B.M.S. COLLEGE OF ENGINEERING
BENGALURU

Autonomous Institute, Affiliated to VTU

Lab Record

Object-Oriented Modeling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by:

GOPAL AGRAWAL (1BM22CS361)

Department of Computer Science and Engineering

B.M.S. College of Engineering

Bull Temple Road, Basavanagudi, Bangalore 560 019

September-January 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 **Gopal Agrawal (1BM22CS361)** , during the 5th Semester Sep 24- Jan2025.

Signature of the Faculty Incharge:

Prof. Rekha G S

Assistant Professor

Department of Computer Science and Engineering

B.M.S. College of Engineering, Bangalore

Table of Contents

1. Hotel Management System

1.1 Introduction

The hospitality industry is a fast-paced and dynamic environment where efficient management is crucial for smooth operations and customer satisfaction. A Hotel Management System is needed to streamline operations, enhance guest experiences, and reduce manual errors. The system should automate key processes such as room booking, guest check-in/check-out, payment processing, and inventory management, while providing robust analytics and reporting for business decision-making.

1.2 SRS-Software Requirements Specification

1.2.1 Purpose of this Document

The primary purpose of this document is to outline the requirements and specifications for the development of a Hotel Management System. This document serves as a foundation for stakeholders, including the development team, management, and end-users, to understand the system's objectives, scope, and functionalities. It provides a clear reference for all phases of the software development lifecycle.

1.2.2 Scope of this Document

This document encompasses the overall working and primary objectives of the Hotel Management System. The system aims to simplify and automate hotel operations, including reservations, check-ins, check-outs, billing, and customer management. The project is designed to provide value by improving operational efficiency and customer satisfaction. The estimated development cost is $50,000, and the expected completion time is six months.

1.2.3 Overview

The Hotel Management System is a comprehensive software solution designed to handle various aspects of hotel operations. It facilitates seamless communication between staff and guests, automates routine tasks, and ensures the accuracy and security of data. The system will include modules for reservations, room allocation, billing, customer feedback, and reporting.

1.2.4 General Description

General Functions

The Hotel Management System will streamline hotel operations by automating key processes. Key functions include:

Facilitating online and offline reservations.

- Managing customer check-ins and check-outs.
- Handling billing and invoicing.
- Generating reports for hotel management.

User Objectives

The system will cater to the following user objectives:

- Simplify operational tasks for hotel staff.
- Enhance guest experience through efficient service.
- Provide management with data-driven insights.

User Characteristics

The user community will consist of:

- Hotel Staff: Front desk operators, housekeeping, and management.
- Guests: Individuals or groups seeking accommodation.
- Administrators: IT personnel responsible for system maintenance.

Benefits

- Increased efficiency in operations.
- Enhanced customer satisfaction.
- Improved data accuracy and security.
- Detailed reporting and analytics for better decision-making.

1.2.5 Functional Requirements

The functional requirements of the Hotel Management System include:

Reservation Management:

a. Accept online and offline bookings.

b. Maintain a database of room availability.

Check-In/Check-Out Management:

a. Handle guest check-ins and check-outs seamlessly

b. Generate digital keys and allocate rooms.

Billing and Payment Processing:

a. Calculate total charges automatically.

b. Support multiple payment methods.

Customer Management:

a. Maintain guest profiles and history.

Reporting:

a. Generate daily, weekly, and monthly reports.

1.2.6 Interface Requirements

Software Interfaces

The system will interface with:

- Online booking platforms via APIs.
- Payment gateways for secure transactions.
- Reporting tools for data visualization.

User Interfaces

- **Staff Interface**: Intuitive dashboards for managing reservations, billing, and reports.
- **Guest Interface**: Mobile-friendly application or website for reservations and feedback.

1.2.7 Performance Requirements

The Hotel Management System must meet the following performance metrics:

- Handle up to 1,000 concurrent users.

- Process a reservation request within 2 seconds.
- Ensure 99.9% system uptime.
- Maintain an error rate below 0.1%.

1.2.8. Design Constraints

Hardware Constraints

- The system will require dedicated servers with a minimum of 16 GB RAM and 1 TB SSD.

Software Constraints

- The system must be developed using a secure programming framework such as Java or Python.
- Database management will use MySQL or PostgreSQL.

Regulatory Constraints

- Must comply with GDPR and PCI DSS for data protection and payment security.

1.2.9 Non-Functional Attributes

The system must exhibit the following attributes:

- **Security**: Role-based access control and data encryption.
- **Portability**: Compatibility with major operating systems.
- **Reliability**: Automatic failover mechanisms.
- **Scalability**: Capability to add new features and support more users.
- **Data Integrity**: Ensure accurate and consistent data storage.

1.2.10 Preliminary Schedule and Budget

Schedule

The project will be executed in the following phases:

- Requirements Gathering: 1 month
- Design: 1 month
- Development: 3 months
- Testing and Deployment: 1 month

Budget

The estimated budget for the project is $50,000, distributed

as follows: ● Development: $30,000

- Testing: $10,000
- Deployment and Maintenance: $10,000

1.3 Class Diagram:

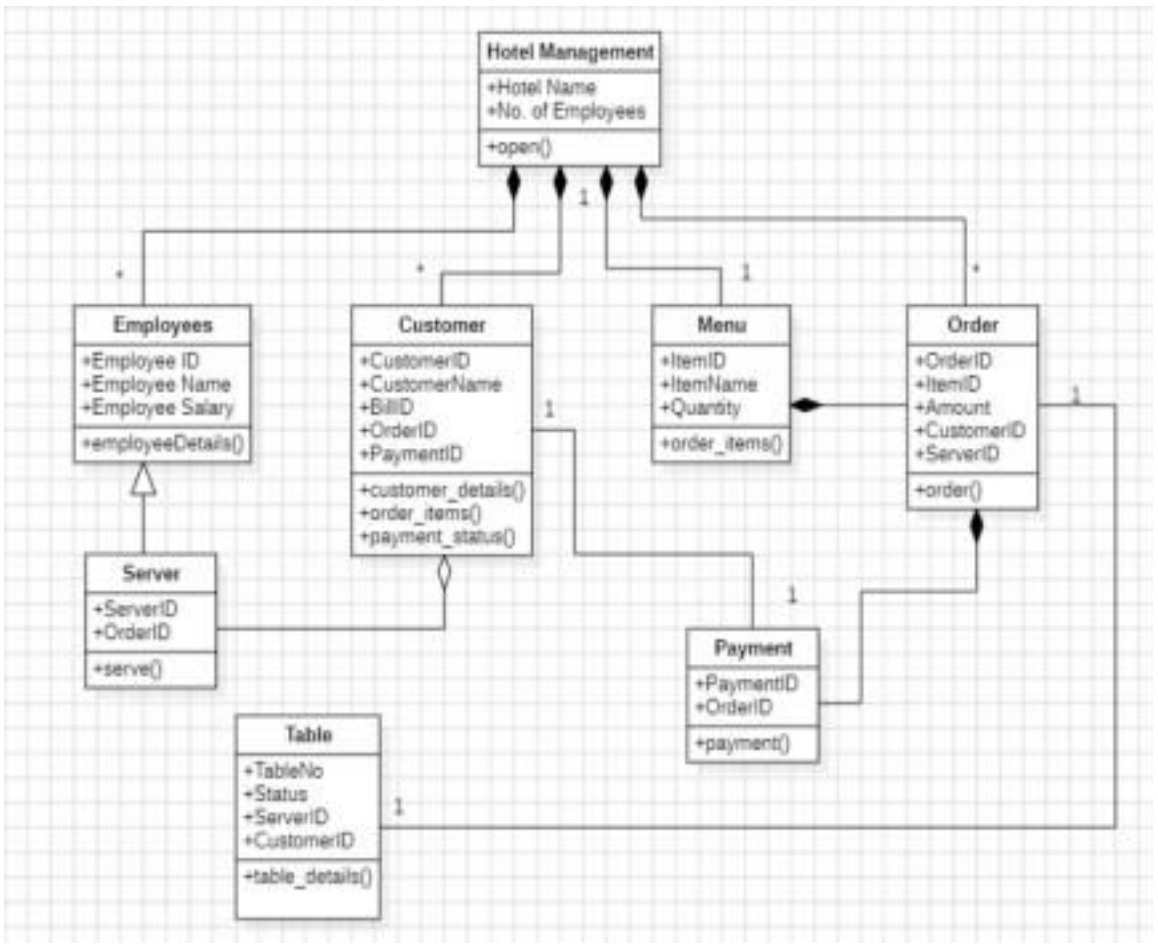


Fig 1.1

Brief Description:

The class diagram for the Hotel Management System (HMS) represents the key entities, their

attributes, and relationships in the system. Below are the main components typically included in

the class diagram for HMS:

1. Guest:

○ Attributes: Guest ID, Name, Contact Information.

○ Methods: BookRoom(), Checkout(), Pay().

2. Room:

○ Attributes: Room ID,Guest ID.

○ Methods: CheckAvailability(), isFull().

3. Booking:

○ Attributes: Booking ID, Check-in Date, Check-out Date, Total Price, Payment Status.

○ Methods: CreateBooking(), ModifyBooking(), CancelBooking().

4. Payment:

○ Attributes: Payment ID, Amount, Payment Method (Cash, Credit Card, Online), Payment Date.

○ Methods: ProcessPayment(), GenerateInvoice().

5. Staff:

○ Attributes: Staff ID, Name, Role (Receptionist, Manager, Housekeeping), Contact Information, Shift Details.

○ Methods: AssignRoom(), ManageBookings(), UpdateInventory().

6. Administrator:

○ Attributes: Admin ID, Name, Email, Role.

○ Methods: ManageStaff(), GenerateReports(), ConfigureSystem().

1.4 State Diagram :

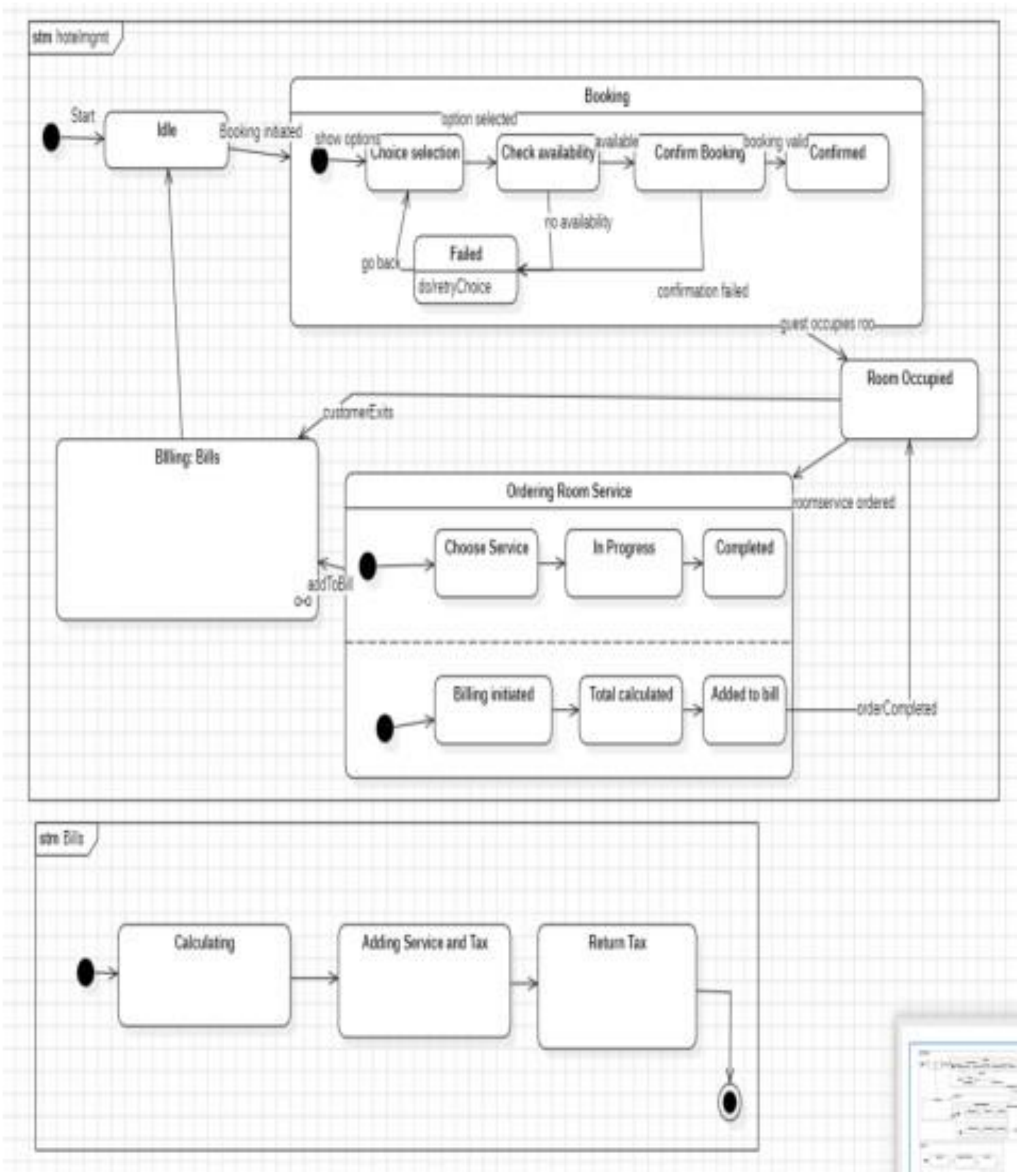


Fig 1.2

Brief Description:

The state diagram for the Hotel Management System (HMS) represents the lifecycle of two key processes: Booking and Payment, detailing the stages and transitions for each.

