in frequent issues such as booking conflicts, inaccurate billing, delays in check-in/check-out processing, and difficulty in tracking room availability. To address these inefficiencies, the hotel requires a computerized Hotel Management System that can organize and manage room details, guest information, reservations, billing, and occupancy status in a structured manner. The system should enable reception staff to create and manage bookings, perform check-in and check-out operations, update room status automatically, and generate bills for completed stays. The manager should have access to summary reports to monitor overall hotel operations. The objective is to design an object-oriented solution that provides clarity, modularity, and easy maintenance while improving accuracy, speed, and reliability of daily hotel management tasks.

1.2. SRS-Software Requirements Specification

1.2.1. Introduction

- **Purpose:** To define the requirements for a system that automates hotel operations such as room booking, check-in/check-out, billing, and room availability management.
- **Scope:** The system handles reservation creation, customer information, room allocation, billing generation, and report viewing for staff and managers. It ensures accuracy, speed, and improved service for both hotel staff and guests.
- **Overview:** The system streamlines hotel operations, reduces manual errors, provides real-time room status updates, and enhances overall customer satisfaction.

1.2.2. General Description

The system automates hotel reservation and guest management processes, enabling secure and efficient handling of bookings, check-ins, check-outs, payments, and room status updates. Users include customers, hotel staff, and administrators. Benefits include faster service, error reduction, centralized data storage, and improved operational reliability.

1.2.3. Functional Requirements

- Room reservation creation, modification, and cancellation.
- Customer registration and record management.
- Check-in and check-out processing with automatic room status updates.
- Billing calculation based on stay duration and additional services.
- Generation of invoices and reports for management.
- Room availability tracking and conflict prevention.
- Staff and admin login with role-based access.

1.2.4. Interface Requirements

- Hotel staff interface for bookings, check-in/check-out, and billing.
- Customer interface for viewing availability and reservations (if applicable).
- Administrator interface for reports and system configuration.
- Database interface for storing room, guest, and billing data.

1.2.5. Performance Requirements

- Response time: ≤3 seconds for major operations.
- System must handle 100+ daily transactions.
- Uptime: 99% during hotel operating hours.

1.2.6. Design Constraints

- The system must run on Windows/Linux desktop environment.
- Local database storage without mandatory Internet dependency.
- Must follow basic security and data protection guidelines.

1.2.7. Non-Functional Attributes

- Security: Role-based access control and secure storage of customer information.
- Reliability: Stable operation without failures during working hours.
- Scalability: Ability to add more rooms or features in future versions.
- Usability: Simple, clean interface requiring minimal training for staff.

1.2.8. Preliminary Schedule & Budget

- Development duration: 6–8 months.
- Budget: ~Rs.1,50,000 - Rs.2,25,000.
- Phases: Requirements → Design → Development → Testing → Deployment.

1.3. Class Diagram

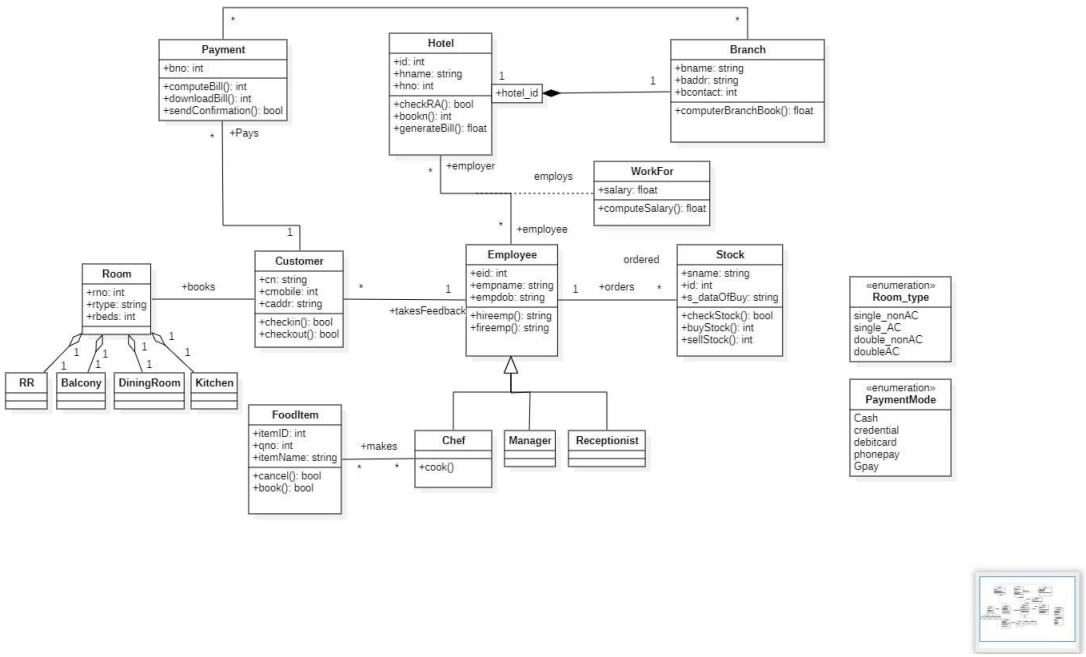


Fig 1.1 Class Diagram for Hotel Management System

Classes and Their Responsibilities

- Hotel
  - Represents the main hotel entity with name, location, and contact details.
  - Acts as the top-level class managing overall operations.
  - Maintains a list of branches and hotel-wide resources.
- Branch
  - Represents an individual unit of the hotel.
  - Contains its own rooms, employees, and services.
  - Stores branch-specific information like branch ID, address, and phone number.
  - Performs operations such as handling bookings, staff, and service reports.
- Customer
  - Represents hotel guests who avail services.
  - Maintains personal details and booking records.
  - Performs actions such as booking a room, requesting services, and making payments.
- Employee (Base Class)
  - Represents all staff members with common attributes like employee ID, name, and salary.
  - Acts as the parent class for different staff roles.
  - Provides a structure for maintaining staff hierarchy and shared functionality.

- Manager (Extends Employee)
    - Handles administrative duties such as authorizing bookings.
    - Manages staff, resources, and branch operations.
    - Accesses branch reports and overall performance data.

- Receptionist (Extends Employee)
    - Handles customer interactions at the front desk.
    - Manages check-in, check-out, cancellations, and room allocation.

- Chef (Extends Employee)
    - Responsible for food preparation and kitchen management.
    - Coordinates with stock and menu-related operations.

- Room
    - Represents individual rooms in a branch.
    - Stores information such as room number, type, price, and availability.
    - Tracks booking status and maintenance requirements.

- Payment
    - Represents financial transactions linked to bookings.
    - Stores details such as amount, payment ID, and payment method.
    - Generates receipts and confirms payment status.

- Stock
    - Represents inventory items used in hotel operations, especially the kitchen.
    - Tracks available quantities and triggers restocking when necessary.

- FoodItems
    - Represents menu items available for customers.
    - Linked with stock and chef operations to prepare dishes.

Relationships and Interactions

- Hotel to Branch
    - A hotel contains multiple branches.
    - The relationship shows aggregation (whole–part relationship).
- Branch to Employee
    - Each branch employs multiple staff members.
    - Employees work only under one assigned branch.

- Employee Hierarchy
    - Manager, Receptionist, and Chef inherit from the Employee class.
    - Ensures modularity and specialization without code duplication.
- Customer to Room
    - A customer can book one or more rooms.
    - Rooms can be booked by different customers over time.
    - Relationship supports booking operations and reservation history.
- Customer to Payment
    - Customers make payments related to their bookings.
    - Each payment is tied to a specific booking transaction.
- Chef to FoodItems and Stock
    - Chef prepares dishes using food items.

1.4. State Diagram

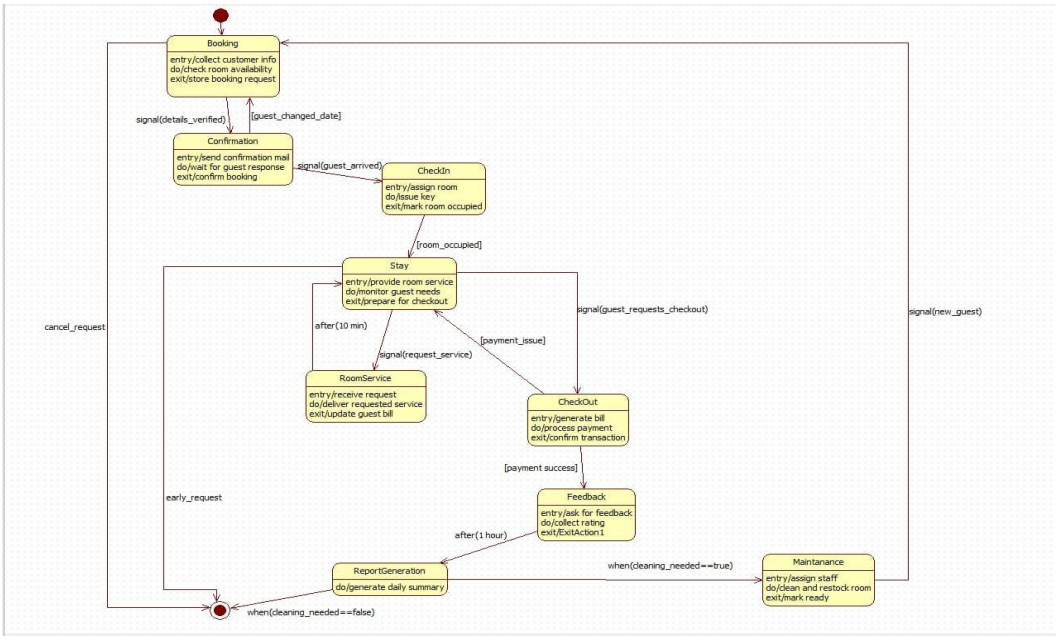
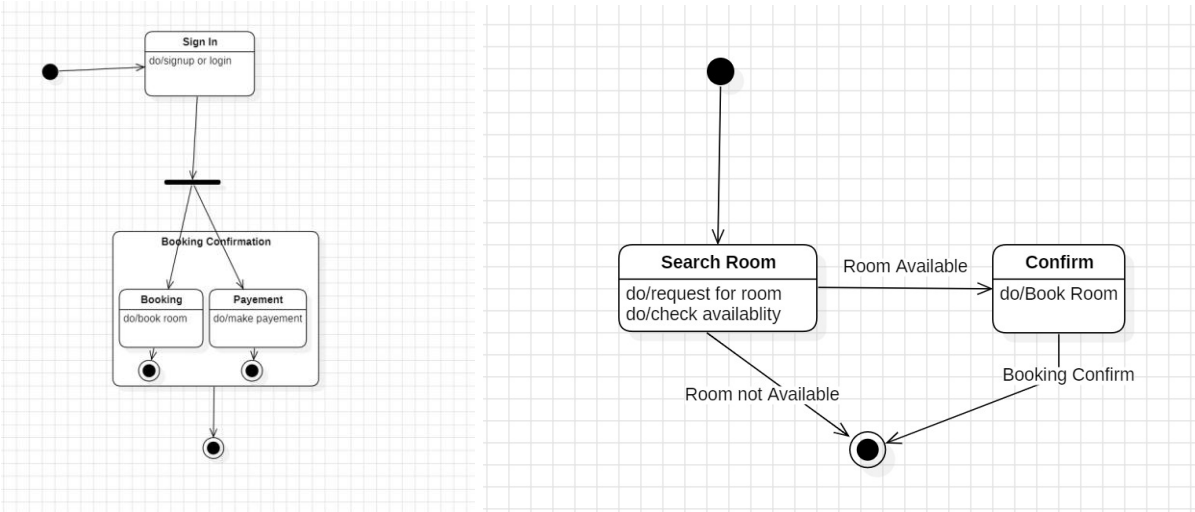


Fig 1.2 State Diagram for Hotel Management System(1)

This state machine diagram represents the dynamic flow of the hotel booking lifecycle from the moment a customer initiates a booking until checkout and post-stay feedback. The process begins at the Booking state where customer details are collected, followed by Confirmation, Check-in, Stay, and Room Service states. Transitions occur based on events such as guest arrival, payment success, or maintenance requirements. The diagram also includes system-driven states like Report Generation and Maintenance to ensure operational efficiency. Overall, it highlights how the booking system responds to different events and moves through various stages of a guest's hotel experience.

Fig 1.3 State Diagram for Hotel Management System(2)



The room search diagram shows a simple decision-based flow for checking room availability. The process starts with the Search Room state, where the system processes the customer's request and checks availability. If a room is available, the flow transitions to the Confirm state where booking can be completed. If no rooms are available, the process terminates at the final state. This diagram provides a clear depiction of the basic decision-making logic involved in room searching and reservation.