● Booking Process:

The booking process begins with the Booking Initiated state when a guest starts reserving a room. It then transitions to the Room Selected state once a room is chosen. From here, the process moves to Booking Confirmed if the booking is finalized or to Booking Canceled if the guest or staff cancels it.

● Payment Process:

Payment starts in the Payment Pending state when the booking is confirmed. It transitions to Payment In Process when the payment begins. Depending on the outcome, it moves to Payment Successful if the transaction is completed or to Payment Failed in case of errors like insufficient funds or technical issues, allowing retries if needed.

1.5 Use-Case Diagram:

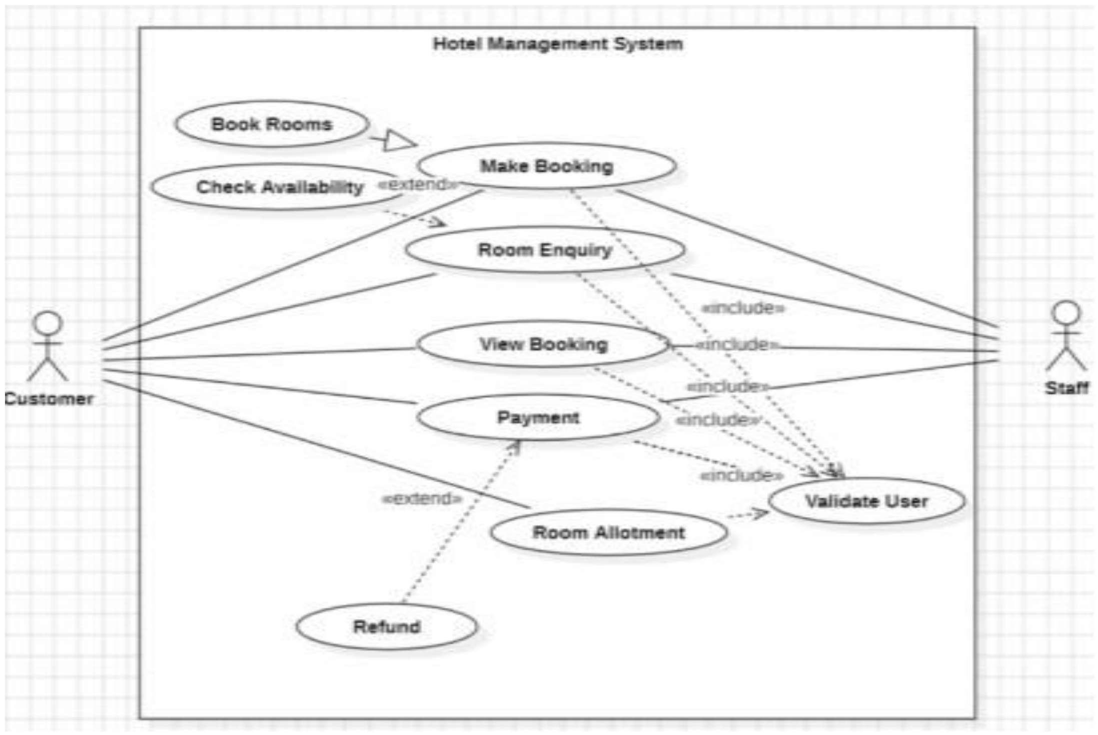


Fig 1.3

Brief Description:

The use case diagram illustrates the interactions between users and the system by identifying the primary use cases and actors.

Actors: Guests (Customer), Staff (Receptionist, Manager, Housekeeping).

Use Cases:Book Room, Check Room Availability, Make Payment, Generate Reports,Manage Staff, Update Inventory.

Relationships: Guests interact with use cases like Book Room and Make Payment.Staff interacts with Check Room Availability and Manage Inventory.Administrator handles Generate Reports and Manage Staff.

1.6 Sequence Diagram:

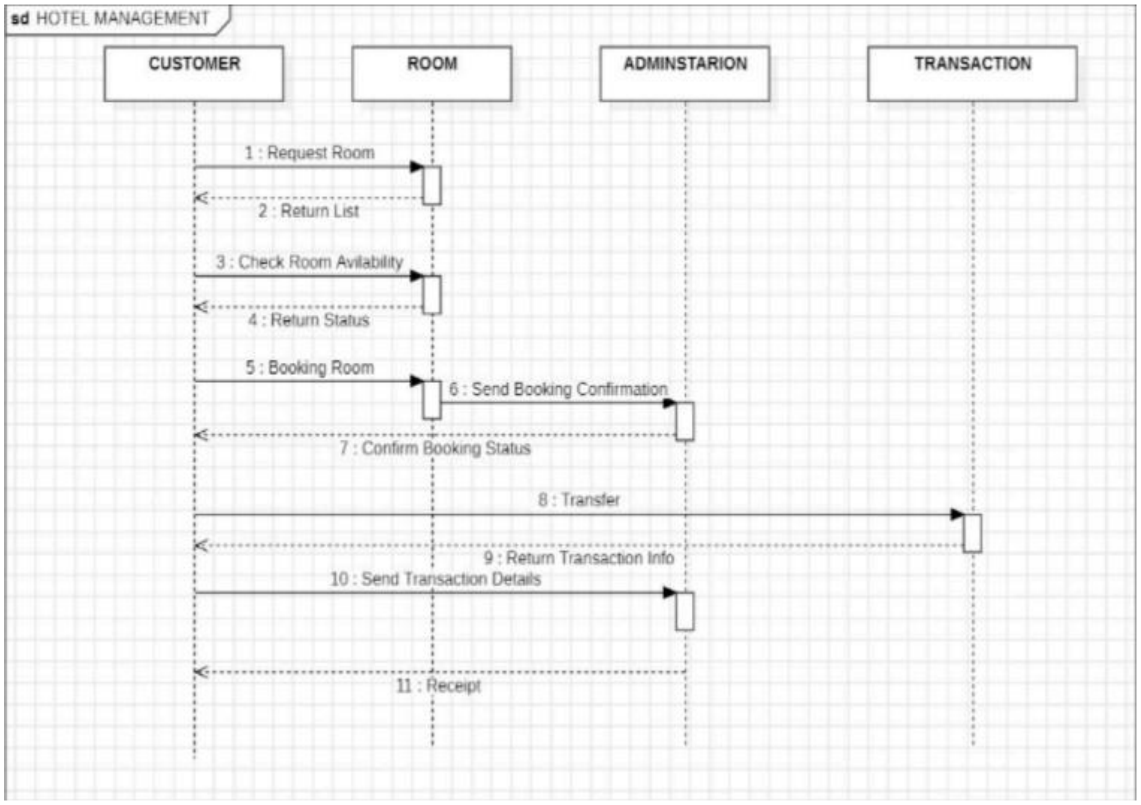


Fig 1.4

Brief Description:

The sequence diagram demonstrates the order of interactions between objects or actors in the system to perform specific tasks.Actors: Guest, Receptionist, System. Steps:1. The guest requests room availability.2. The receptionist checks availability in the system.3. The system responds with available rooms.4. The receptionist confirms the booking and updates the system.5. The system generates a booking confirmation.This diagram helps visualize the step-by-step communication flow required to book a room.

1.7 Activity Diagram:

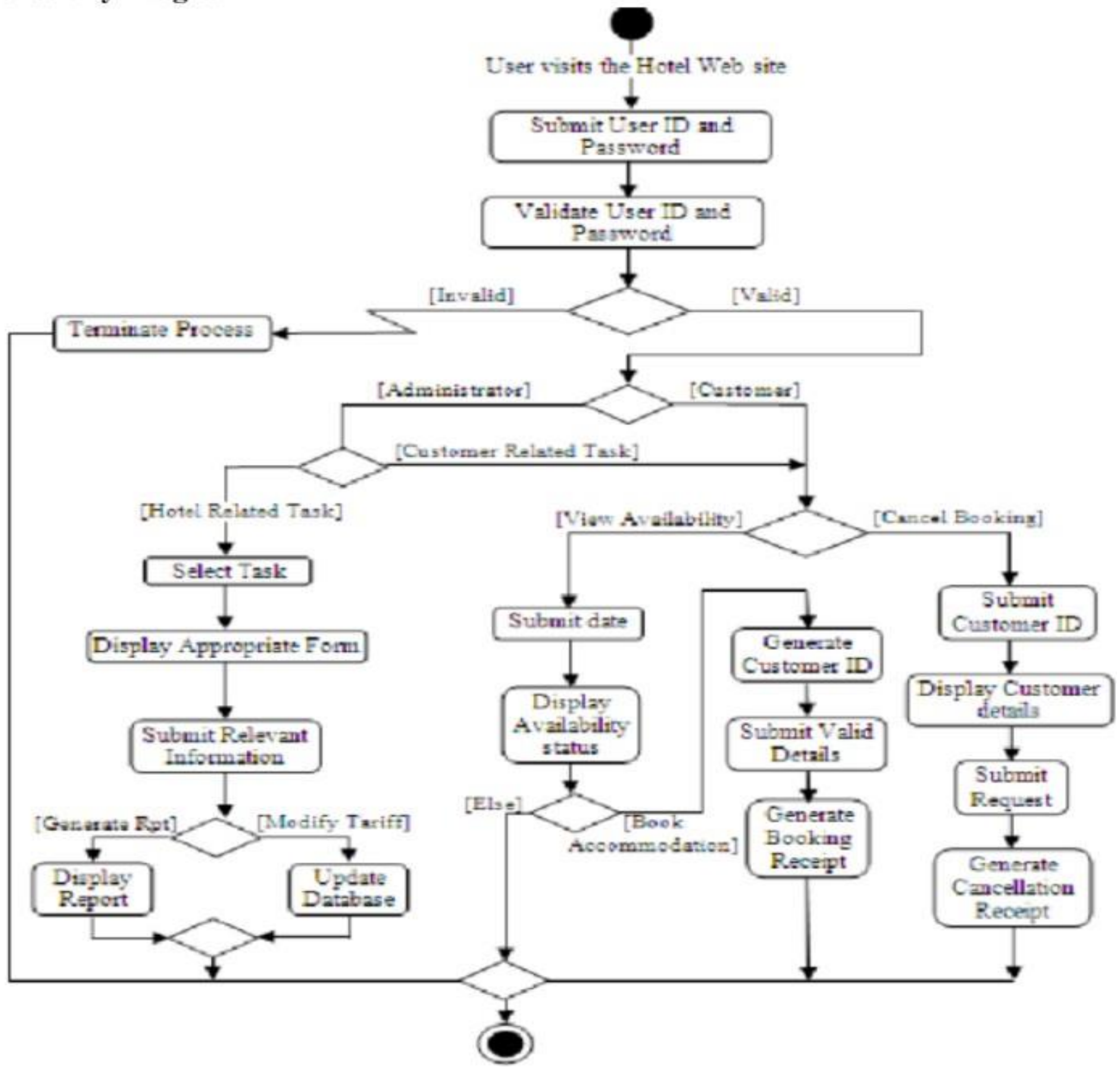


Fig 1.5

The activity diagram focuses on workflows and processes within the HMS, showing how tasks are performed and decision points in the workflow.

● Example: Guest Check-In Process:

○ Activities:

■ Guests arrive at reception.

■ Receptionist retrieves booking details.

■ If a booking exists, proceed to assign the room.

■ If no booking exists, create a new booking.

■ Process payment.

■ Hand over room keys and update system status to "Occupied".

○ Decision Points:

■ Check if the booking exists.

■ Verify payment status before assigning the room.

○ This diagram provides a clear understanding of the workflow for check-ins.

2.CREDIT CARD PROCESSING SYSTEM

2.1 Introduction

The Credit Card Management System (CCMS) is designed to simplify and automate credit card-related processes. It aims to manage user accounts, process applications, track transactions, generate statements, calculate rewards, ensure payment compliance, and prevent fraudulent activities. The system provides a secure, user-friendly interface for customers and staff, ensuring efficient management of credit card operations.

2.2 SRS - Software Requirements Specification

2.2.1 Purpose of this Document

This Software Requirements Specification (SRS) document provides a detailed description of the requirements for the Credit Card Processing System. It serves as a definitive source for technical requirements, system behaviors, and constraints that will guide the development team in building a secure and efficient credit card processing solution. This document will be used by developers, testers, project managers, and stakeholders to ensure alignment on system requirements and expectations.