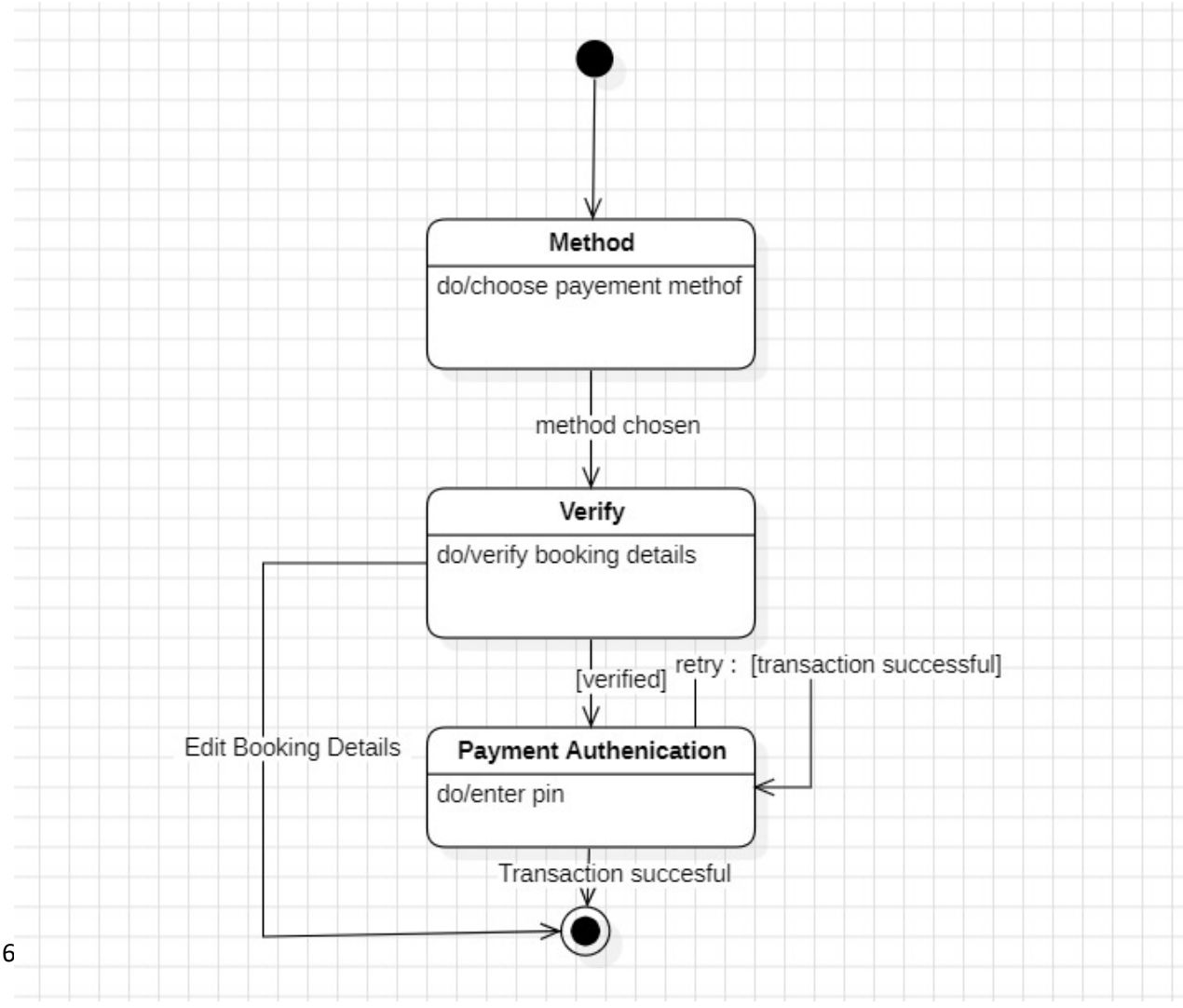


Fig 1.4 State Diagram for Hotel Management System(3)

The payment state machine diagram outlines the sequential steps involved in completing a hotel payment transaction. It begins with the Method state, where the customer chooses a payment method. It then moves to the Verify state, which checks booking and payment details. If everything is valid, the flow proceeds to Payment Authentication, where the user provides security credentials such as a PIN or OTP. Upon successful authentication, the process ends at the final state. If verification fails, the user may edit booking details or retry the transaction. This diagram effectively demonstrates the dynamic process and decision paths in payment handling.

1.5. Use Case Diagram

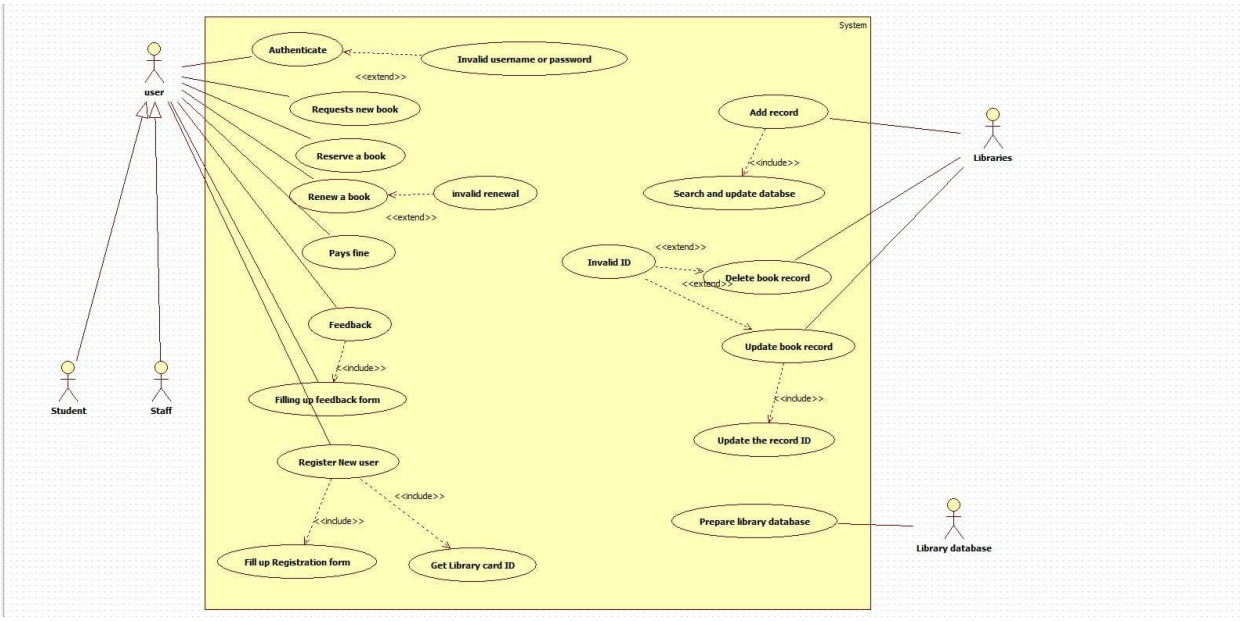
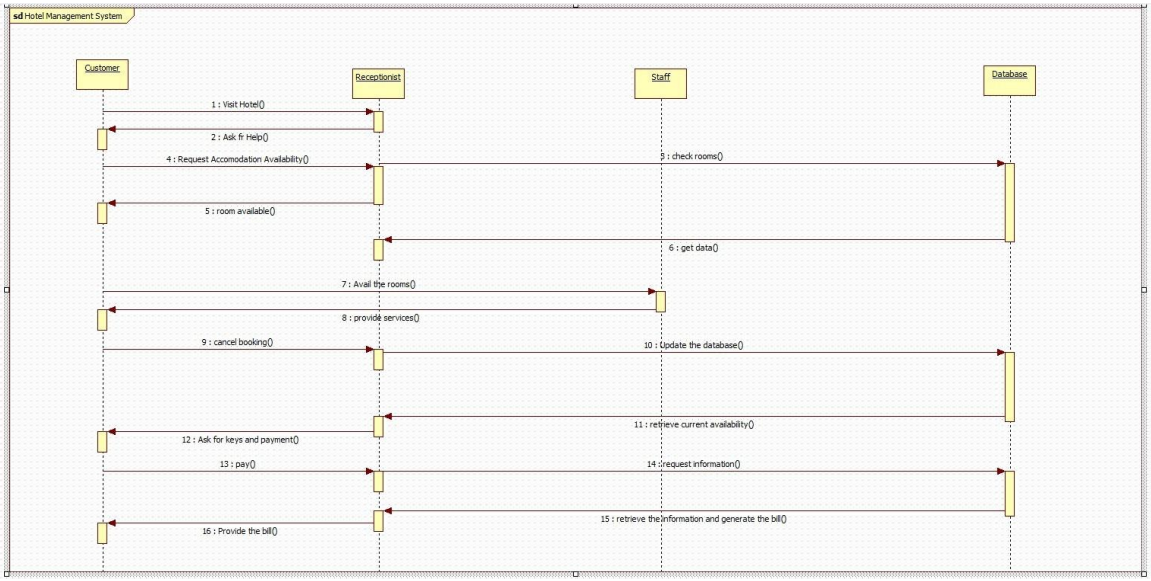


Fig 1.5 Use Case Diagram for Hotel Management System

The use case diagram represents the functional interactions between different users and the library management system. The main actors—Students, Staff, Librarians, and the Library Database—engage with the system through activities such as authentication, searching for books, reserving books, paying fines, giving feedback, and managing book records. Librarians are responsible for adding, updating, and deleting records, while the system includes exception flows like invalid credentials or invalid IDs. Overall, the diagram captures the complete set of services provided by the library system and the roles played by different users.

1.6. Sequence Diagram

Fig 1.6 Sequence Diagram for Hotel Management System



The diagram shows interactions between the Customer, Receptionist, Staff, and Database. The sequence begins when the customer asks the receptionist to check room availability. The receptionist contacts the staff, who query the database for real-time room information. It then outlines different outcomes: Successful booking,Cancellation. Finally, the payment process is shown, where the customer requests the bill, and the receptionist retrieves billing details from the database.

1.7. Activity Diagram

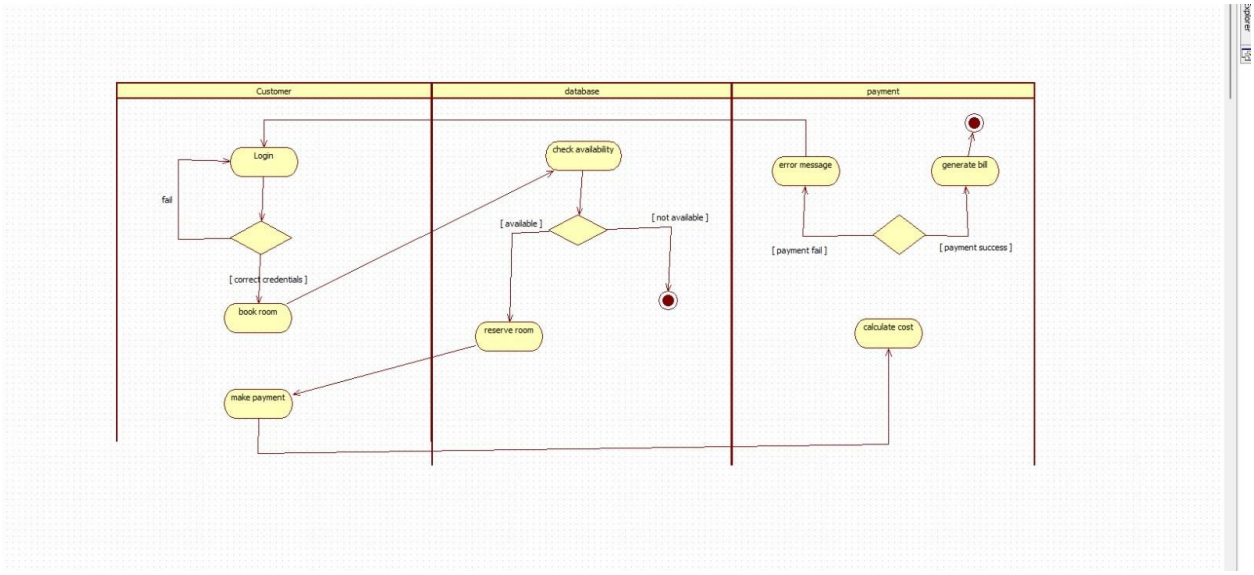


Fig 1.7 Activity Diagram for Hotel Management System

The diagram is divided into three sections, called swimlanes, representing the main actors or systems involved: the Customer, the Database, and the Payment system. The process begins with the customer's attempt to log in. If the login is successful, the customer can proceed to book a room. This action prompts the system to check the database for room availability. If a room is

available, it is reserved for the customer, who is then directed to make a payment. If no room is available, the process ends. The payment step involves calculating the total cost. If the payment transaction is successful, a bill is generated, and the process is complete. However, if the payment fails, an error message is displayed, and the process terminates.

2. Credit Card Processing

2.1. Problem Statement

Manual and fragmented methods of handling credit card transactions are prone to delays, errors, and security risks. A secure Credit Card Transaction Processing System is needed to automate the entire lifecycle from initiation and authorization to settlement and billing ensuring fast, accurate, and reliable payment processing. This system will improve efficiency, reduce fraud, and provide a seamless experience for businesses, banks, and customers.

2.2. SRS - Software Requirement Specification

2.2.1. Introduction

- Purpose: To define requirements for a system that securely processes credit card transactions, from initiation to settlement and billing.
- Scope: The system captures card details, requests authorization, processes approvals/ declines, settles funds, and updates cardholder billing. It ensures speed, security, and accuracy for merchants, banks, and customers.
- Overview: The system provides real-time transaction processing, reduces errors, and improves customer satisfaction.

2.2.2. General Description

The system automates credit card transaction handling, ensuring secure communication between merchants, processors, card networks, and banks. Users include merchants, cardholders, and financial institutions. Benefits: faster approvals, reduced fraud, and reliable settlements.

2.2.3. Functional Requirements

- Capture card details (POS/online).
- Securely transmit transaction data.
- Request authorization from card networks and issuing banks.
- Approve/decline based on funds and fraud checks.
- Batch and settle approved transactions.
- Transfer funds to merchant accounts.
- Update cardholder billing and statements.

2.24 Interface Requirements

- POS/Payment gateway → Processor.
- Processor → Card networks & banks.
- Merchant dashboard for reports.

5. Performance Requirements

- Response time: ≤4 seconds.
- Handle 10,000+ TPS (transactions per second).
- Uptime: 99.9%.

6. Design Constraints

- PCI DSS compliance.
- Strong encryption (TLS, tokenization).
- Must support multiple currencies.

7. Non-Functional Attributes

- Security: Fraud detection & data protection.
- Reliability: Redundant & fault-tolerant.
- Scalability: Millions of daily transactions.
- Usability: Simple merchant dashboard.

8. Preliminary Schedule & Budget

- Duration: 6–8 months.
- Budget: ~Rs.1,50,000 - Rs.2,25,000
- Phases: Requirements → Development → Testing → Deployment.

2.3. Class Diagram

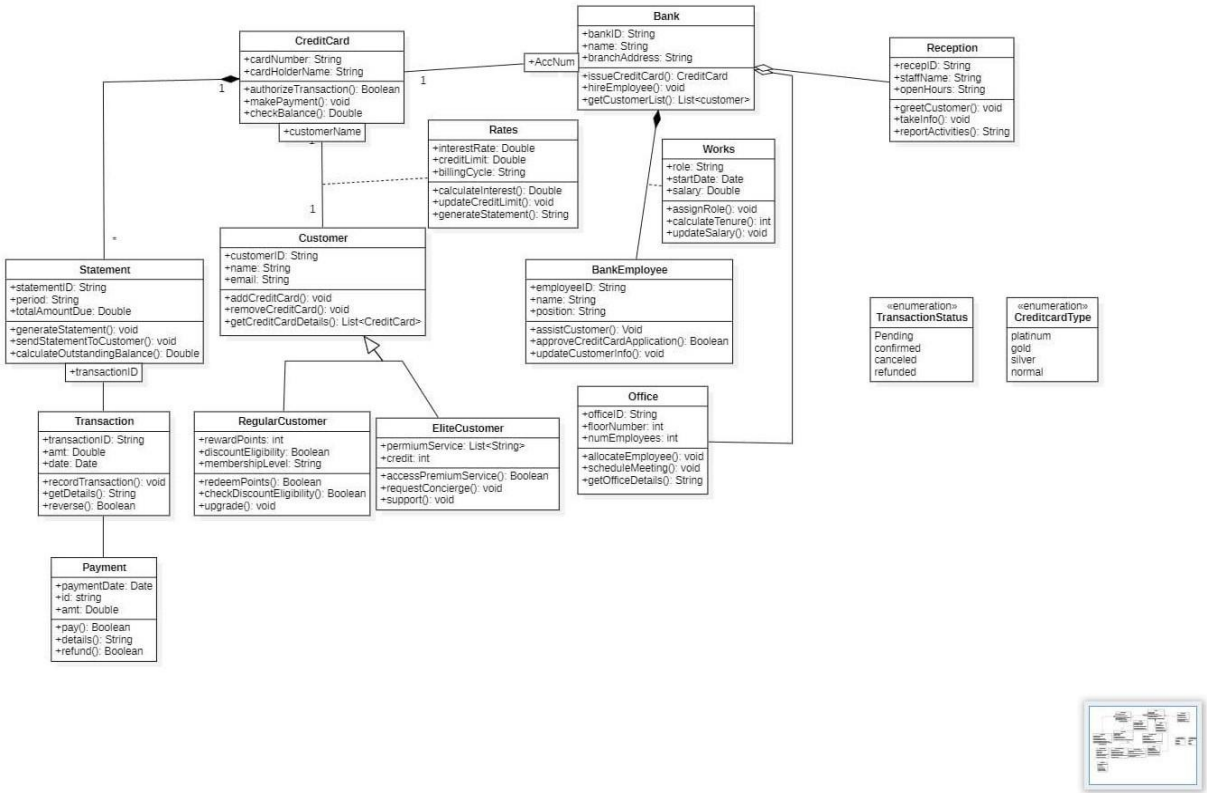


Fig 2.3 Class Diagram for Credit Card Processing

This is a Class Diagram that outlines the static structure of a banking system, focusing on credit card services. It defines various classes such as Bank, Customer, CreditCard, BankEmployee, and Transaction. The Bank class has attributes like name and address and can perform actions such as issuing credit cards and managing employees. The Customer class, which has a relationship with the Bank, can have one or more CreditCards. Customers are further specialized into RegularCustomer and EliteCustomer, each with unique attributes and behaviors. The CreditCard class contains details like card number and cardholder name and is associated with Statement and Transaction classes. This diagram effectively maps out the entities within the system, their properties, and how they are related to one another.

2.4. State Diagram

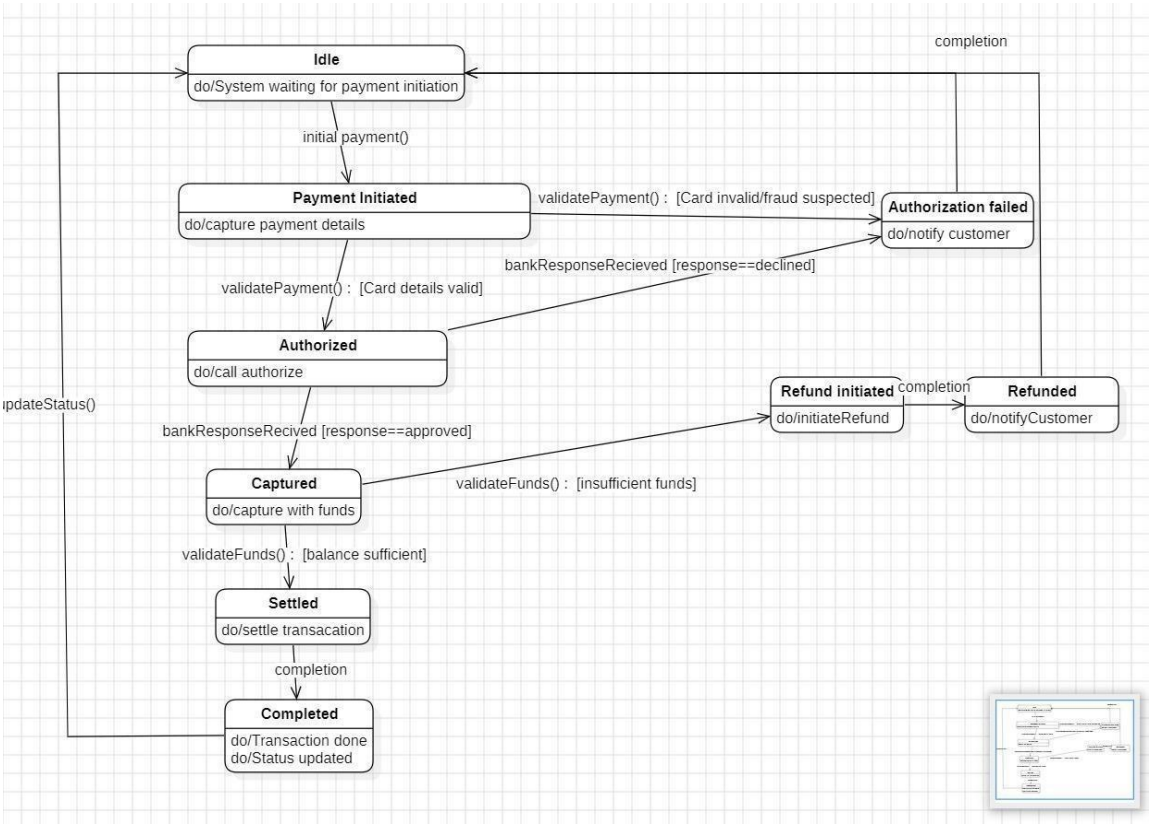


Fig 2.2 State Diagram for Credit Card Processing(1)

This diagram is a State Machine Diagram that illustrates the lifecycle of a payment transaction. The process begins in the Idle state, waiting for a payment to be initiated. Upon initiation, it transitions to the Payment Initiated state, where payment details are captured. The system then attempts to validate the payment. If the card is invalid or fraud is suspected, the state changes to Authorization failed, and the customer is notified. If the card details are valid, it moves to the Authorized state. Based on the bank's response, the transaction is either Captured (if approved) or moves to Authorization failed (if declined). From the Captured state, if funds are sufficient, the transaction becomes Settled and then Completed. If funds are insufficient, it can transition to

a failure state. The diagram also shows a path for refunds, moving from a completed or settled state to Refund initiated and finally Refunded.

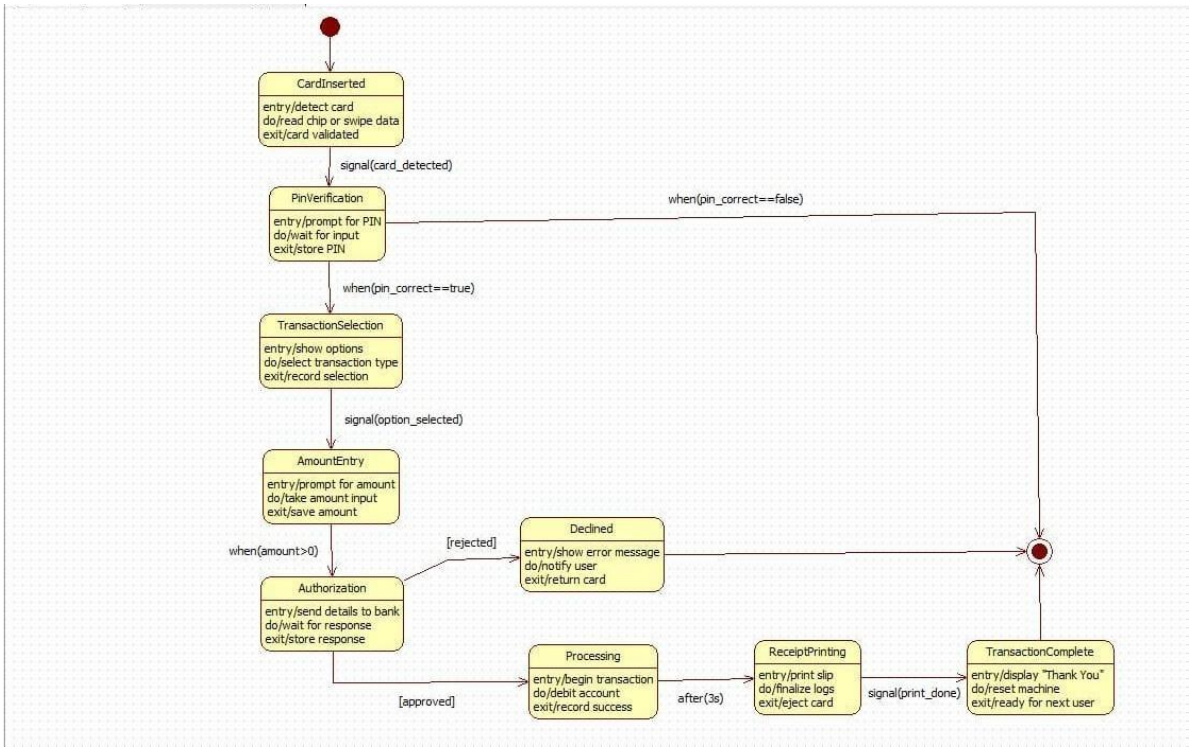


Fig 2.3 State Diagram for Credit Card Processing(2)

This State Machine Diagram details the process of an ATM or Point-of-Sale (POS) transaction. The flow starts when a CardInserted is detected. The system then transitions to PinVerification, prompting the user for a PIN. If the PIN is correct, it moves to TransactionSelection, allowing the user to choose an action like withdrawal. After the user enters an AmountEntry, the system proceeds to Authorization by sending the details to the bank. If the transaction is approved, it goes into Processing, followed by ReceiptPrinting, and finally ends in the TransactionComplete state. If the authorization is rejected at any point (e.g., incorrect PIN or bank denial), the state shifts to Declined, an error message is shown, and the process terminates.

2.5. Use Case Diagram

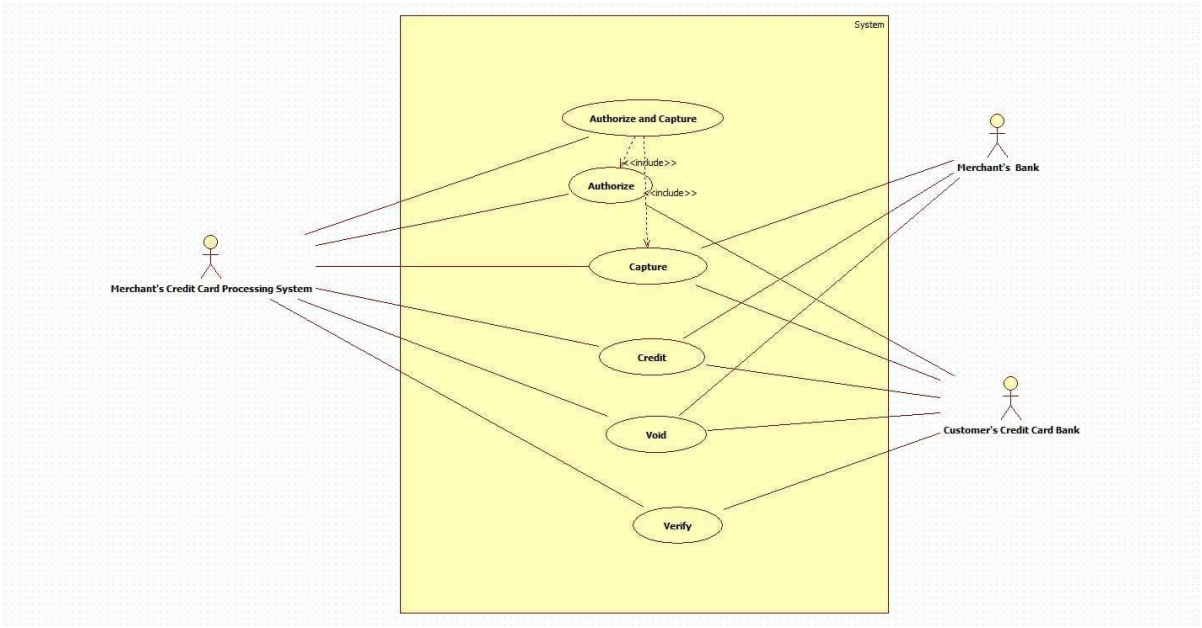


Fig 2.4 Use Case Diagram for Credit Card Processing

This is a Use Case Diagram for a Merchant's Credit Card Processing System. It identifies the primary actors interacting with the system: the Merchant's Credit Card Processing System itself, the Merchant's Bank, and the Customer's Credit Card Bank. The central system provides several key functions, or use cases, including Authorize, Capture, Credit (for refunds), Void (to cancel transactions), and Verify. The Authorize and Capture use cases are combined into a larger Authorize and Capture use case, indicating that a payment is typically authorized first and then captured. The diagram shows which actors interact with each use case; for example, both the merchant's system and the customer's bank are involved in authorization and verification processes.