2.2.2 Scope of this Document

The Credit Card Processing System will provide a comprehensive solution for handling credit card transactions securely and efficiently. The system will:

- Process credit card payments for both online and point-of-sale transactions
- Validate credit card information
- Handle payment authorization

● Manage transaction settlements

● Generate transaction reports

● Provide integration capabilities with major payment gateways

● Development Timeline: 12 months Estimated Cost: $500,000 - $750,000

2.2.3 Overview

The Credit Card Processing System is a secure payment processing solution that enables businesses to accept credit card payments across multiple channels. The system will support major credit card networks (Visa, MasterCard, American Express, Discover), implement industry-standard security protocols, and provide real-time transaction processing capabilities

2.2.4. General Description

The CCMS aims to simplify credit card management for both users and administrators. Customers can apply for cards, make payments, and view transaction histories, while administrators can approve applications, monitor transactions, and detect fraud. The system ensures secure and efficient processing to improve customer satisfaction and operational efficiency.

2.2.5. Functional Requirements

1. Customers can apply for credit cards.
2. Administrators can review and approve or reject applications.
3. Users can view their credit card statements and check transaction histories.
4. The system allows users to make payments and calculate late fees.
5. Fraud detection mechanisms will monitor suspicious activities.

2.2.6. Interface Requirements

● User Interface: A web-based portal for customers and administrators.

● Database Interface: Secure storage of customer and transaction data.

● Payment Gateway Integration: For processing payments securely.

2.2.7. Performance Requirements

● Process card applications within 2 seconds.

● Generate statements for a user within 5 seconds.

● Handle 1,000 concurrent users efficiently.

2.2.8. Design Constraints

● The system must comply with data security standards like PCI DSS.

● Transactions should be encrypted using secure protocols like HTTPS.

2.2.9. Non Functional Attributes

● Security: Protect customer data and ensure secure payments.

● Scalability: Handle a growing number of customers and transactions.

● Reliability: Ensure uptime of 99.9% for uninterrupted service.

2.2.10. Preliminary Schedule and Budget

● Schedule:
Requirements Gathering: 1 month

System Design: 1.5 months

Development and Testing: 3 months

Deployment and Training: 0.5 months

● Budget:
Development: ₹6,00,000

Infrastructure and Hosting: ₹2,00,000

Maintenance and Support: ₹1,50,000

2.3. Class Diagram

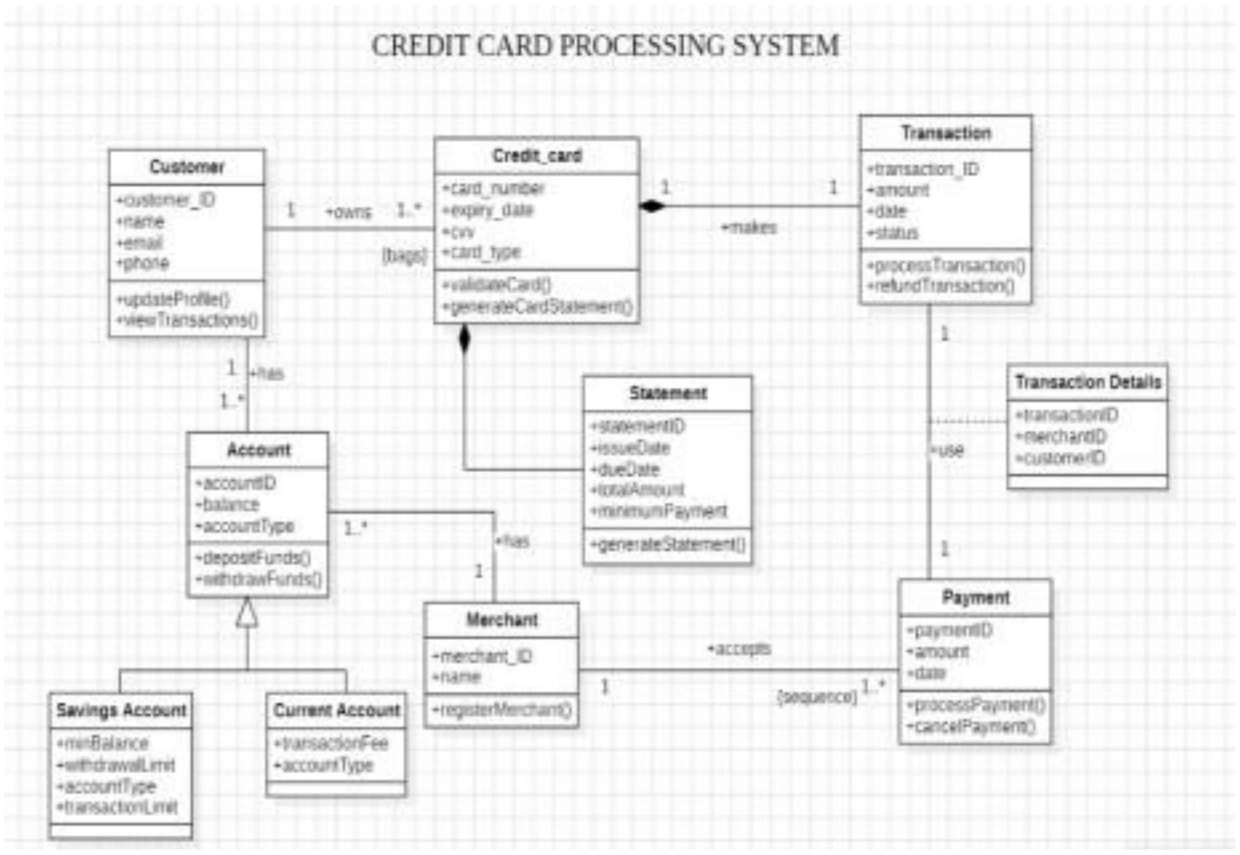


Fig 2.1

Brief Description:

The class diagram outlines the core entities and their relationships:

● Classes:

○ Customer: Attributes include Name, Address, Contact Info, Account Details.

○ CreditCard: Attributes include Card Number, Expiration Date, Credit Limit, Reward Points.

○ Transaction: Attributes include Transaction ID, Date, Amount, Type.

○ Payment: Attributes include Payment ID, Date, Amount, Status.

○ Administrator: Attributes include Admin ID, Name, Privileges.

● Relationships:

○ A Customer owns one or more CreditCards.

○ Each CreditCard is linked to multiple Transactions.

○ A Payment is associated with a CreditCard and recorded by an Administrator.

2.4. State Diagram

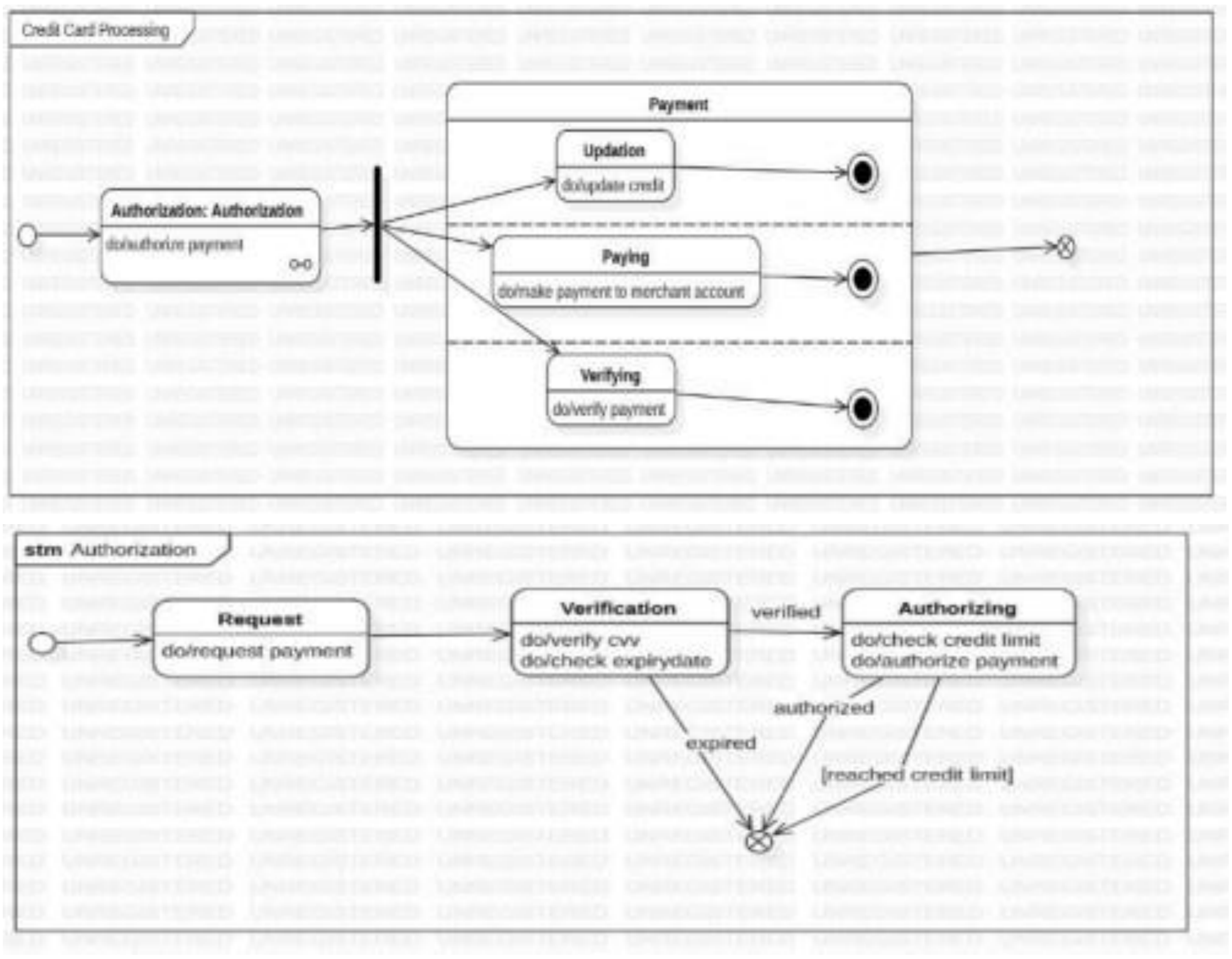


Fig:2.2

Brief Description:

The state diagram illustrates the lifecycle of credit cards and payments:

1. Credit Card States:
○ Application Submitted → Under Review → Approved → Activated → Blocked (if flagged for fraud).
2. Payment States:

○ Payment Pending → Processing → Successful or Failed.
○ Failed payments can retry or result in late fee calculations.

2.5. Use Case Diagram

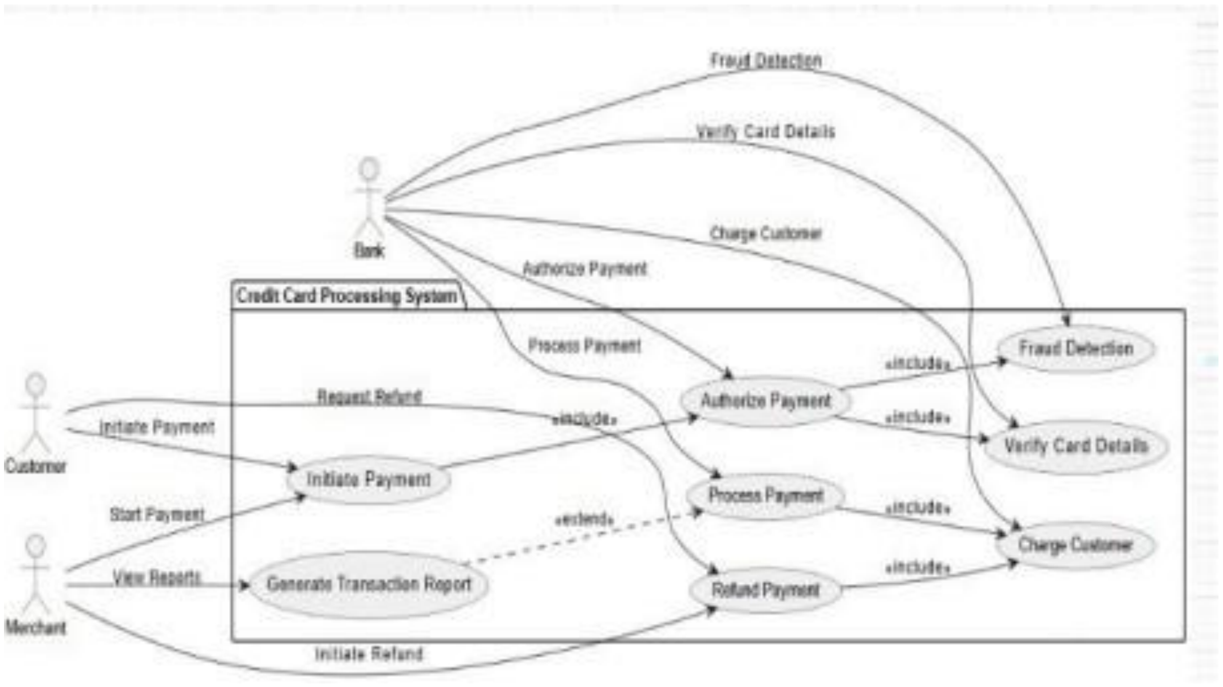


fig:2.3

Brief Description:

The use case diagram identifies the interactions between actors and the system:

● Actors:
○ Customers, Administrators, and Payment Gateways.
● Use Cases:
○ Apply for Credit Card, Approve Application, View Transactions, Make Payment, Generate Statement, Detect Fraud.