2.6. Sequence Diagram

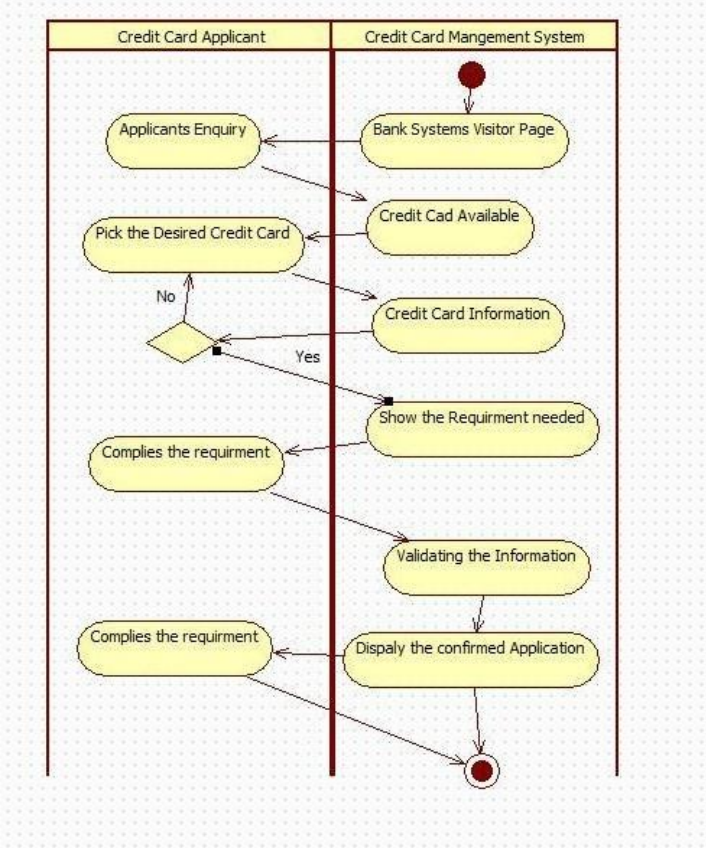


Fig 2.5 Sequence Diagram for Credit Card Processing

This Sequence Diagram illustrates the interactions between multiple components during an online credit card transaction. The participants include the Card Holder, Merchant, App Payment (payment gateway), Bank Office, Database, and Payment Provider. The sequence starts when the Card Holder makes a Purchase from the Merchant. The Merchant requests payment from the App Payment system, which returns a payment URL to the cardholder. The cardholder submits their information, which the App Payment system processes by masking the Primary Account Number (PAN) and verifying it with the Bank Office and Database. Once the information is verified, the App Payment system sends the credit card information to the Payment Provider for processing. The final payment result is relayed back through the App Payment system to both the Merchant and the Card Holder, completing the transaction sequence.

2.7. Activity Diagram

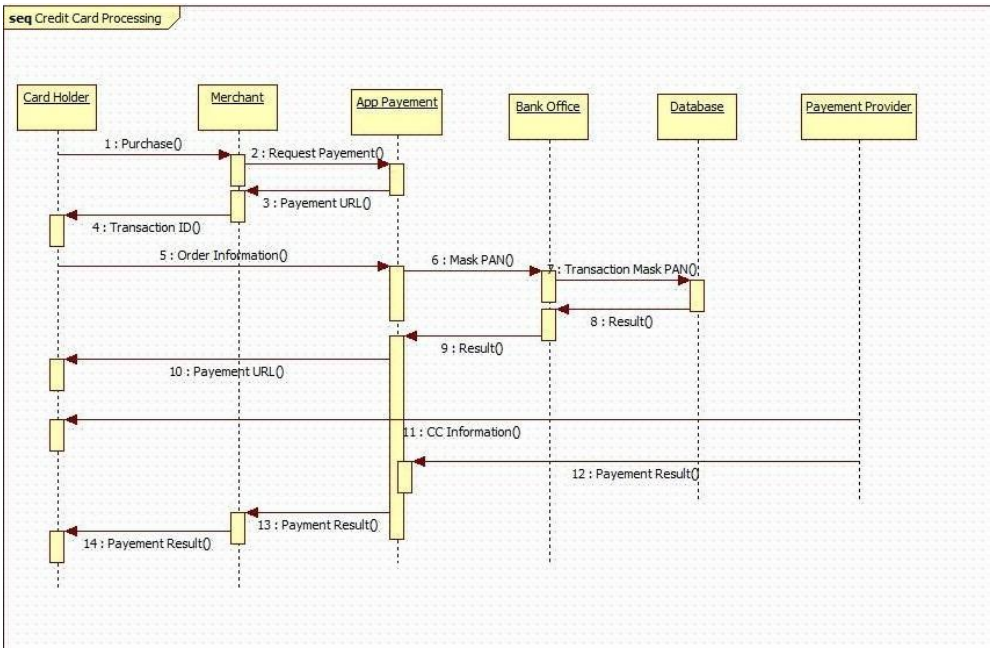


Fig 2.6 Activity Diagram for Credit Card Processing

This is an Activity Diagram that describes the workflow for applying for a credit card. It uses two swimlanes to distinguish between the actions of the Credit Card Applicant and the Credit Card Management System. The process begins with the applicant making an enquiry on the bank's system. The system shows available credit cards, and the applicant picks one. The system then displays the requirements. A decision point follows: if the applicant does not meet the requirements, they may need to pick a different card. If they do, they proceed to comply with the requirements by submitting information. The system then validates this information and displays the confirmed application to the applicant for final review before the process concludes.

3. Library Management System

3.1. Problem Statement

Traditional library operations that rely on manual record keeping are time-consuming, error-prone, and inefficient in managing books, users, and circulation. A Library Management System (LMS) is needed to automate cataloging, circulation, and reporting processes, ensuring accurate record maintenance, faster transactions, and better accessibility of resources. This system will streamline library operations and enhance user experience for both staff and members

3.2. SRS-Software Requirements Specification

1. Introduction

- Purpose:The purpose of this document is to specify the requirements for a Library Management System (LMS). The system will automate cataloging, circulation, and user management while providing efficient access to books and reports.
- Scope:The LMS will allow librarians to manage books, track borrow/return transactions, register users, and generate reports. Users can search for books, view availability, and issue/return books. The system will ensure accuracy, reduce manual workload, and improve user satisfaction.
- Overview:The LMS integrates book management, circulation, and reporting into one computerized platform. It improves efficiency, minimizes errors, and provides easy access to resources.

2. General Description

The LMS automates core library functions, replacing manual logs with a centralized database. It benefits librarians by simplifying management tasks and helps users by making book searches and borrowing more efficient. Key features include cataloging, circulation, reporting, and user account management.

3. Functional Requirements

- Add, update, or delete book records.
- Register and manage library users.
- Issue and return books with due-date tracking.
- Search for books by title, author, or category.
- Track overdue books and generate fines.
- Generate daily/weekly/monthly reports.

4. Interface Requirements

- User Interface: Dashboard for librarians and students.
- Database Interface: Centralized storage for books, users, and transactions.
- Report Interface: Printable/exportable reports in PDF/Excel.

5. Performance Requirements

- Search results within 2 seconds.
- Support at least 1,000 concurrent users.
- Maintain error-free transaction logs.

3.5.6 Design Constraints

- The system must be compatible with Windows/Linux.
- Must use a relational DBMS (MySQL/PostgreSQL).
- User authentication required for access.

3.2.7 Non-Functional Attributes

- Security: Role-based access for librarians and members.
- Reliability: Daily backup of data.
- Usability: Simple UI for non-technical users.
- Scalability: Ability to support more branches in the future.

8. Preliminary Schedule & Budget

- Development Duration: 3–4 months.
- Budget Estimate: Rs.75,000–Rs.1,00,000.
- Phases: Requirement Analysis → Development → Testing → Deployment → Training.

2.3 Class Diagram

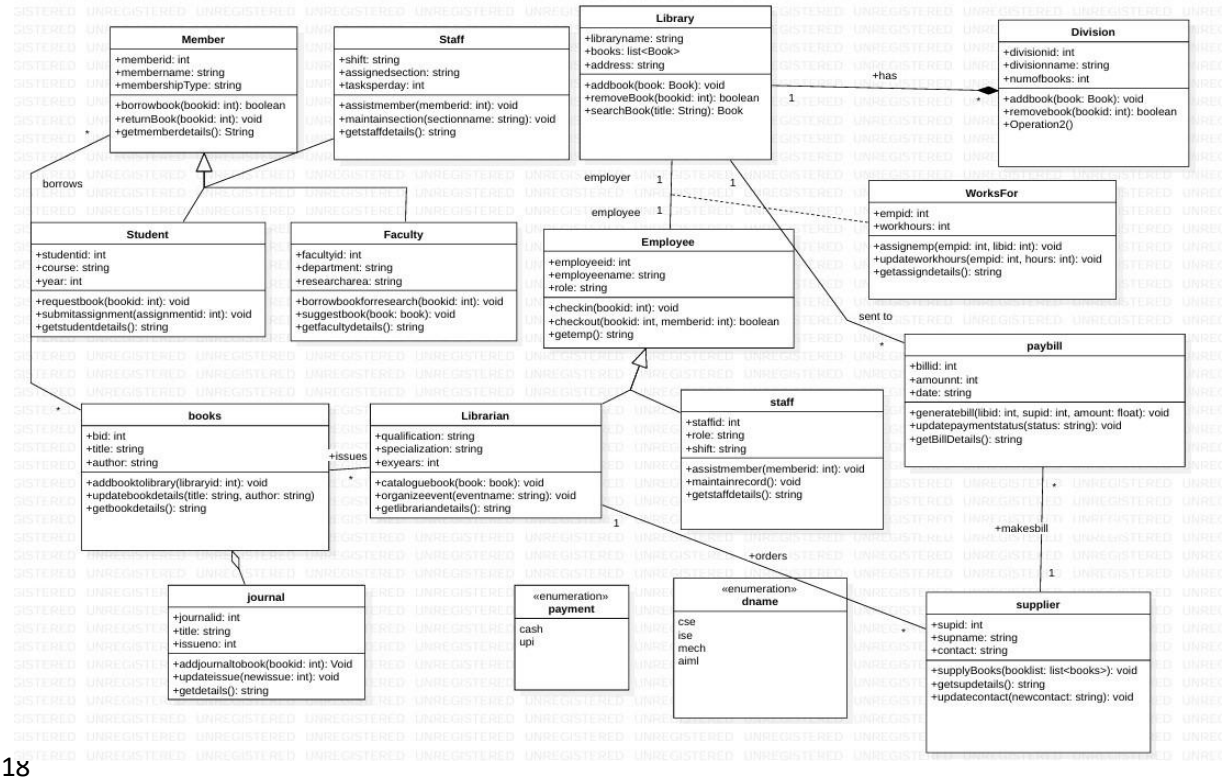


Fig 3.1 Class Diagram for Library Management System

This is a comprehensive Class Diagram representing the structure of a Library Management System. It details the different entities (classes) within the system, their attributes (data), and their methods (behaviors). Key classes include Library, Member, Book, Staff, and Supplier. The diagram shows relationships between these classes, such as inheritance, where Student and Faculty are types of Member. It also illustrates associations, for example, a Member borrows a Book, and a Librarian (a type of Staff) issues a Book. This diagram serves as a blueprint for the system, defining how different components are organized and interact with each other at a structural level.

4. State Diagram

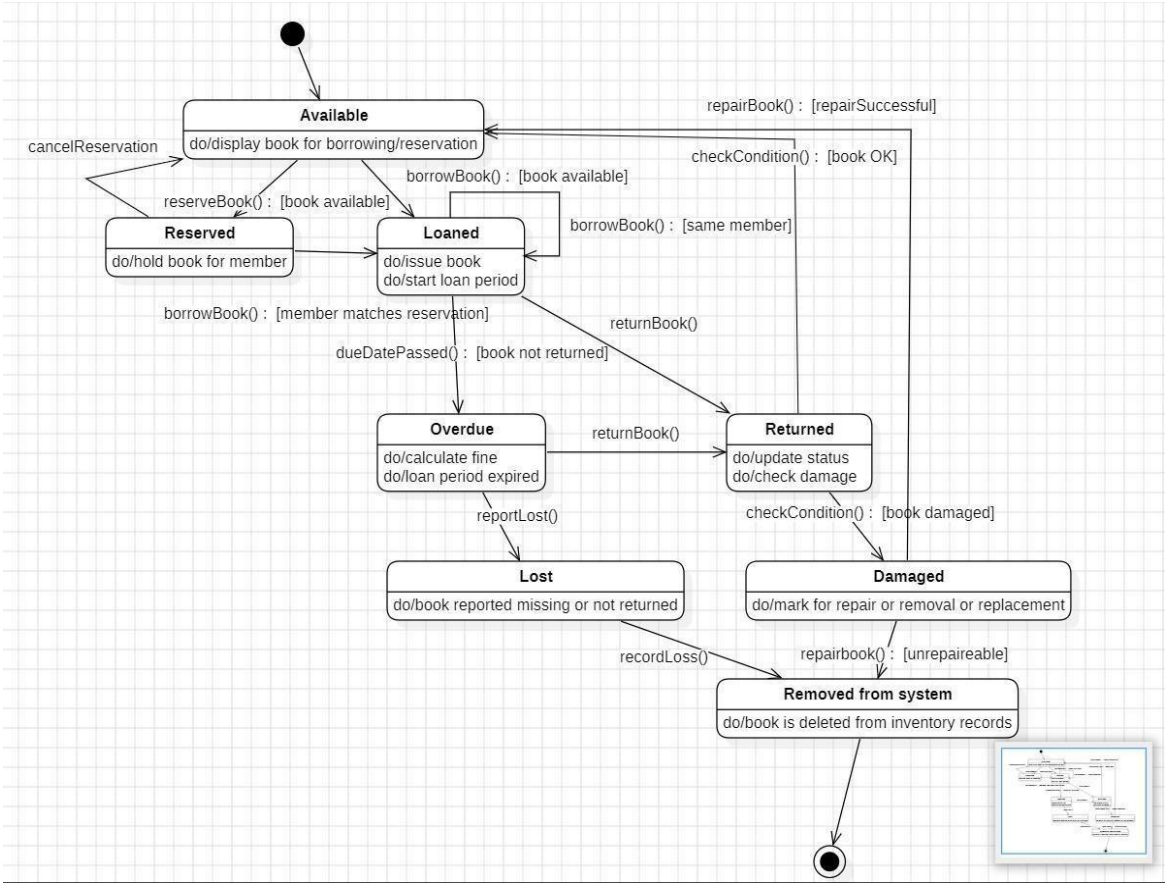


Fig 3.2 State Diagram for Library Management System

This State Machine Diagram illustrates the complete lifecycle of a book within the library system. It tracks the various states a book can be in, starting from Available on the shelf. From there, a book can be Reserved or directly Loaned. If a loaned book is not returned on time, it becomes Overdue. When a book is returned, its state changes to Returned, and its condition is checked. Depending on the condition, it either goes back to Available or is marked as Damaged. The diagram also accounts for situations where a book is Lost or damaged beyond repair, in which case it is Removed from system, marking the end of its lifecycle.

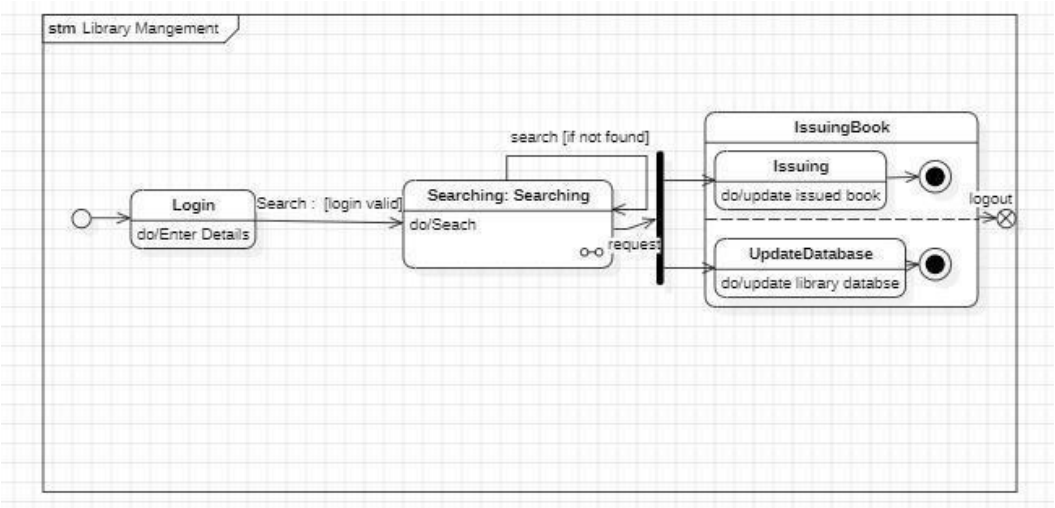


Fig 3.3 Advance State Diagram for Library Management System (1)

This diagram is a high-level State Machine Diagram for the Library Management system, focusing on a user's session. The process begins with a Login state. Upon valid authentication, the system transitions to a Searching state, where the user can look for books. From the searching state, the user can make a request, which moves the system into the IssuingBook composite state. This state includes the internal processes of Issuing the book and UpdateDatabase. After the issuing process is complete, or if the user chooses to, the system transitions to a final logout state, ending the session.

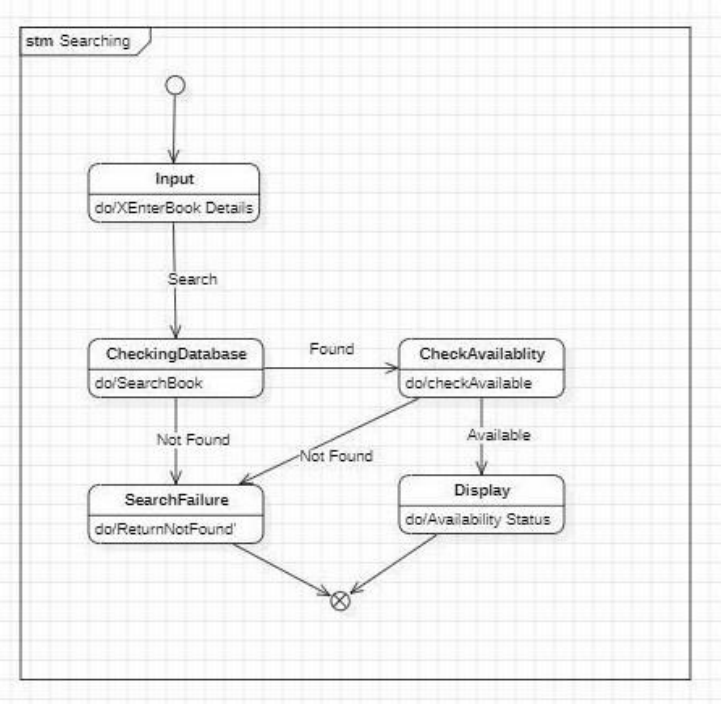


Fig 3.4 Advance State Diagram for Library Management System (2)

This is a detailed State Machine Diagram that focuses specifically on the "Searching" process within the library system. The workflow starts in the Input state, where the user enters the details of the book they are looking for. The system then transitions to CheckingDatabase to search its records. If the book is found, the system moves to CheckAvailability. If it's not found in the database, the state becomes SearchFailure. From the availability check, if the book is available, its status is shown in the Display state. If it is not available, this also leads to a failure or status update. The diagram shows how the system handles a search query from input to final output.

5. Use Case Diagram

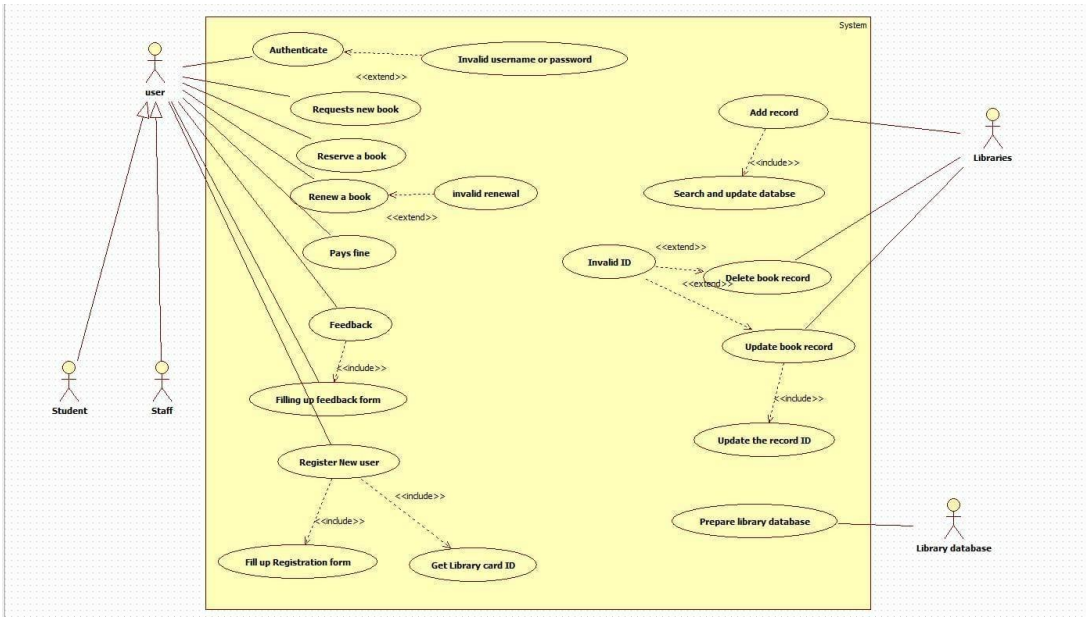


Fig 3.5 Use Case Diagram for Library Management System

This is a Use Case Diagram that provides a functional overview of the library system from the perspective of its users (actors). The primary actors are the general user (which can be a Student or Staff), the Librarian, and the Library database. The diagram outlines all the actions or use cases available, such as Authenticate, Reserve a book, Renew a book, and Pays fine for regular users. For administrative users like the Librarian, use cases include Add record, Search and update database, and Delete book record. The diagram uses relationships like "extend" and "include" to show dependencies and optional interactions, for instance, an Invalid username or password message extends the Authenticate use case.

6. Sequence Diagram

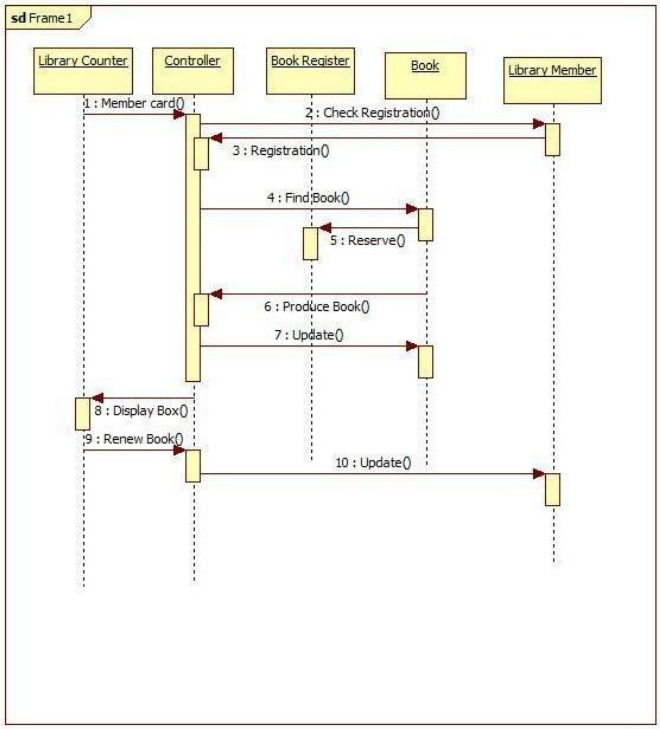


Fig 3.6 Sequence Diagram for Library Management System

This Sequence Diagram illustrates the step-by-step interaction between different components of the system during a book reservation or renewal process at a library counter. The participants (lifelines) include the Library Counter, a central Controller, Book Register, Book, and Library Member. The sequence begins when a member presents their card at the counter. The Controller then orchestrates the entire process by sending messages to the other components in a specific order to check the member's registration, find and reserve the book, and update the book's status. The diagram clearly shows the flow of communication and the order in which operations are performed to fulfill the user's request.

7. Activity Diagram

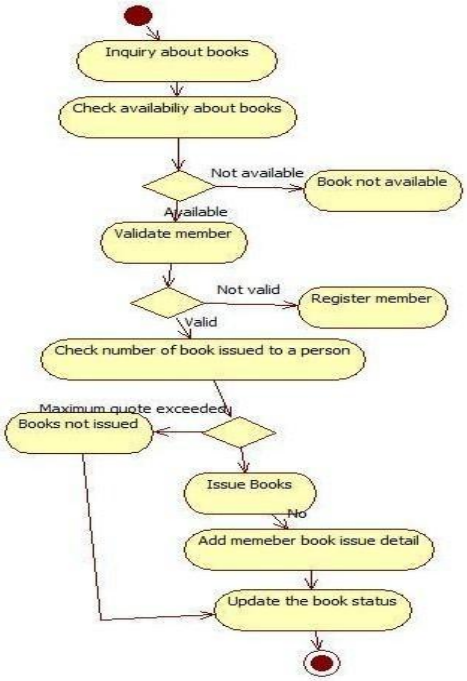


Fig 3.7 Activity Diagram for Library Management System

This Activity Diagram details the business logic for the process of issuing a book to a member. The workflow starts with an Inquiry about books and proceeds to Check availability. A decision point follows: if the book is not available, the process terminates. If it is available, the system then validates the member's status. If the member is not valid, they may be prompted to register. For a valid member, the system checks the number of books they have already issued. Another decision checks if they have exceeded their borrowing limit. If the limit is not exceeded, the book is issued, details are recorded, the book's status is updated, and the process completes successfully. If the limit is exceeded, the book is not issued.

4. Stock Maintenance System

4.1. Problem Statement

Organizations that manage inventory often rely on manual registers or outdated spreadsheets to track stock levels, incoming items, outgoing items, and reorder requirements. This leads to frequent issues such as inaccurate stock counts, delayed replenishment, loss of items, and poor visibility into inventory movement. To resolve these challenges, a computerized Stock Maintenance System is required. The system should record all stock entries and withdrawals, maintain real-time quantity levels, generate alerts when items fall below minimum thresholds, and produce reports for analysis. It must support staff in managing inventory efficiently and provide administrators with accurate data for decision-making. The goal is to design a system that improves accuracy, reduces manual errors, tracks stock flow transparently, and ensures timely replenishment of materials.

4.2. SRS-Software Requirements Specification

4.2.1. Introduction

Purpose: To define the requirements for a system that manages and tracks stock levels, item entries, item issues, and reorder needs within an organization.
Scope: The system records incoming and outgoing stock, updates available quantities, maintains supplier details, generates low-stock alerts, and provides inventory reports for staff and administrators. It ensures accuracy, speed, and reliability in managing inventory operations.
Overview: The system automates stock handling, reduces manual errors, provides real-time stock visibility, and improves decision-making for inventory control.

4.2.2. General Description

The Stock Maintenance System automates the process of monitoring inventory levels, tracking item movements, and managing reorder points. Users include store staff, warehouse managers, and administrators. The system ensures accurate stock updates, reduces shortages, prevents overstocking, and centralizes all inventory-related information for quick access.

4.2.3. Functional Requirements

- Record new stock items with item details.
- Update item quantities when stock is added or issued.
- Track incoming (purchase) and outgoing (issue) transactions.
- Generate notifications when items fall below minimum thresholds.
- Maintain supplier details and purchase information.
- Generate daily, weekly, and monthly stock reports.
- Provide admin login and role-based permissions.
- View and search inventory by item type, category, or supplier.

4.2.4. Interface Requirements

- User interface for staff to add, update, and issue stock.
- Admin interface for managing users, suppliers, and system settings.

- Database interface for storing item, supplier, and transaction records.
- Report interface for viewing and exporting stock summaries.

4.2.5. Performance Requirements

- Response time: ≤2–3 seconds per transaction.
- System must handle 1,000+ stock records efficiently.
- System uptime: 99% during operational hours.

4.2.6. Design Constraints

- The system must run on Windows/Linux desktop environment.
- Local database storage without mandatory Internet dependency.
- Must support barcode/manual entry for items if required.

4.2.7. Non-Functional Attributes

- Security: Authorized access only; secure handling of inventory data.
- Reliability: Stable operation with accurate and consistent stock updates.
- Scalability: Ability to add more categories, users, or warehouses in future.
- Usability: Simple interface suitable for staff with minimal training.