2.6. Sequence Diagram

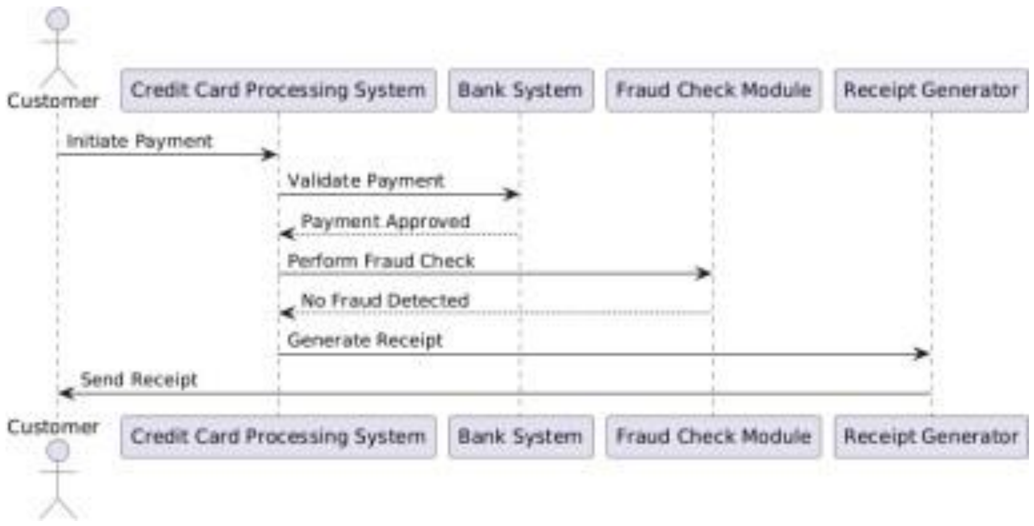


Fig 2.4

Brief Description:

The sequence diagram demonstrates the interaction flow:

● Example: Make Payment

1. Customer selects "Make Payment".
2. System fetches payment details and displays the due amount.
3. Customer confirms the payment.
4. System processes the payment via the payment gateway.
5. Gateway confirms success or failure.
6. System updates the payment status and reflects it in the customer’s account.

2.7. Activity Diagram

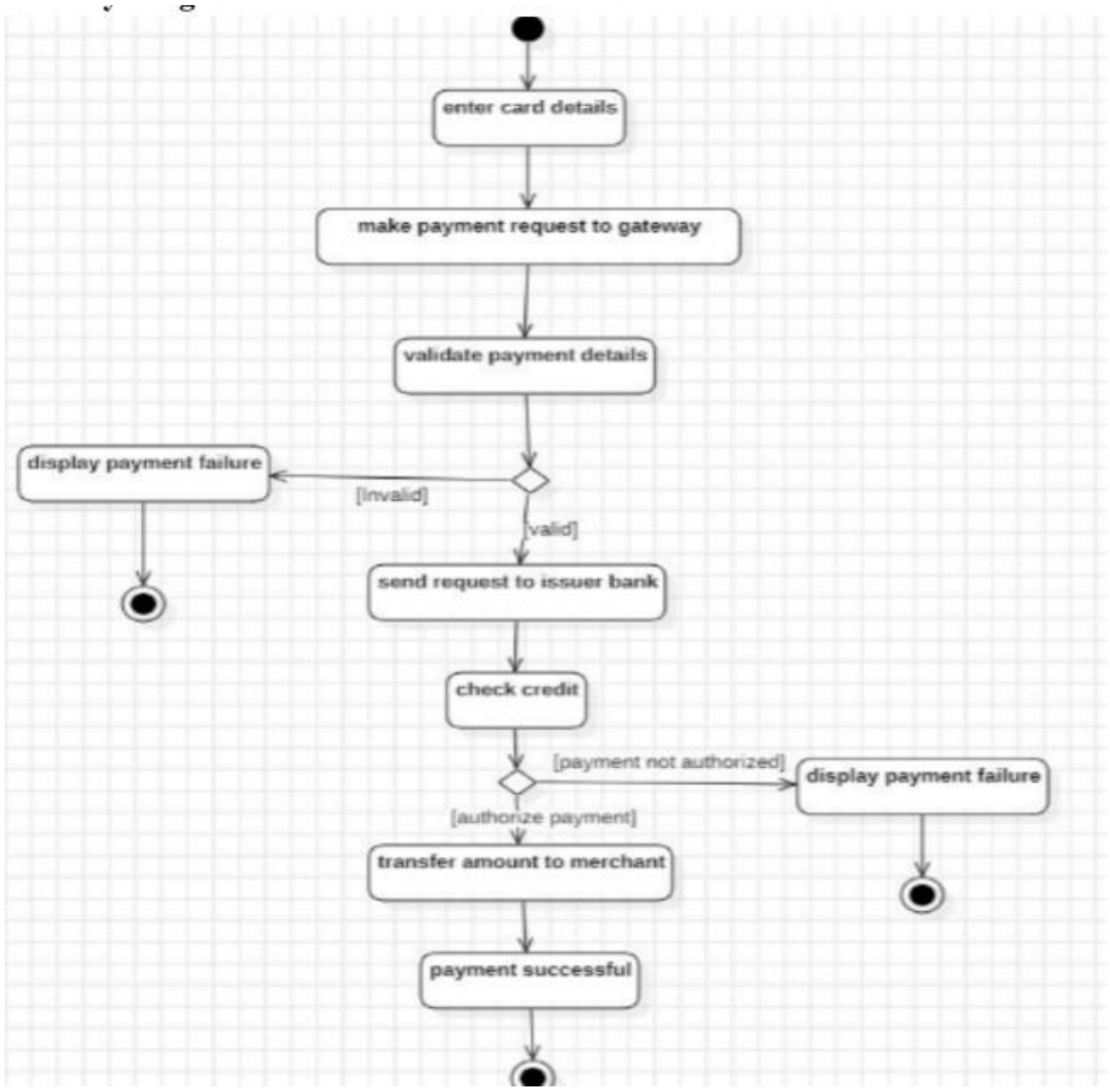


Fig 2.5

Brief Description:

The customer initiates the process by entering card details and receives either a receipt or an error message. The payment system validates the card, processes the transaction, and manages technical aspects, while the bank verifies the account, checks balance, and authorizes payment. The merchant generates a receipt, updates inventory and completes the transaction.

3. Library Management System

3.1. Problem Statement

The Library Management System (LMS) is designed to streamline the management of library resources and operations. It enables users to search for books, issue and return them, track borrowing history, and manage fines, while also providing administrators with tools for cataloging, inventory management, and user account handling.

3.2 SRS- Software Requirement Specification

3.2.1 Purpose of this Document

This document outlines the requirements and specifications for the development of a Library Management System (LMS). It serves as a comprehensive guide for the development team and stakeholders, detailing the system's objectives, functionalities, and constraints to ensure smooth implementation and fulfillment of user needs.

3.2.2 Scope of this Document

The LMS aims to streamline library operations, including cataloging, user management, borrowing, and returns. It will reduce manual workload, improve efficiency, and enhance user satisfaction. The system will provide value by ensuring accurate record-keeping and real-time updates. The estimated development timeline is three months with a projected budget of $20,000.

3.2.3 Overview

The Library Management System will be a web-based application allowing library staff to manage books, users, and transactions efficiently. Users can search for books, view availability, and borrow or reserve them. Staff can update inventory, manage fines, and generate reports. The system will be user-friendly, scalable, and secure, ensuring robust library operations.

3.2.4. General Description

The LMS will automate the management of library resources and user transactions. It will allow users to search and borrow books, while staff can manage inventory and track transactions. Key features include user registration, book cataloging, borrowing/return management, fine calculation, and report generation. The system is crucial for libraries aiming to improve service delivery and operational efficiency.

User Characteristics:

- **Students/Readers**: Basic knowledge of web navigation to search and reserve books.
- **Library Staff**: Familiarity with library processes and inventory management.
- **Administrators**: Skilled in data analysis and report generation for decision-making.

3.2.5. Functional Requirements

1. **User Registration and Login**: Users must be able to register, log in, and update profiles.
2. **Book Catalog Management**: Staff can add, delete, or update book records.
3. **Search and Reservation**: Users can search for books by title, author, or category and reserve them if available.
4. **Borrow and Return Management**: Tracks books borrowed and returned, updating availability status.
5. **Fine Calculation**: Automatically calculates fines for overdue books.
6. **Report Generation**: Admins can generate reports on inventory, overdue books, and user activity.

3.2.6. Interface Requirements

- **User Interface**: A web-based interface for users and staff, optimized for desktops and mobile devices.
- **Database Interface**: Seamless integration with the backend database for managing book records and user data.
- **API Integration**: Compatibility with external systems for online book purchase and supplier management.

3.2.7. Performance Requirements

- **Response Time**: Search queries should return results within 2 seconds.
- **Concurrent Users**: System must support up to 200 concurrent users without performance degradation.
- **Error Rate**: Maximum acceptable error rate is 0.1% for transaction processing.

3.2.8. Design Constraints

- The system must use a relational database like MySQL or PostgreSQL.
- Web development must leverage modern frameworks such as React or Angular.
- The application must be compatible with commonly used browsers like Chrome, Firefox, and Safari.

3.2.9. Non-Functional Attributes

- **Security**: Role-based access control and data encryption for sensitive information.
- **Reliability**: 99.9% uptime guaranteed with fallback mechanisms for system failures.
- **Scalability**: Ability to handle growing user base and inventory.
- **Data Integrity**: Robust validation to prevent duplicate or incorrect entries.

3.2.10. Preliminary Schedule and Budget

- **Schedule**:
  - Requirement Gathering: 2 weeks
  - Design and Development: 8 weeks
  - Testing and Deployment: 4 weeks
- **Budget**:
  - Development Costs: $15,000
  - Testing and QA: $3,000
  - Hosting and Maintenance: $2,000
  - **Total**: $20,000

3.3 Class Diagram:

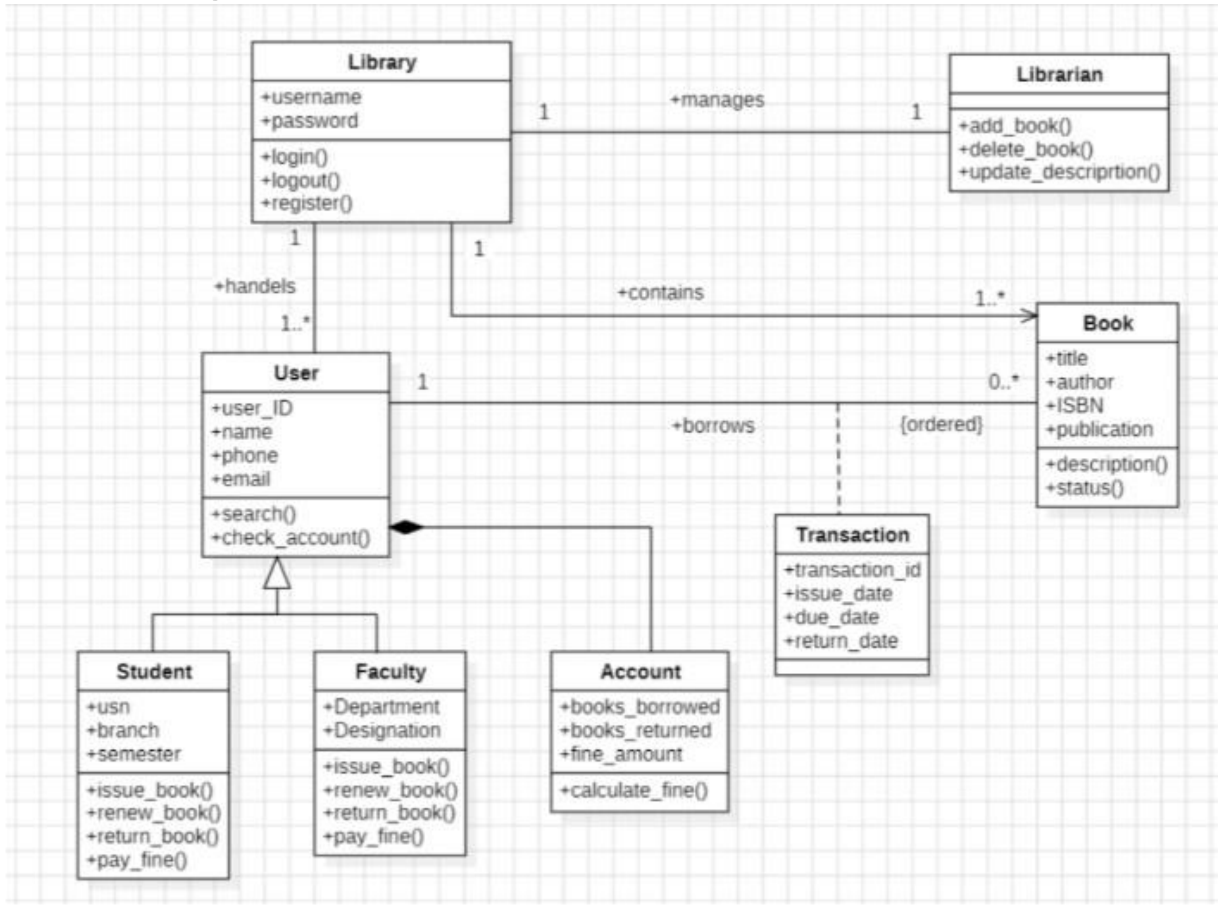


Fig 3.1

Brief Description:

Classes:

● Book: Attributes include Title, Author, Genre, ISBN, Availability.

● User: Attributes include Name, User ID, Contact Info, Borrowing History.

● Transaction: Attributes include Transaction ID, Book ID, User ID, Issue Date, Return Date, Fine.

● Administrator: Attributes include Admin ID, Name, Privileges.

Relationships:

● A User can borrow multiple Books.

● Each Transaction links a Book and a User.

● An Administrator manages the Book inventory.

3.4 State Diagram:

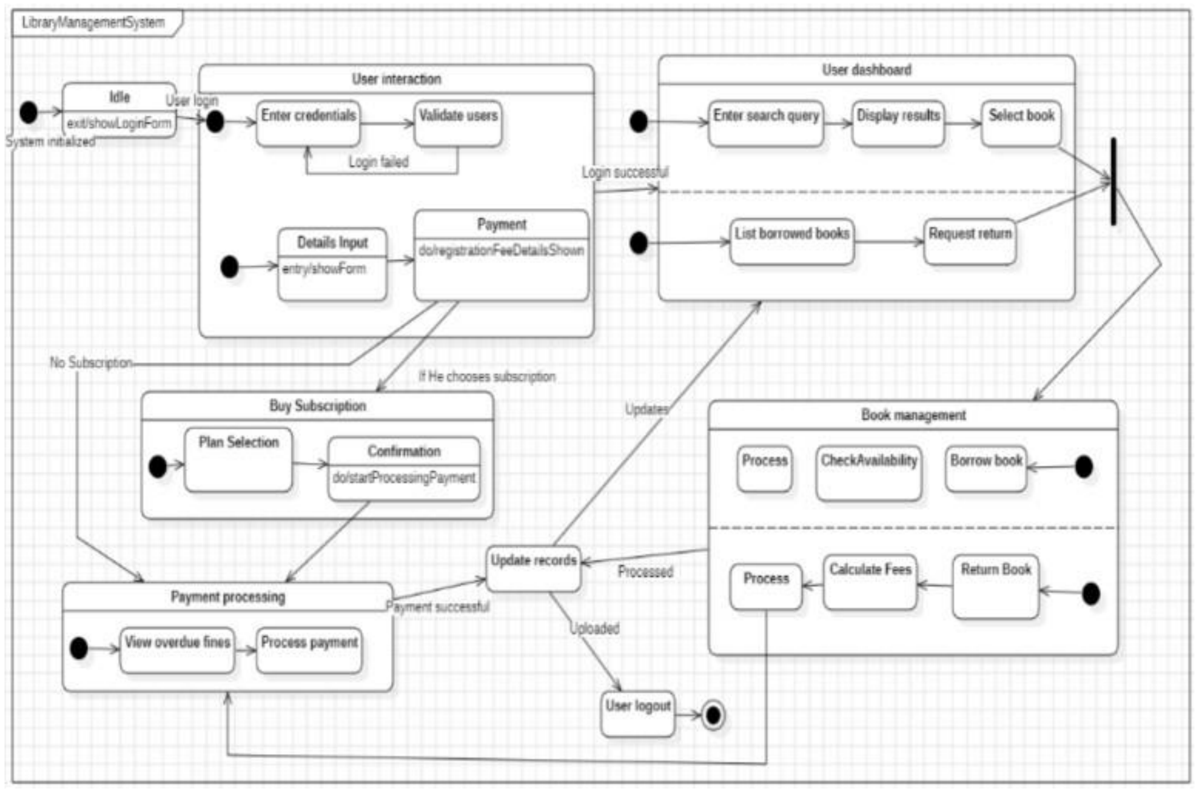


Fig 3.2

Brief Description:

● Book Available: The book is in the library.

● Book Issued: The book is borrowed by a user.

● Book Returned: The book is returned and updated in the inventory.

● Overdue Fine Imposed: If the book is not returned on time, a fine is applied.

3.5 Use Case Diagram:

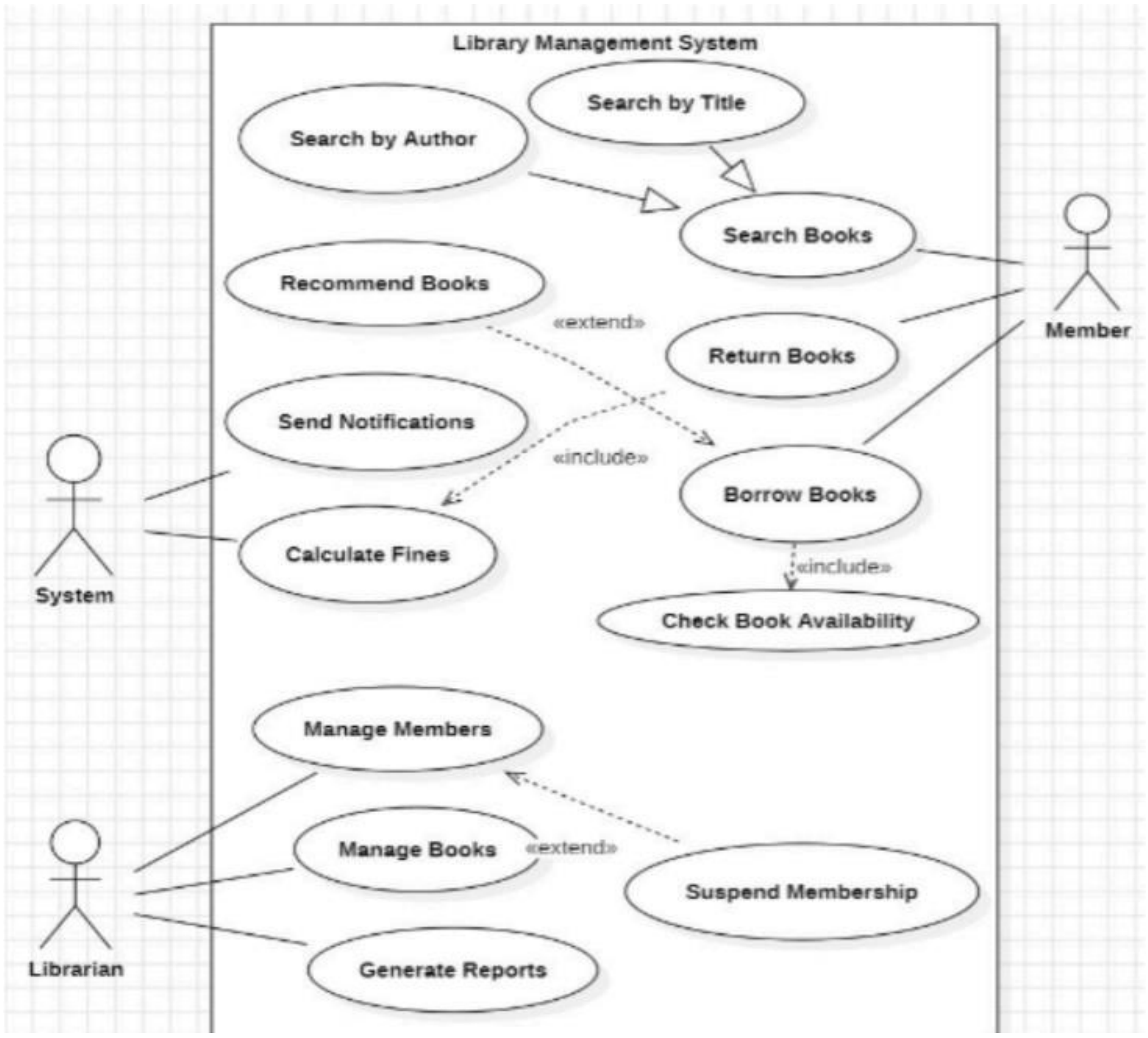


Fig 3.3

Brief Description:

Actors: ● Users, Administrators.

Use Cases:● Search Book, Borrow Book, Return Book, Pay Fine, Add/Remove Book,

Generate Reports.

3.6 Sequence Diagram:

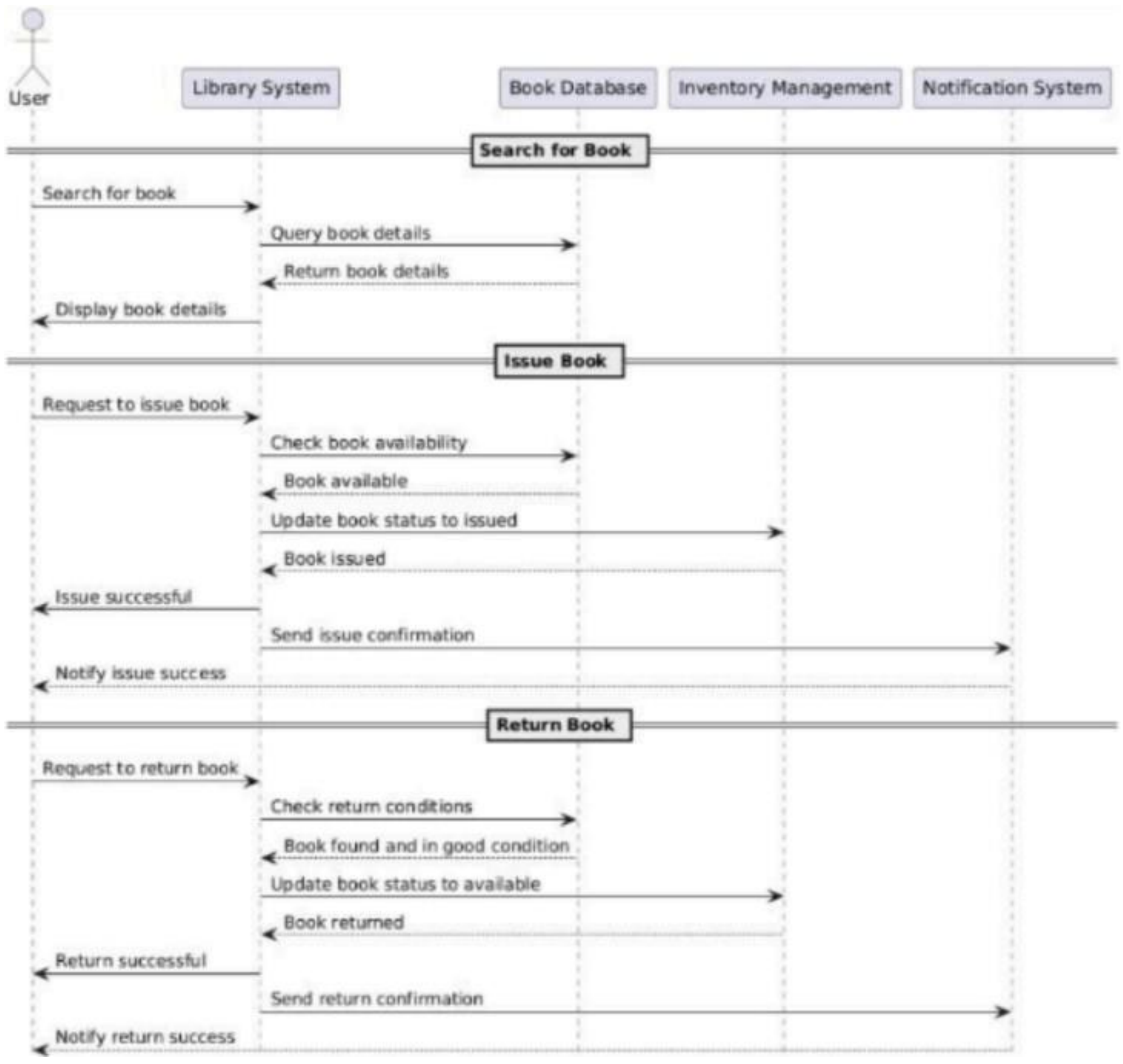


Fig 3.4

The sequence diagram illustrates the process of borrowing a book:

1. User searches for a book

.2. System displays available books

.3. User requests to borrow a book.