4.2.8. Preliminary Schedule & Budget

- Development duration: 4–6 months.
- Budget:~Rs.1,50,000 - Rs.2,25,000
- Phases: Requirements → Design → Development → Testing → Deployment.

4.3. Class Diagram

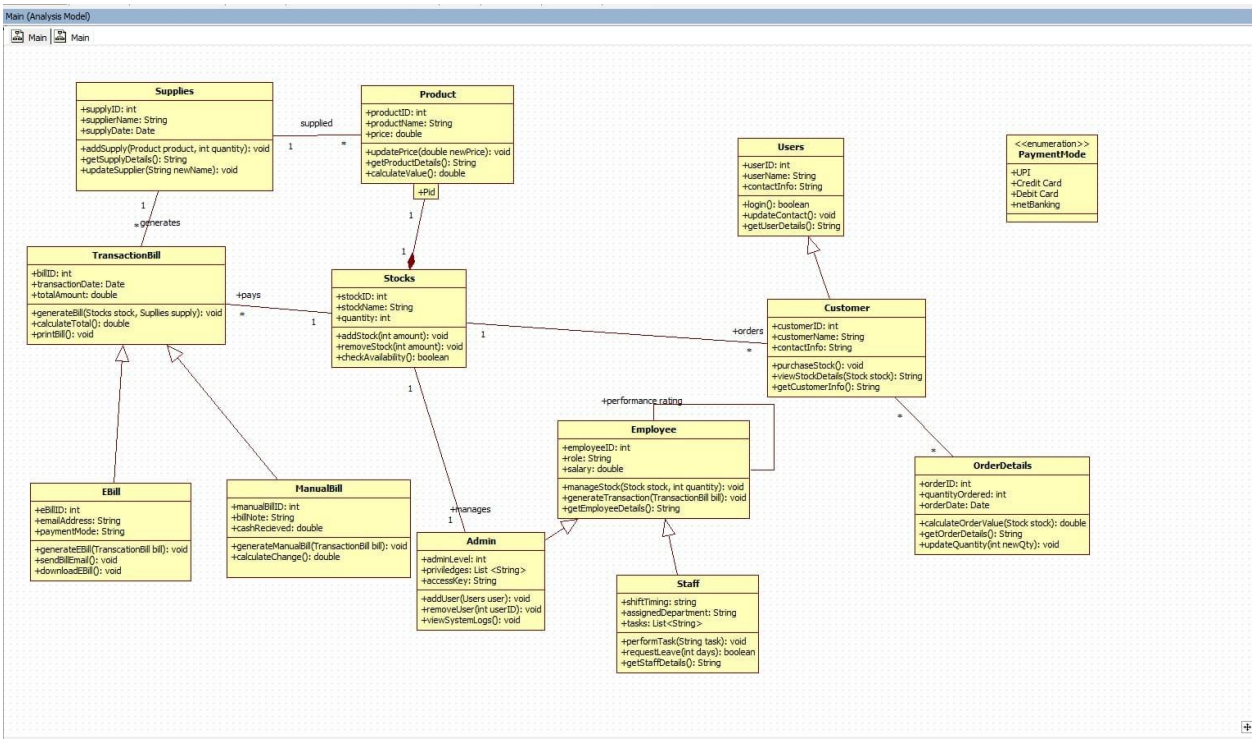


Fig 4.1 Class Diagram for Stock Maintenance System

This is a Class Diagram that provides a detailed blueprint of an inventory or stock management system. It defines the main components (classes) such as Product, Stocks, Suppliers, Customer, and Employee. The diagram outlines the properties (attributes) and functionalities (methods) of each class. It also illustrates the relationships between them; for example, a Customer places an OrderDetails, an Employee manages Stocks, and a Supplier is associated with TransactionBill and supplies Product. The inheritance relationship is also shown, with Admin and Staff being specialized types of Employee, and EBill and ManualBill being types of TransactionBill.

4.4 State Diagram

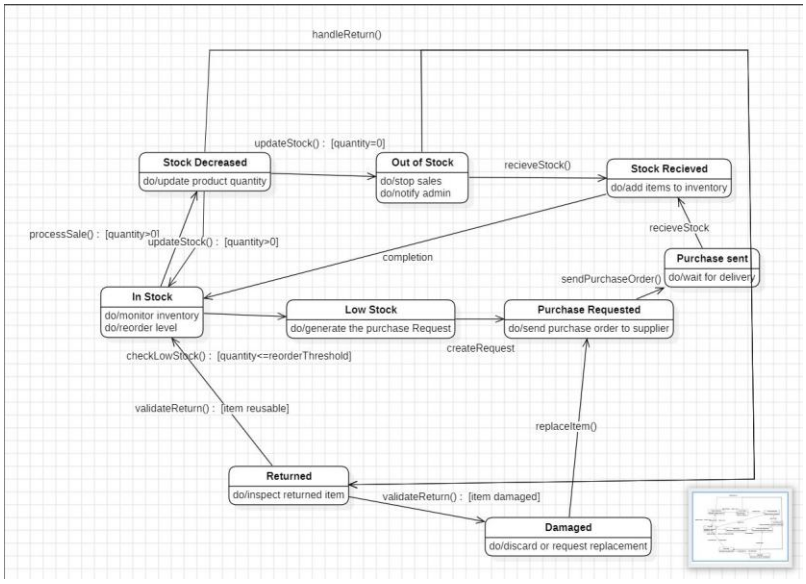


Fig 4.2 State Diagram for Stock Maintenance System(1)

This State Machine Diagram visualizes the dynamic behavior and lifecycle of stock for a particular item. The stock begins in the In Stock state. A sale (processSale) transitions it to Stock Decreased. If the quantity reaches zero, it moves to Out of Stock, which triggers a notification to the admin. When new inventory arrives (recieveStock), the state becomes Stock Recieved and then returns to In Stock. The diagram also handles reordering: if the stock level falls to a certain threshold, it enters the Low Stock state, which leads to a Purchase Requested and Purchase sent sequence to replenish inventory. Furthermore, it manages returns, with states like Returned and Damaged to handle items coming back from customers.

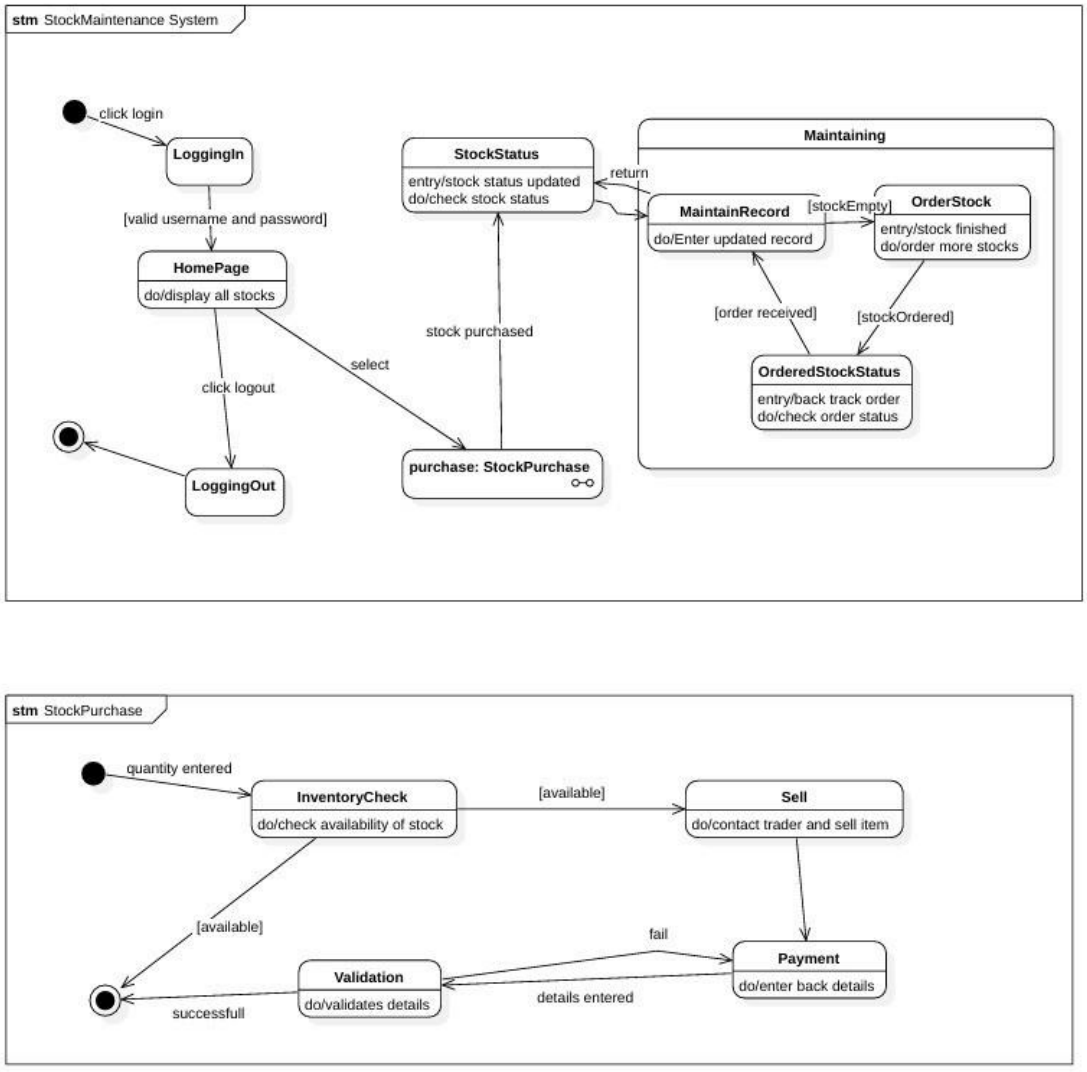


Fig 4.3 State Diagram for Stock Maintenance System(2)

These two diagrams are State Machine Diagrams that illustrate the user interaction and processes within the Stock Maintenance System. The first diagram shows the overall system flow, starting from a user LoggingIn. Upon successful login, the user reaches the HomePage. From here, they can select a stock to view its StockStatus or initiate a purchase. The Maintaining section is a composite state that includes actions like MaintainRecord and OrderStock. The second diagram provides a detailed view of the StockPurchase process. It starts with an InventoryCheck to verify availability. If available, it moves to Validation of details. Upon successful validation, the process proceeds to Sell and Payment, completing the purchase transaction.

4.5 Use Case Diagram

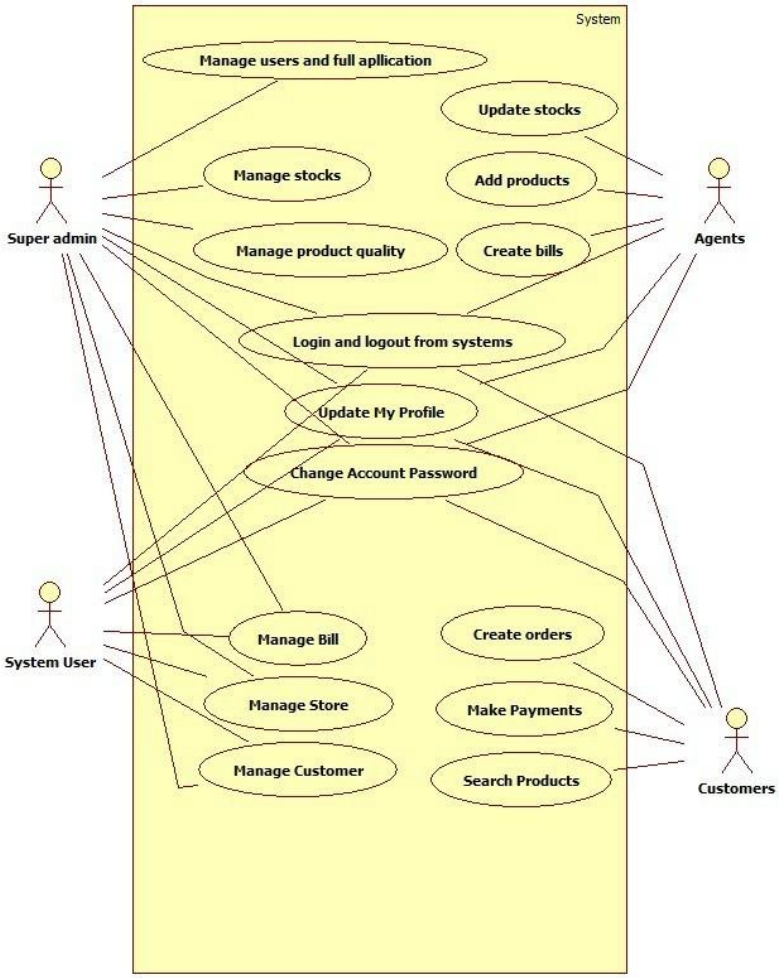


Fig 4.4 Use Case Diagram for Stock Maintenance System

This is a Use Case Diagram that outlines the functionalities of the system and the actors who interact with it. It identifies four main actors: Super admin, System User, Agents, and Customers. The Super admin has the highest level of access, with use cases like Manage users and full application. Agents are responsible for inventory-related tasks such as Update stocks and Add products. System Users have more general permissions like managing bills and customers. Customers can perform actions like Create orders, Make Payments, and Search Products. The diagram provides a clear, high-level view of what the system does and who can do what.

4.6 Sequence Diagram

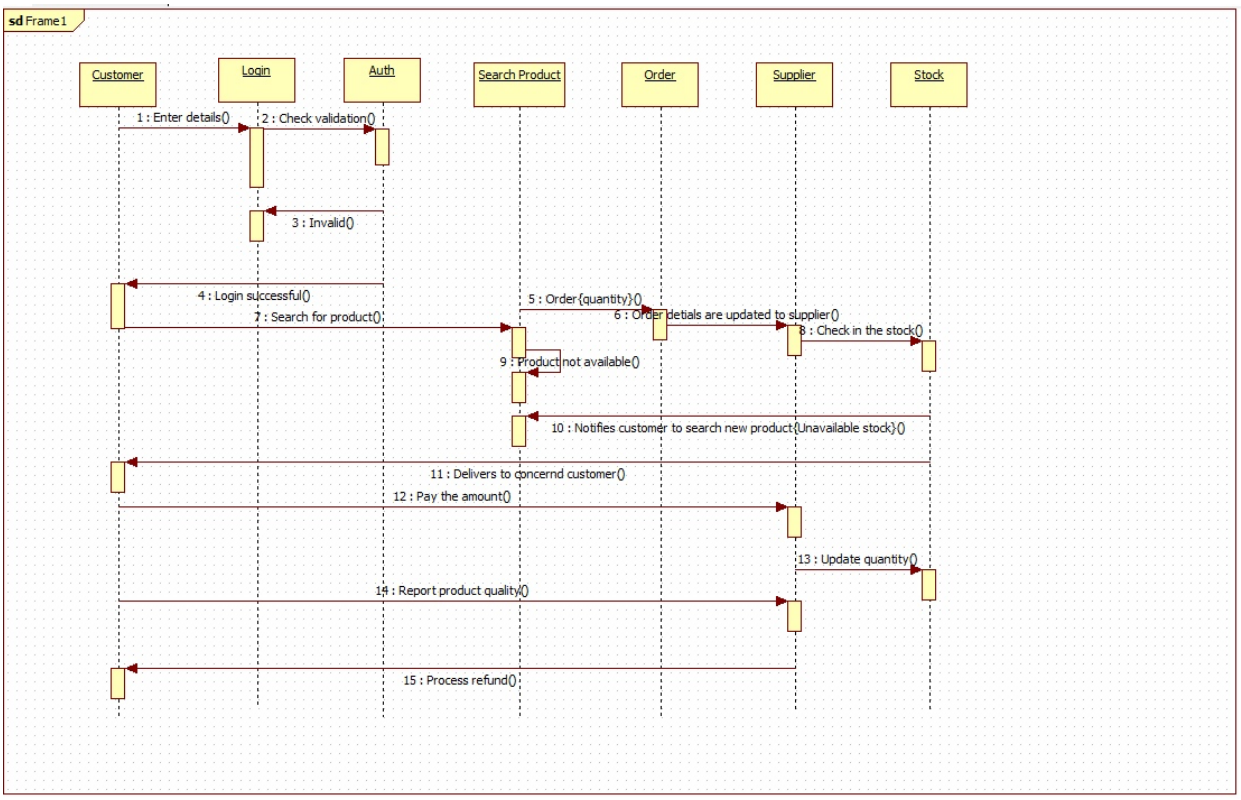


Fig 4.5 Sequence Diagram for Stock Maintenance System

This Sequence Diagram illustrates the interactions that occur when a customer orders a product. The participants are Customer, Login, Auth (Authentication), Search Product, Order, Supplier, and Stock. The sequence begins with the Customer entering their details to log in. After successful authentication, the customer searches for a product. The system checks the Stock. If the product is available, the customer places an Order, and the details are updated with the Supplier. The diagram also shows subsequent steps like the supplier delivering the product, the customer making a payment, and potentially reporting on product quality, which updates the supplier's records. It effectively shows the message flow between different system components over time.

4.7 Activity Diagram

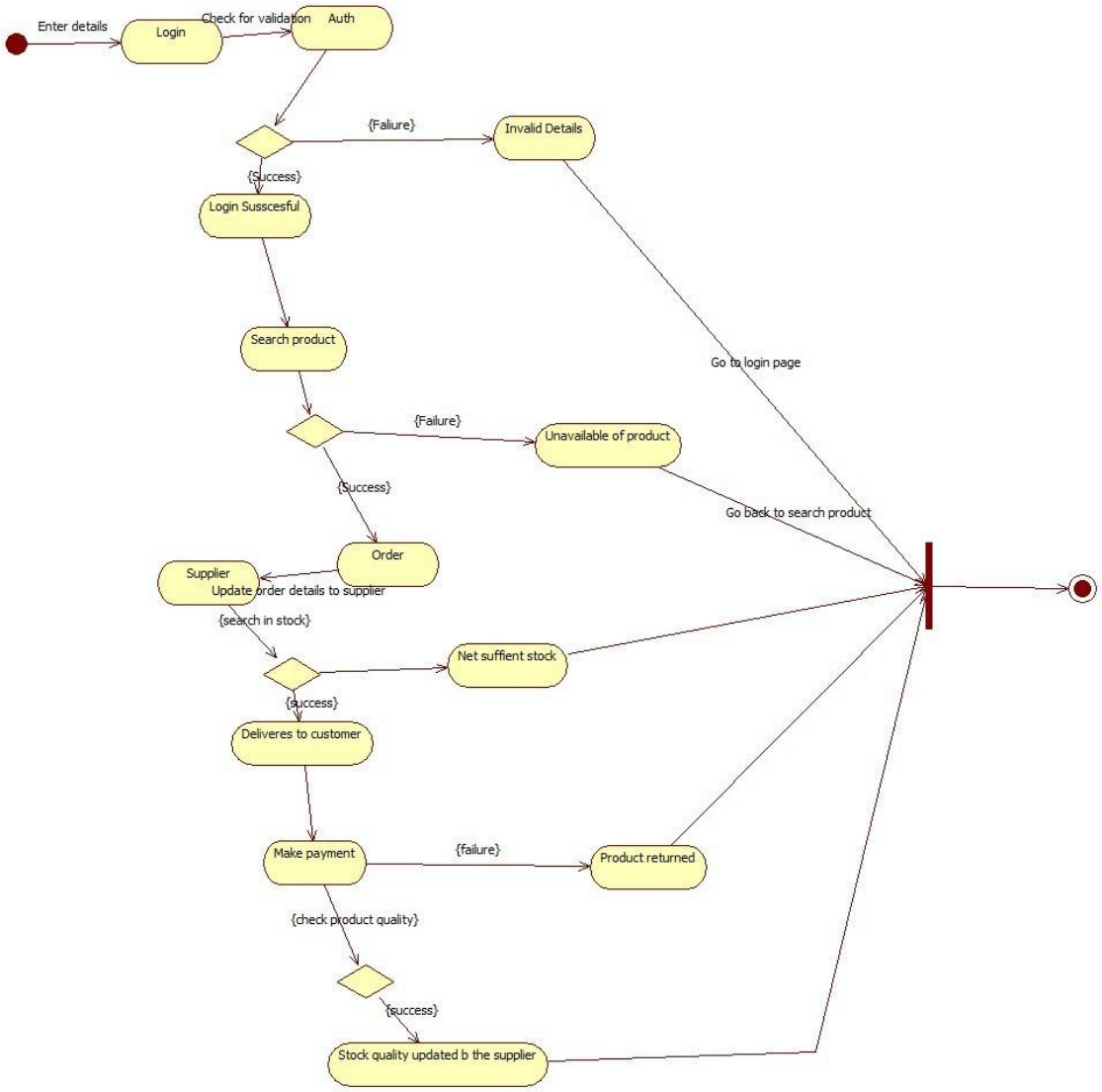


Fig 4.5 Activity Diagram for Stock Maintenance System

This Activity Diagram maps out the end-to-end workflow for a customer purchasing an item. The process starts with the user entering their details for Login and Authentication. A decision node checks for success or failure. If successful, the flow continues to Search product. Another decision checks if the product is available. If it is, the customer can place an Order. The system then checks if there is sufficient stock. If so, the product is delivered, and the customer proceeds to Make payment. The final steps involve a quality check, which may result in updating the stock quality with the supplier, after which the entire process comes to an end. The diagram clearly shows the various paths and decision points in the transaction process.

5. Passport Authentication System

5.1. Problem Statement

Existing passport authentication systems face issues such as fraud, counterfeiting, outdated technology, and inefficiencies in verification processes. Manual and inconsistent methods increase the risks of identity theft, human error, and delays at borders. A secure, efficient, and standardized Passport Authentication System is required to automate verification, enhance fraud detection, reduce processing times, and ensure global interoperability while protecting user privacy.

5.2. SRS-Software Requirements Specification

5.2.1. Introduction

- Purpose:The purpose of this document is to define the requirements for a Passport Authentication System (PAS) that ensures secure, reliable, and efficient passport verification. It will reduce fraud, enhance border security, and streamline processing.
- Scope: Verify the authenticity of passports using biometric and digital verification,Detect fraudulent or counterfeit passports, Support real-time communication with central/global databases for cross-border verification, Reduce manual errors and processing delays, Ensure data security, privacy, and compliance with international standards.
- Overview: This system integrates biometric validation, machine-readable zones (MRZ), RFID/e-passport verification, and secure database checks. It will be used at airports, borders, embassies, and other checkpoints.

5.2.2 General Description

The PAS automates the verification process by scanning passports, cross-checking details with databases, and validating biometric features. Users include immigration officers, border security staff, and government agencies. Benefits include improved fraud detection, reduced delays, better interoperability, and increased traveler trust.

5.2.3 Functional Requirements

- Scan and read passports (MRZ, RFID chip, barcode).
- Verify passport authenticity against security features.
- Cross-check details with government/Interpol databases.
- Authenticate traveler using biometric data (face/fingerprint match).
- Flag fraudulent, stolen, or expired passports.
- Generate real-time alerts and logs.
- Support multi-country interoperability standards (ICAO compliant).
- Provide an admin panel for monitoring and reporting.

4. Interface Requirements

- Hardware Interface: Biometric scanners, passport readers.
- Software Interface: Secure connection to central databases.

5.2.5 Performance Requirements

- Verification response time: ≤ 5 seconds per passport.
- Support 5,000+ passport verifications per day per checkpoint.
- System uptime: 99.9%.

5.2.6 Design Constraints

- Must comply with ICAO e-passport standards.
- Must support biometric encryption and secure key management.
- Must comply with data privacy laws (GDPR, etc.).

5.2.7 Non-Functional Attributes

- Security: End-to-end encryption, digital signatures, and tamper-proof storage.
- Reliability: Redundant servers and backup mechanisms.
- Usability: Simple interface for officers with minimal training required.
- Scalability: Expandable to multiple countries and checkpoints.

5.2.8 Preliminary Schedule & Budget

- Development Duration: 8–12 months.
- Budget Estimate: $2M–$5M (depending on scale and global integration).

5.3. Class Diagram

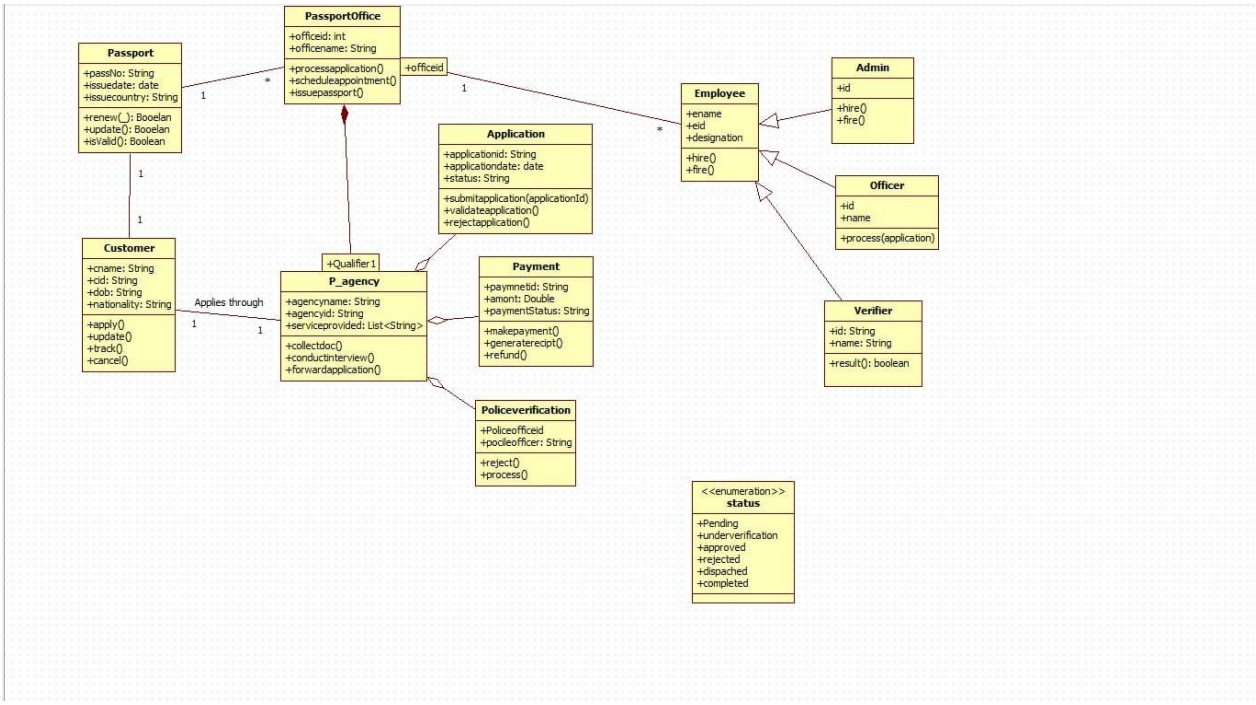


Fig 5.1 Class Diagram for Passport Authentication System

The Class Diagram illustrates the static structure of the system by defining the different objects (classes), their attributes, and their relationships. Key classes include Customer, Passport, Application, PassportOffice, and various types of Employee like Officer and Verifier. It shows, for example, that a Customer can have one Passport and can apply for an Application. A PassportOffice employs various Employee roles and processes applications. This diagram essentially serves as the blueprint for the system's database and object-oriented programming structure.