4. System checks availability and issues the book.

5. System updates the user’s borrowing history.

3.7 Activity Diagram:

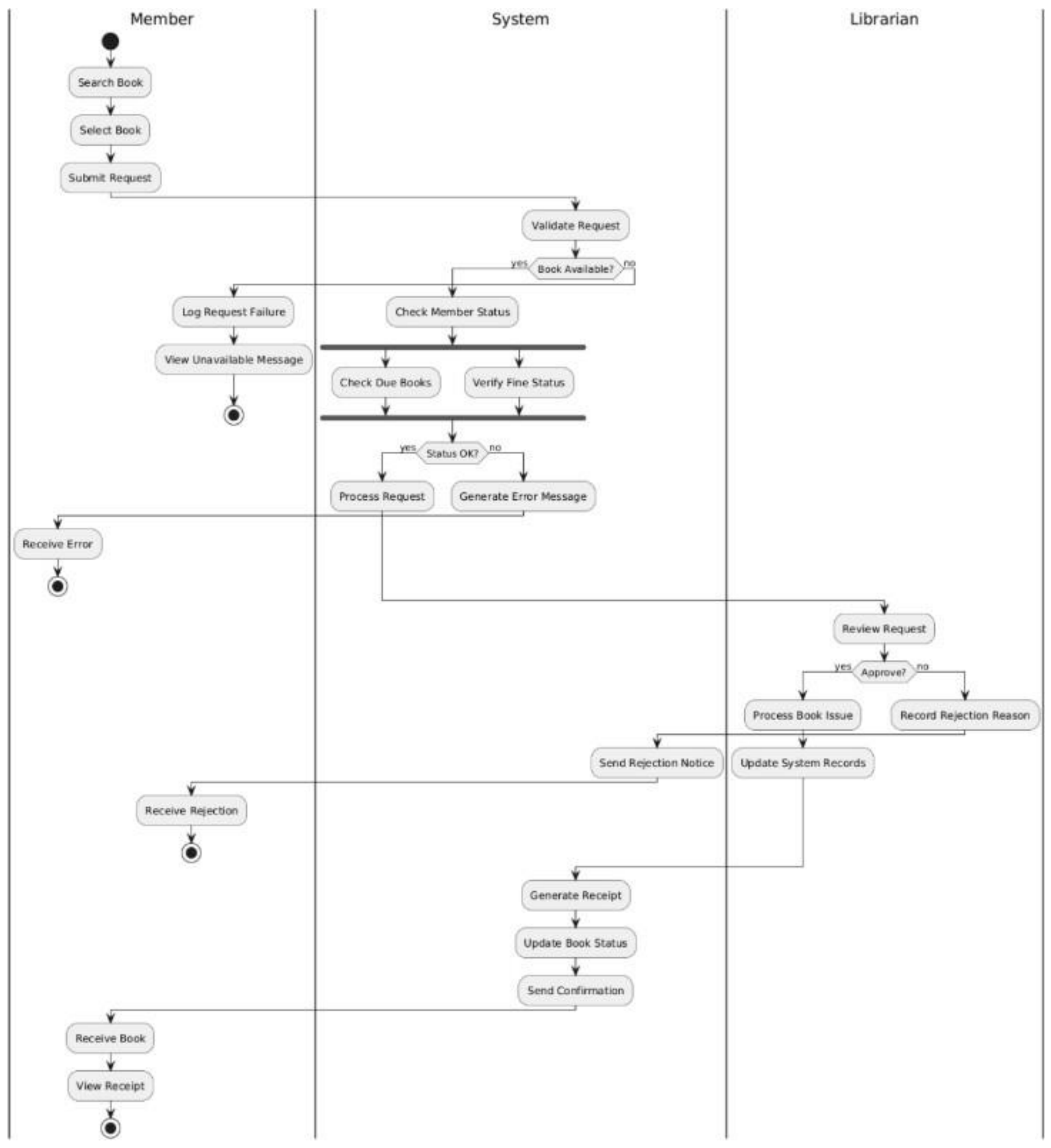


Fig 3.5

The advanced activity diagram includes three swimlanes—Member, System, and Librarian—showing clear responsibility for actions and interactions. It introduces parallel processes, detailed error handling, and staff oversight while ensuring comprehensive tracking, status updates, and a more detailed workflow compared to the simple diagram.

Brief Description:

The activity diagram describes the workflow of returning a book:

1. User returns a borrowed book.

2. System checks the return date.

3. If overdue, calculate the fine and notify the user.

4. Update the book status to "Available".

4. Stock Maintenance System

4.1. Problem Statement

The Stock Maintenance System (SMS) is designed to efficiently track and manage inventory for businesses. It ensures real-time updates on stock availability, monitors stock levels, automates reorder processes, and provides detailed reporting to minimize inventory-related inefficiencies.

4.2 SRS- Software Requirement Specification

4.2.1 Purpose of this Document:

The purpose of this document is to define the requirements for the Stock Maintenance System. It provides a comprehensive guide for developers, stakeholders, and users to understand the purpose, functionalities, and constraints of the system. The document ensures clarity on what the system aims to achieve and serves as a reference throughout the development lifecycle.

4.2.2 Scope of this Document:

The Stock Maintenance System is designed to automate the management of inventory for businesses. It will enable tracking stock levels, updating inventory, managing stock in/out operations, generating reports, and alerting users of low-stock scenarios. The system will reduce manual errors, improve efficiency, and save time. The development is estimated to cost approximately $10,000 and will take three months to complete.

4.2.3 Overview:

The Stock Maintenance System will consist of modules for stock entry, stock updates, order processing, report generation, and user management. The system will provide real-time data on stock availability, support multi-user access, and ensure accuracy through automated operations. The product will primarily cater to small and medium businesses.

4.2.4. General Description

The Stock Maintenance System aims to streamline inventory management by automating stock monitoring and reporting. Users can efficiently manage stock details, access inventory reports, and receive alerts for stock replenishment. The user community includes business owners, inventory managers, and warehouse staff. Features like real-time tracking and automated notifications ensure timely decision-making and operational efficiency.

4.2.5. Functional Requirements

1. Stock Entry Module: Enables users to add new stock details, including item name, quantity, and category.

2. Stock Update Module: Updates stock levels in real-time based on stock in/out operations.

3. Low Stock Alerts: Sends notifications for items reaching predefined minimum thresholds.

4. Report Generation: Generates detailed reports on stock availability, usage patterns, and order history.

5. User Management: Provides access control for different user roles (e.g., admin, inventory manager).

6. Order Processing: Manages stock deduction for customer orders and updates inventory.

4.2.6. Interface Requirements

The system will interact with users via a graphical user interface (GUI) with clear navigation menus. It will also support integration with accounting and sales systems via shared data streams for seamless synchronization. Data communication will be in

XML or JSON formats for compatibility.

4.2.7. Performance Requirements

● The system should handle up to 10,000 stock items without performance degradation.

● Stock updates must be processed in under 2 seconds.

● Report generation should not exceed 5 seconds.

● Error rate for calculations should not exceed 0.1%.

4.2.8. Design Constraints

● The system must use a MySQL database for backend storage.

● It should run on Windows and Linux operating systems.

● The design adheres industry-standard encryption algorithms to ensure data security.

● Limited budget and resources restrict the use of advanced AI features.

4.2.9. Non-Functional Attributes

● **Security:** Ensures data encryption during transmission and storage.

● **Reliability:** 99.9% uptime guarantee to ensure continuous system availability.

● **Scalability:** Capable of supporting an increased number of users and stock items.

● **Portability:** Can be deployed on both desktop and cloud platforms.

● **Data Integrity:** Prevents unauthorized data modification through role-based access controls.

4.2.10. Preliminary Schedule and Budget

● **Schedule:**

- Requirement Analysis: 2 weeks
- System Design: 4 weeks
- Development: 6 weeks
- Testing: 2 weeks
- Deployment and Training: 2 weeks

● **Budget:**

- Development Cost: $8,000
- Testing and Deployment: $2,000
- Total: $10,000

4.3 Class Diagram:

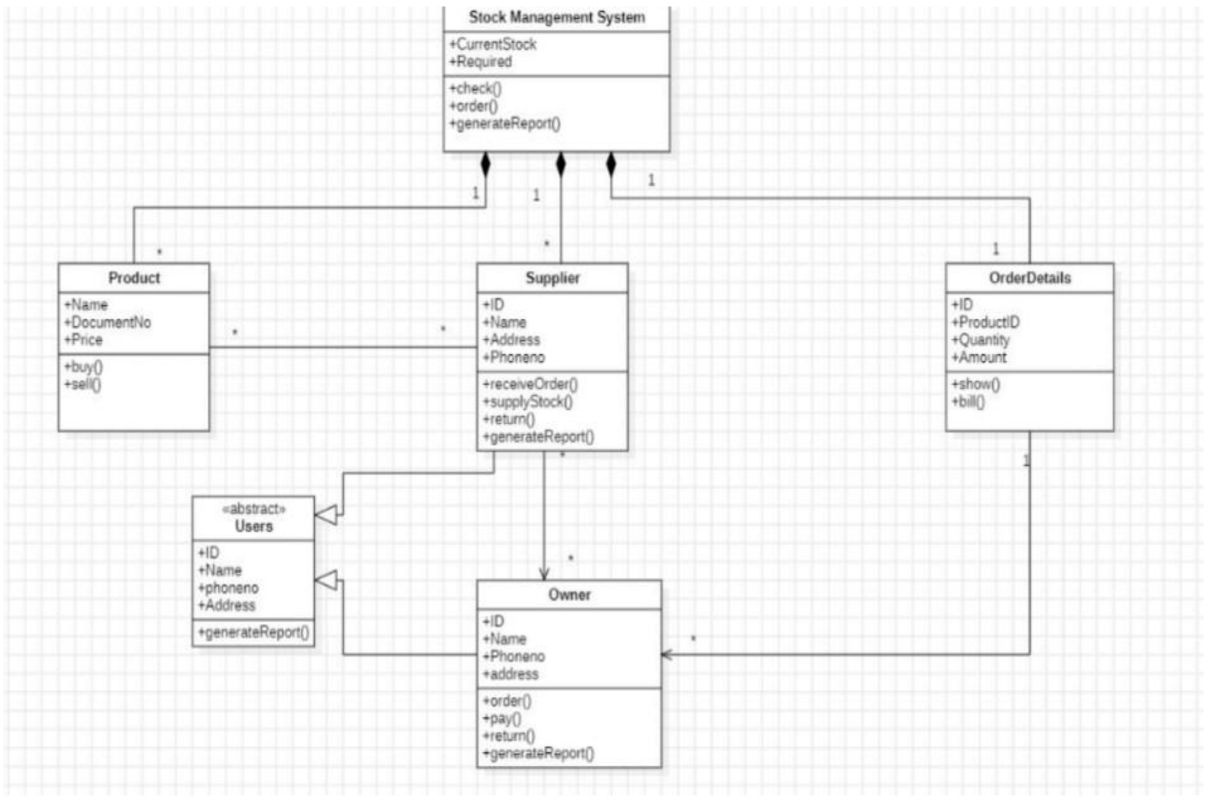


Fig 4.1

Brief Description:

The class diagram includes the following entities:

● Classes:

○ Product: Attributes include Product ID, Name, Category, Price, Stock Quantity.

○ StockTransaction: Attributes include Transaction ID, Product ID, Quantity, Type (Sale/Purchase), Date.

○ Administrator: Attributes include Admin ID, Name, Contact.

○ Supplier: Attributes include Supplier ID, Name, Contact Info.

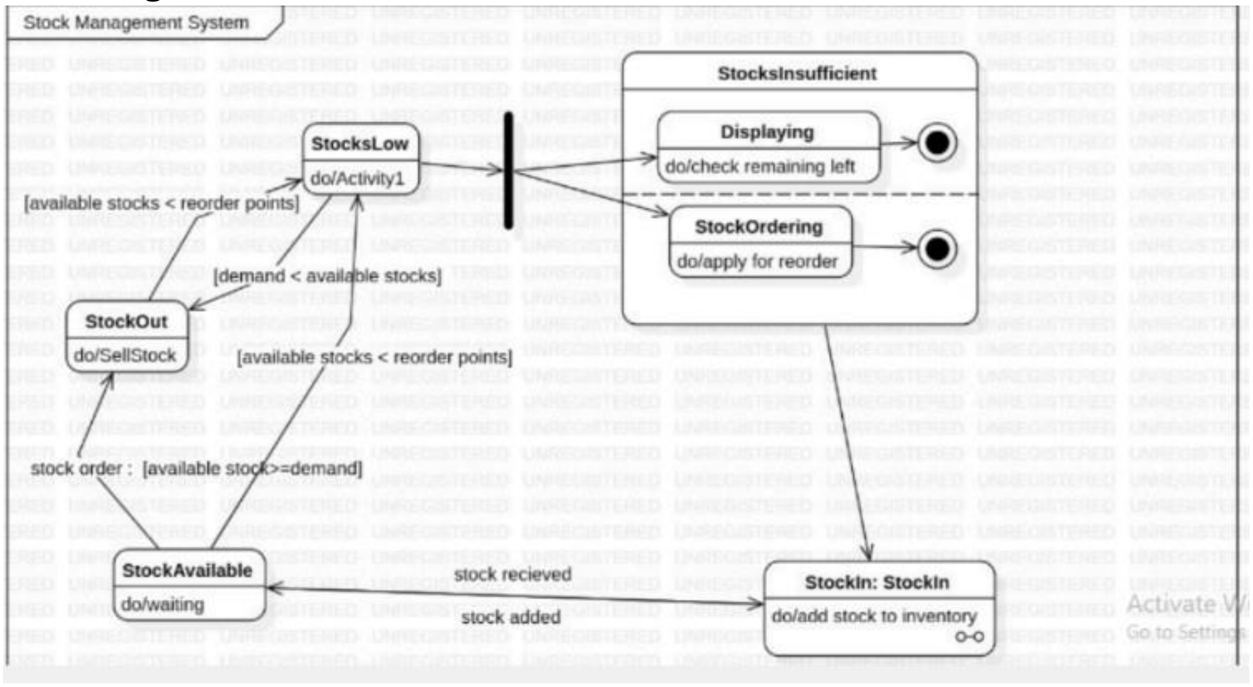
● Relationships:

○ A Supplier supplies multiple Products.