5.5 State Diagram

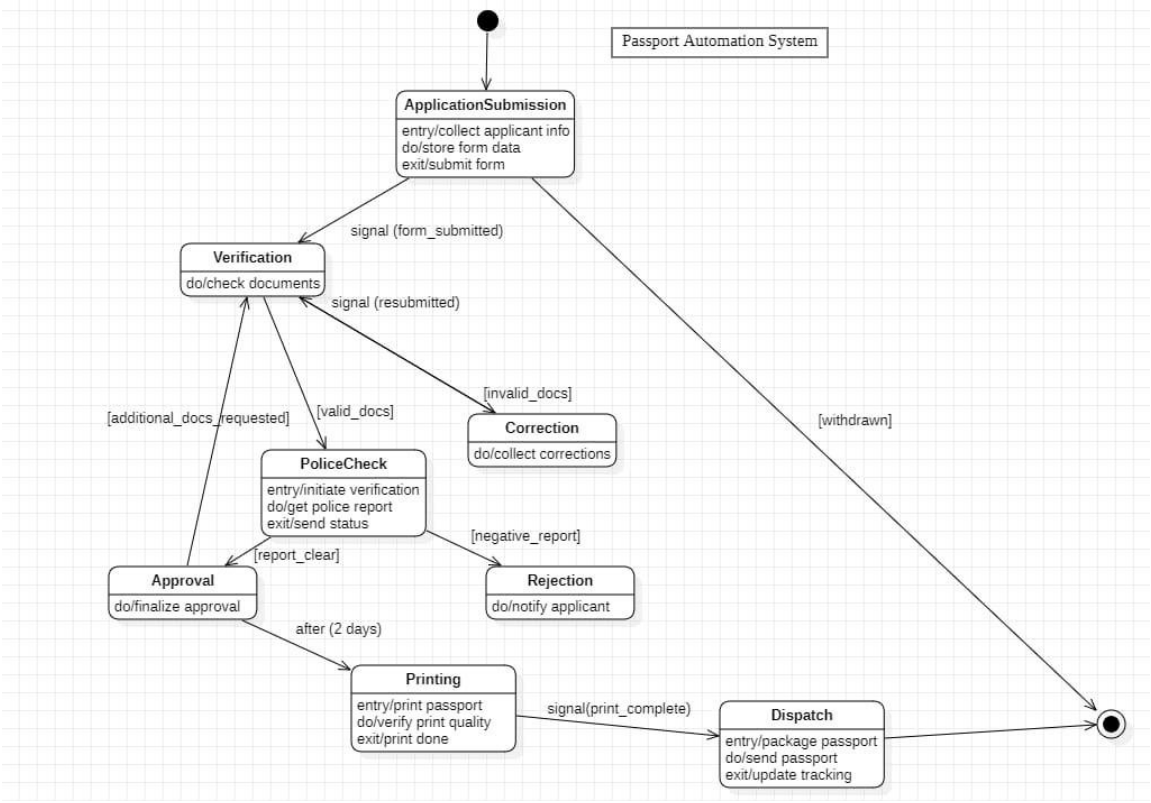


Fig 5.2 State Diagram for Passport Authentication System

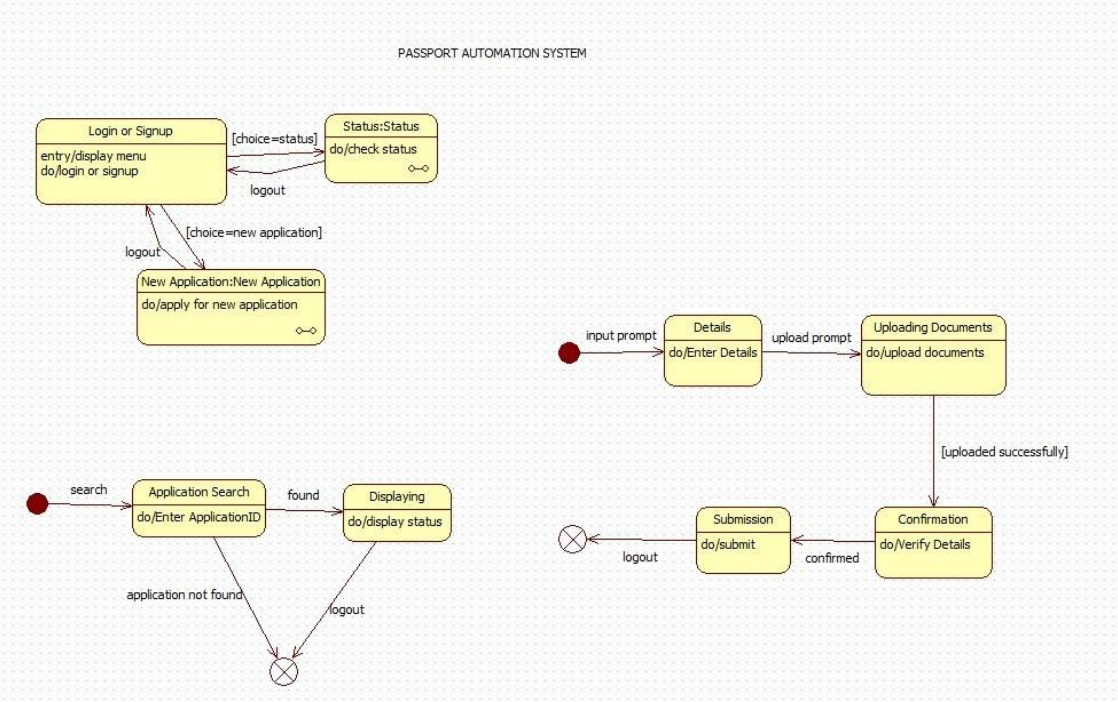


Fig 5.3 Advance State Diagram for Passport Authentication System

The first State Machine Diagram depicts the lifecycle of a passport application from beginning to end. It shows the various states an application can be in, such as ApplicationSubmission, Verification, PoliceCheck, Approval, Printing, and Dispatch. The arrows represent transitions between these states, triggered by events like form_submitted, valid_docs, or negative_report. This diagram is crucial for understanding the dynamic behavior and the overall workflow of the passport processing journey.

The second State Machine Diagram focuses on the user interface and interaction flows. It is composed of several smaller diagrams showing different user activities. One part details the process for a user to Login or Signup, check Status, or start a New Application. Another part outlines the steps for submitting a new application, which includes entering Details, Uploading Documents, Confirmation, and final Submission. The last part shows how a user can search for an application by ID to display its status.

5. Use Case Diagram

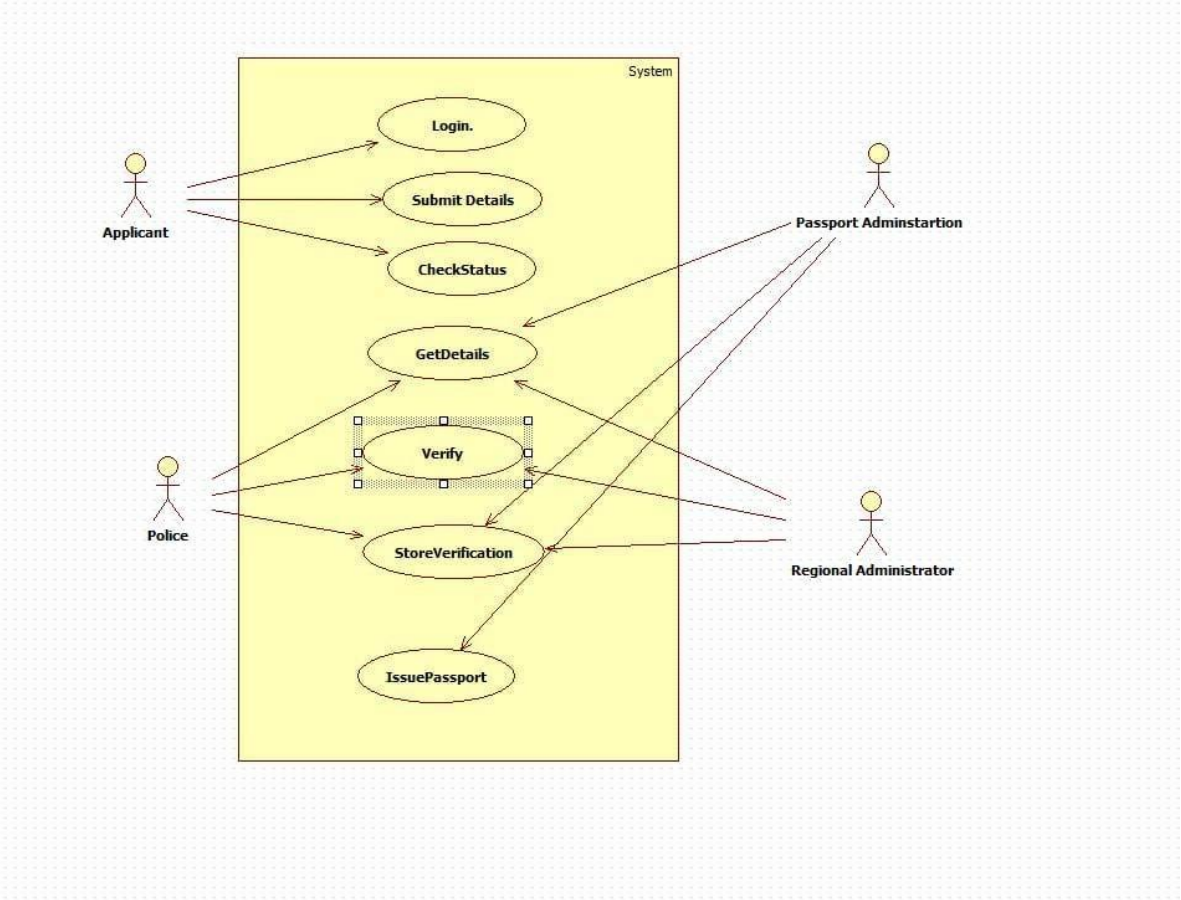


Fig 5.4 Use Case Diagram for Passport Authentication System

The Use Case Diagram provides a high-level overview of the system's functionalities from the perspective of different users (actors). The actors identified are the Applicant, Police, Passport Administrator, and Regional Administrator. It outlines the key actions or "use cases" each actor can perform. For instance, an Applicant can Submit Details and CheckStatus, while the Police actor is involved in StoreVerification. This diagram helps in understanding the system's requirements and who interacts with which function.

6. Sequence Diagram

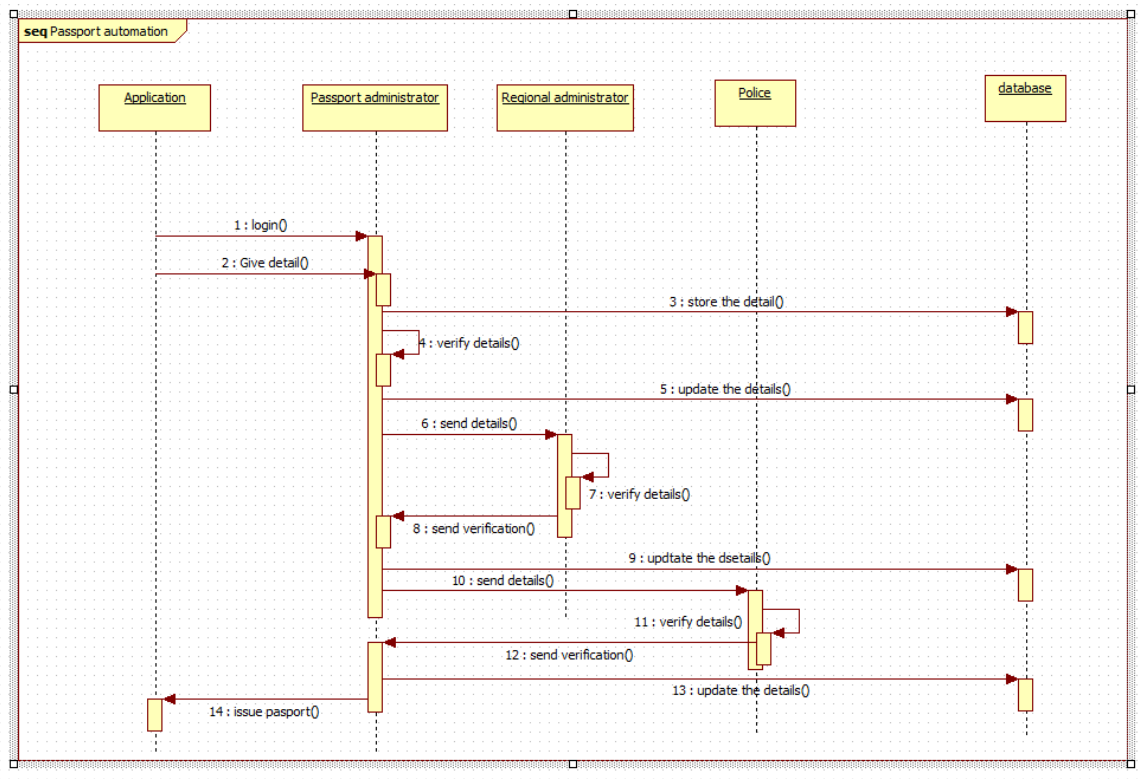


Fig 5.5 Sequence Diagram for Passport Authentication System

The Sequence Diagram details the interactions between different components of the system over time for a specific scenario. It shows the chronological sequence of messages exchanged between the Application, Passport administrator, Regional administrator, Police, and the database. The diagram illustrates the step-by-step process of how details are submitted, stored, verified by different authorities in a specific order, and how the database is updated at each stage, leading up to the final issue passport command.

7. Activity Diagram

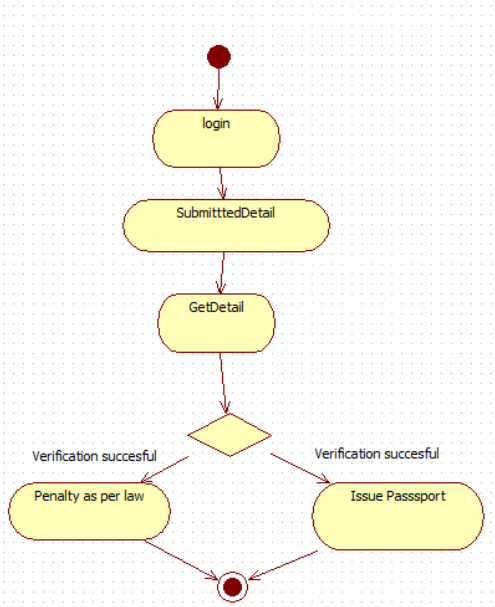


Fig 5.6 Activity Diagram for Passport Authentication System

The Activity Diagram illustrates a specific workflow within the system, focusing on the sequence of actions and decisions. This particular diagram shows the process starting from login, followed by SubmittedDetail and GetDetail. After the details are retrieved, a decision point is reached based on the outcome of the verification. If the verification is successful, the system proceeds to Issue Passport; otherwise, it leads to a "Penalty as per law". This highlights the conditional logic in the verification process.