○ StockTransaction records are linked to Products.

○ An Administrator manages Products and Transactions.

State Diagram:



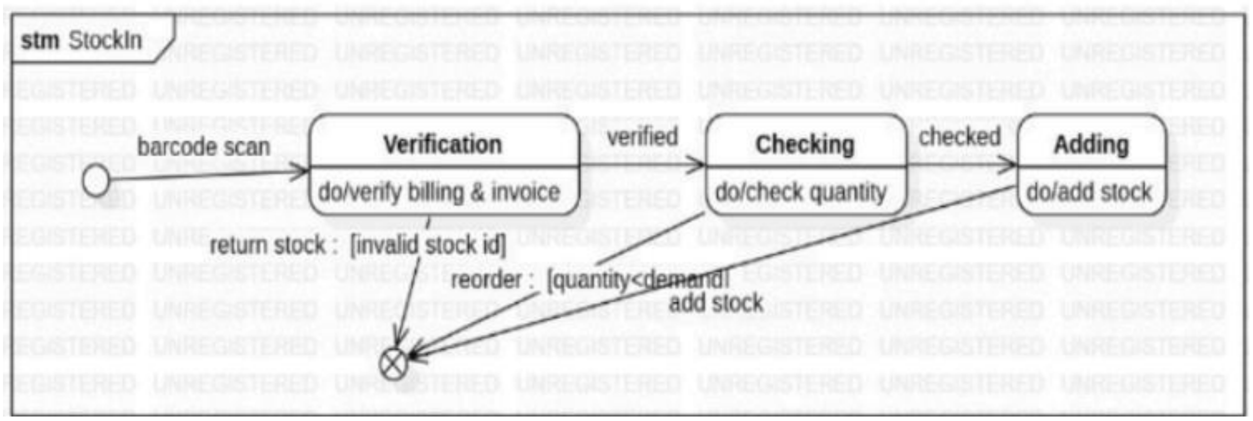


Fig 4.2

Brief Description :

The state diagram shows the lifecycle of stock:

● States:

○ Stock Added: New stock is recorded in the inventory.

○ Stock Updated: Stock levels are updated after sales or returns.

○ Stock Low: Stock falls below the reorder threshold.

○ Reorder Processed: New stock is ordered and added to the inventory.

4.5 Use Case Diagram:

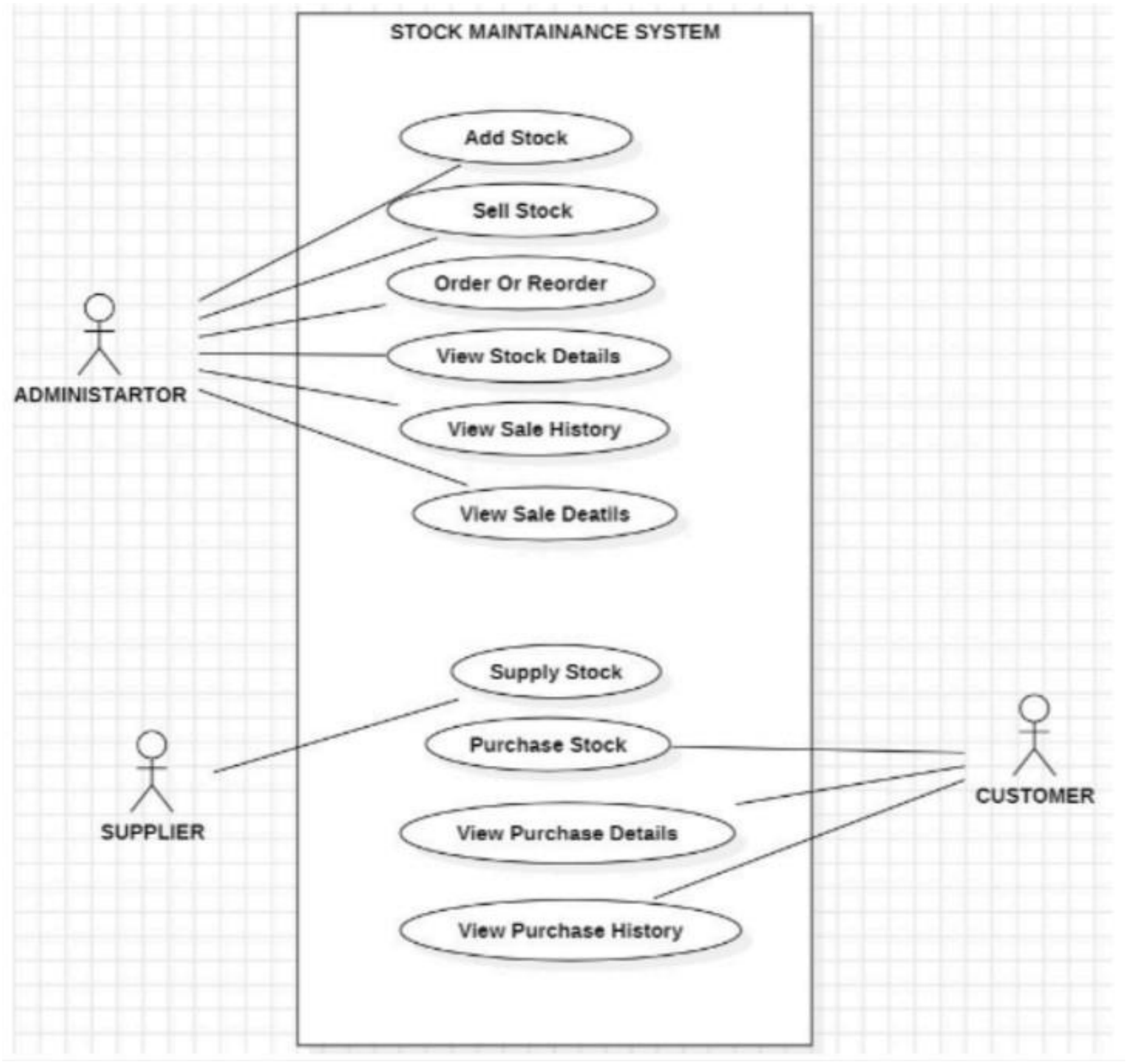


Fig 4.3

Brief Description:

The use case diagram highlights key interactions:

● Actors: ○ Administrator, Staff, Supplier.

● Use Cases:

○ Add/Update Product, Track Stock, Generate Reorder Alert, Record Sale, Generate Reports, Manage Suppliers.

4.6 Sequence Diagram:

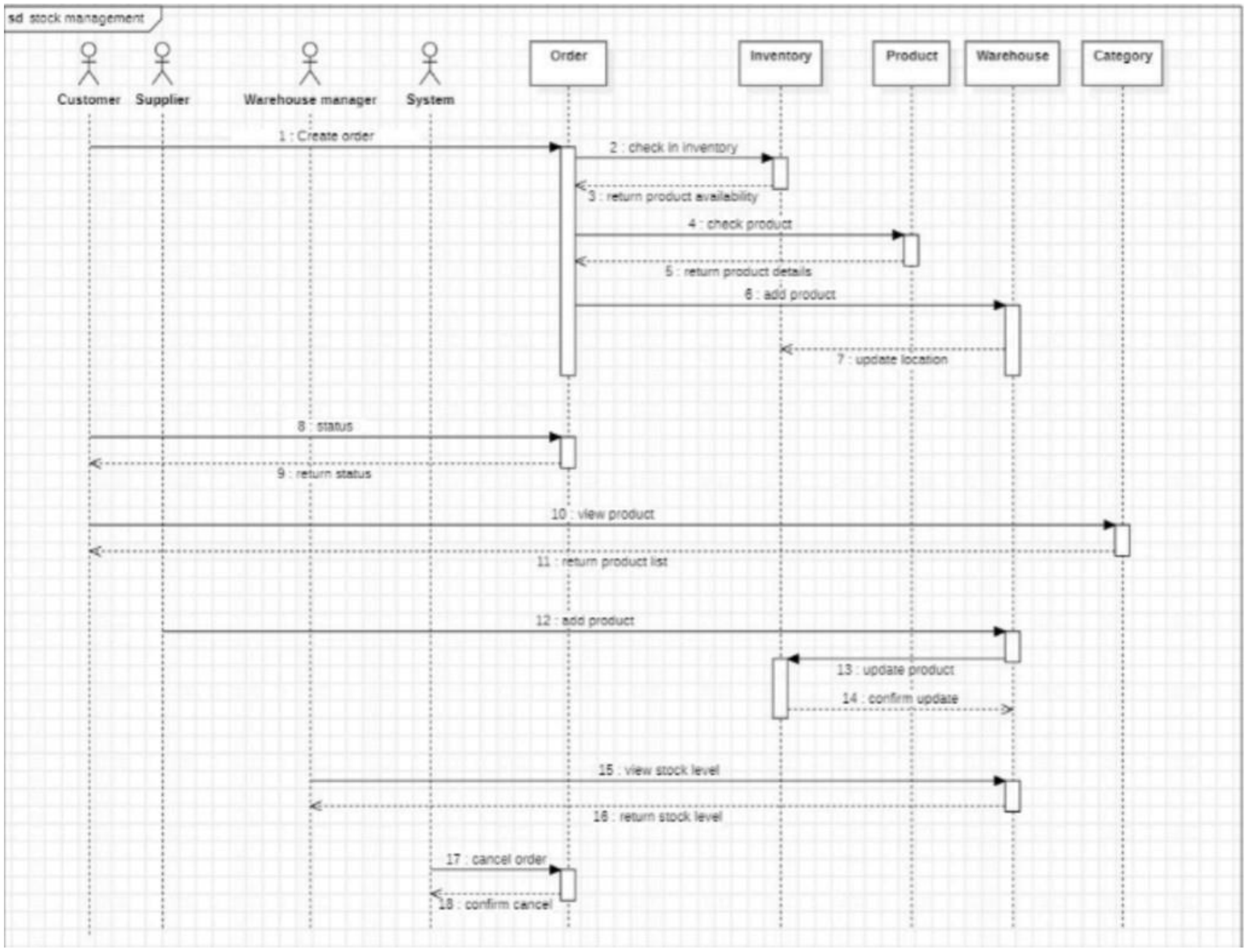


Fig 4.4

Brief Description:

The sequence diagram illustrates the process of :1. Staff scans the product barcode.2. The system fetches product details from the database.3. Staff enters the quantity sold.4. The system deducts the quantity from stock and updates the database.5. The system generates a transaction record and updates sales data.

4.7 Activity Diagram:

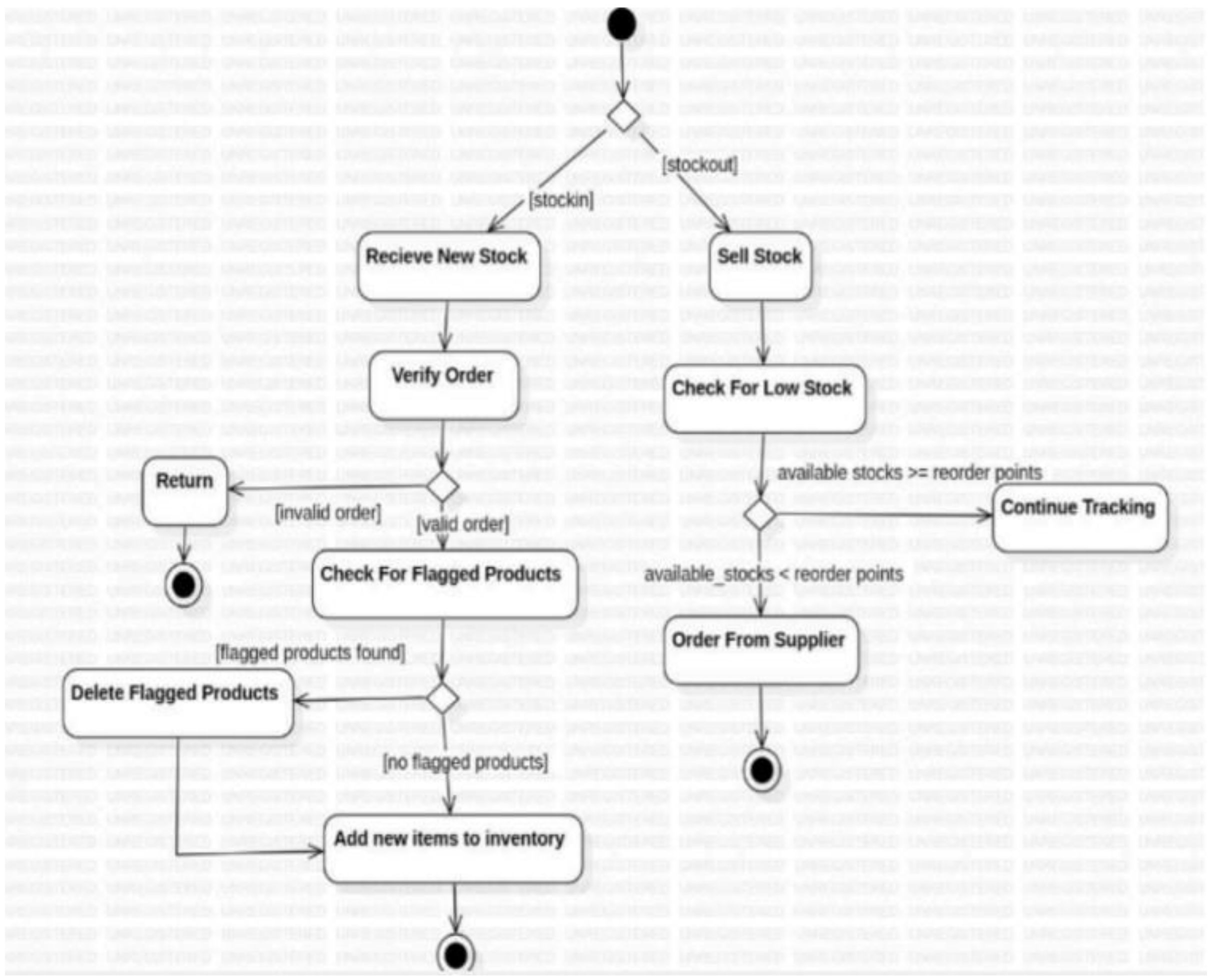


Fig 4.5

Brief Description:

The activity diagram describes the workflow of restocking:

1. Administrator receives a low-stock alert.

2. System suggests reorder quantity based on past sales trends.

3. Administrator places an order with the supplier.

4. Stock is delivered and updated in the system.

5. The system updates stock levels and sends confirmation.

5. Passport Automation System

5.1. Problem Statement

The Passport Automation System (PAS) is designed to simplify and streamline the process of passport application, verification, issuance, and renewal. It provides an online platform for applicants to submit their details, track application progress,and receive updates, while enabling officials to manage and verify applications efficiently.

5.2 SRS- Software Requirement Specification

5.2.1 Purpose of this Document

The purpose of this document is to outline the requirements for a Passport Automation System (PAS) that streamlines the process of applying for, processing, and issuing passports. This document serves as a guide for developers, testers, and stakeholders, ensuring clear understanding and implementation of the system.

5.2.2 Scope of this Document

The Passport Automation System is designed to automate and simplify the workflow of passport management. Applicants can apply, schedule appointments, and track their application status online. Passport officers can verify documents, process applications, and approve or reject them. The system reduces manual intervention, improves efficiency, and ensures accuracy. Development is estimated to take 6 months, with a cost projection of $50,000.

5.2.3 Overview

The Passport Automation System will consist of three main modules:

1. **Applicant Module:** Allows users to register, submit applications, schedule appointments, and track statuses.

2. **Officer Module:** Enables verification of applications, document validation, and status updates.

3. **Administration Module:** Provides tools for managing users, reports, and system configurations.

5.2.4. General Description

Functions of the System

● Online application submission.

● Document verification and appointment scheduling.

● Real-time application tracking for applicants.

● Automated communication via email/SMS for status updates.

● Secure storage and management of applicant data.

User Characteristics

● **Applicants:** Must be able to use the system with basic computer knowledge .

● **Passport Officers:** Require training to use the verification and approval tools.

● **Administrators:** Should have expertise in managing technical and operational workflows.

Benefits

● Reduced manual workload and processing time.

● Improved accuracy and security in document handling.

● Transparent and user-friendly process for applicants.

5.2.5. Functional Requirements

● Applicants can create accounts and submit applications.

● The system validates submitted data and documents.

● Automatic generation of unique application IDs.

● Passport officers can approve/reject applications.

● Automated scheduling of appointments based on availability.

● Status updates sent to applicants via email/SMS.

5.2.6. Interface Requirements

User Interface:

● A responsive web portal for applicants to submit applications and track statuses.

● A dashboard for officers to manage and process applications.

4.2 Hardware Interface:

● Scanners for document validation.

● Servers for data storage and processing.

4.3 Software Interface:

● Integration with government databases for identity verification.

● Email and SMS APIs for notifications.

5.2.7. Performance Requirements

● The system should handle up to 10,000 simultaneous users.

● Document verification must be completed within 5 seconds per upload.

● Status updates should be sent within 1 minute of an event.

5.2.8. Design Constraints

● The system must comply with government data protection laws.

● Hardware should support scalability for increased demand.

● Use of specific algorithms for document validation and security.

5.2.9. Non-Functional Attributes

● **Security:** Data encryption for sensitive information.

● **Scalability:** The system should handle increased loads during peak periods. ●

Reliability: 99.9% uptime for uninterrupted service.

● **Usability:** Intuitive design for ease of use by all user types.

5.2.10. Preliminary Schedule and Budget

Schedule:

- **Requirement Gathering and Analysis:** 1 month
- **Design:** 1 month
- **Development:** 3 months
- **Testing:** 1 month
- **Deployment and Training:** 1 month

Budget:

- **Development Cost:** $30,000
- **Infrastructure Cost:** $10,000
- **Training and Deployment Cost:** $10,000

5.3 Class Diagram:

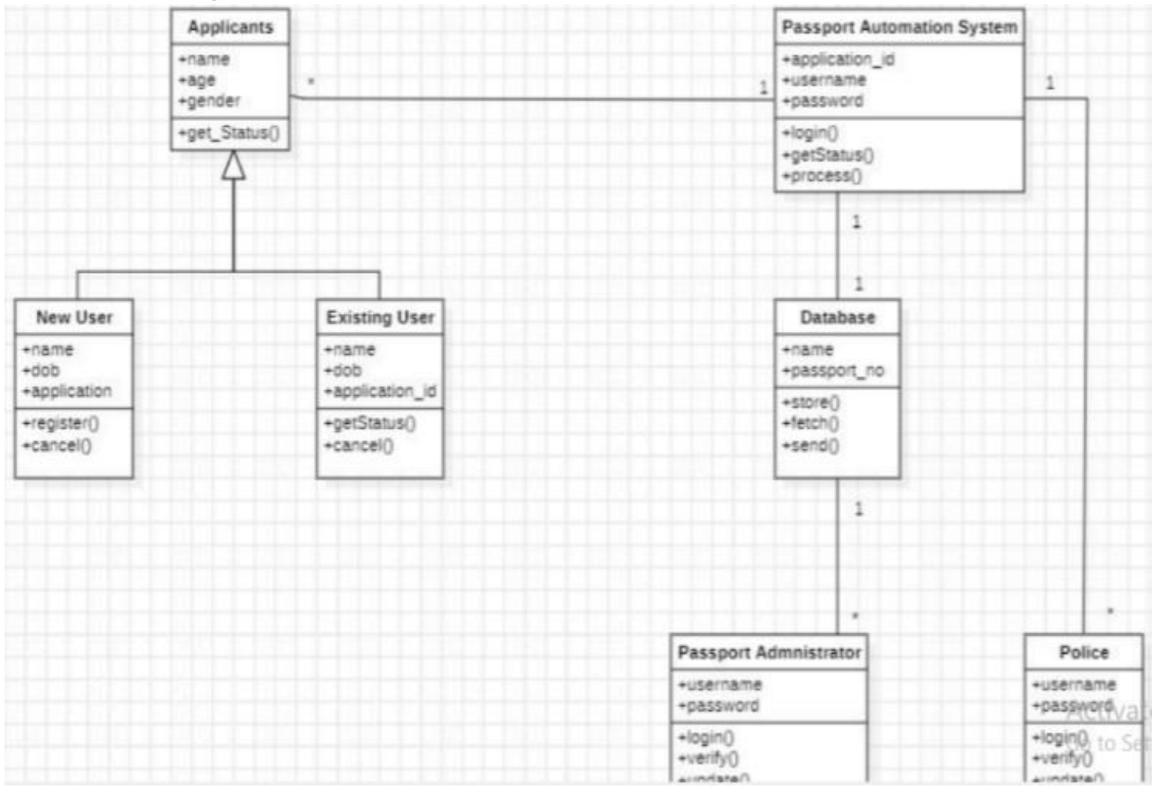


Fig 5.1

Brief Description:

The class diagram includes the following entities:

● Classes:

○ Applicant: Attributes include Applicant ID, Name, Address, Contact, Documents.

○ Application: Attributes include Application ID, Applicant ID, Submission Date,

Status.

○ Official: Attributes include Official ID, Name, Role.

○ Payment: Attributes include Payment ID, Amount, Status, Date.

● Relationships:

○ An Applicant submits an Application.

○ An Official verifies and updates Application statuses. Each Application is linked to a Payment.

5.4 State Diagram:

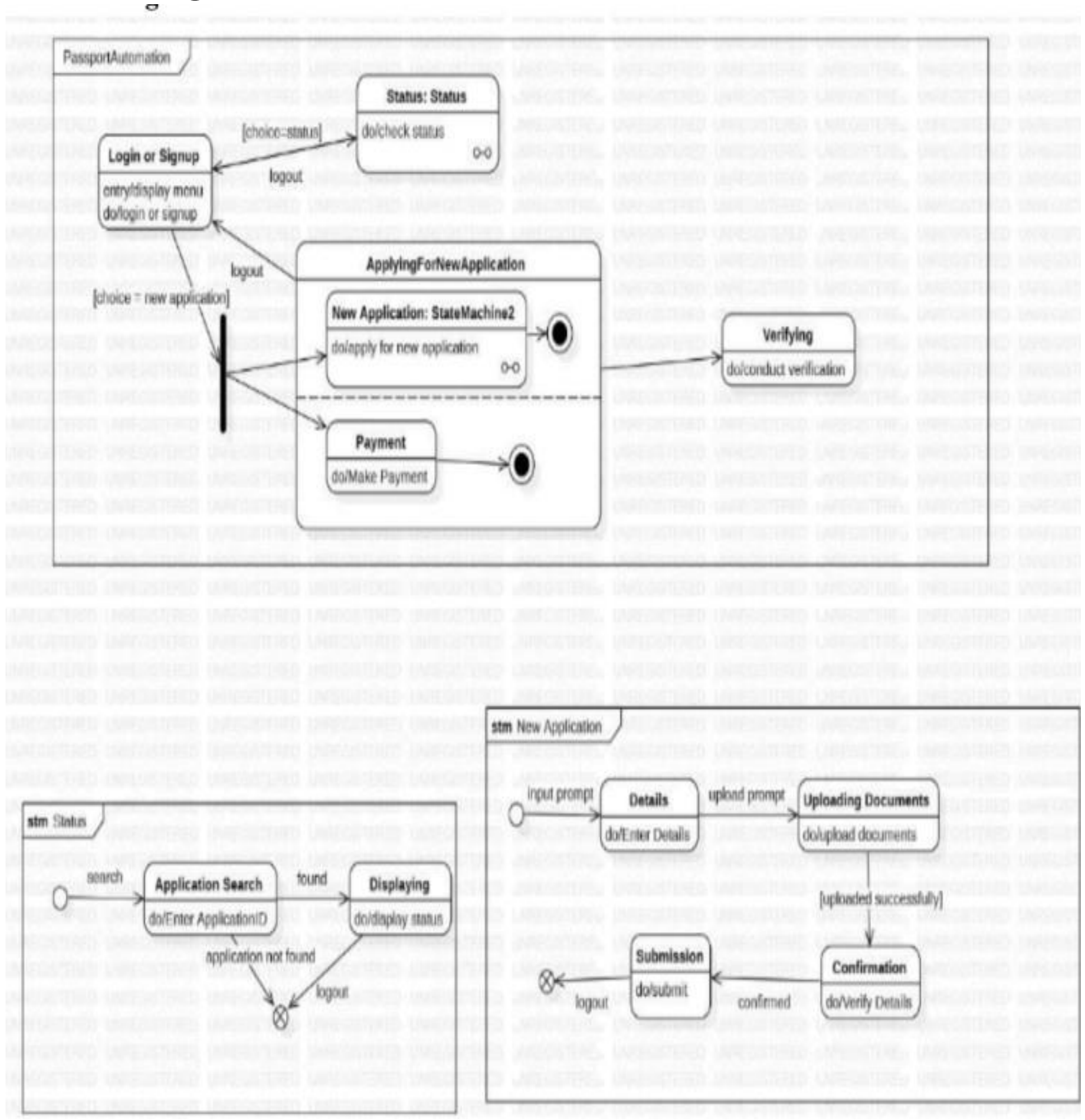


Fig 5.2

Brief Description:

The state diagram shows the lifecycle of a passport application:

● States:

○ Application Submitted: The application is submitted by the applicant.

○ Document Verification: Officials verify the submitted documents.

○ Police Clearance: The application undergoes police verificati**on.**

5.5 Use case diagram:

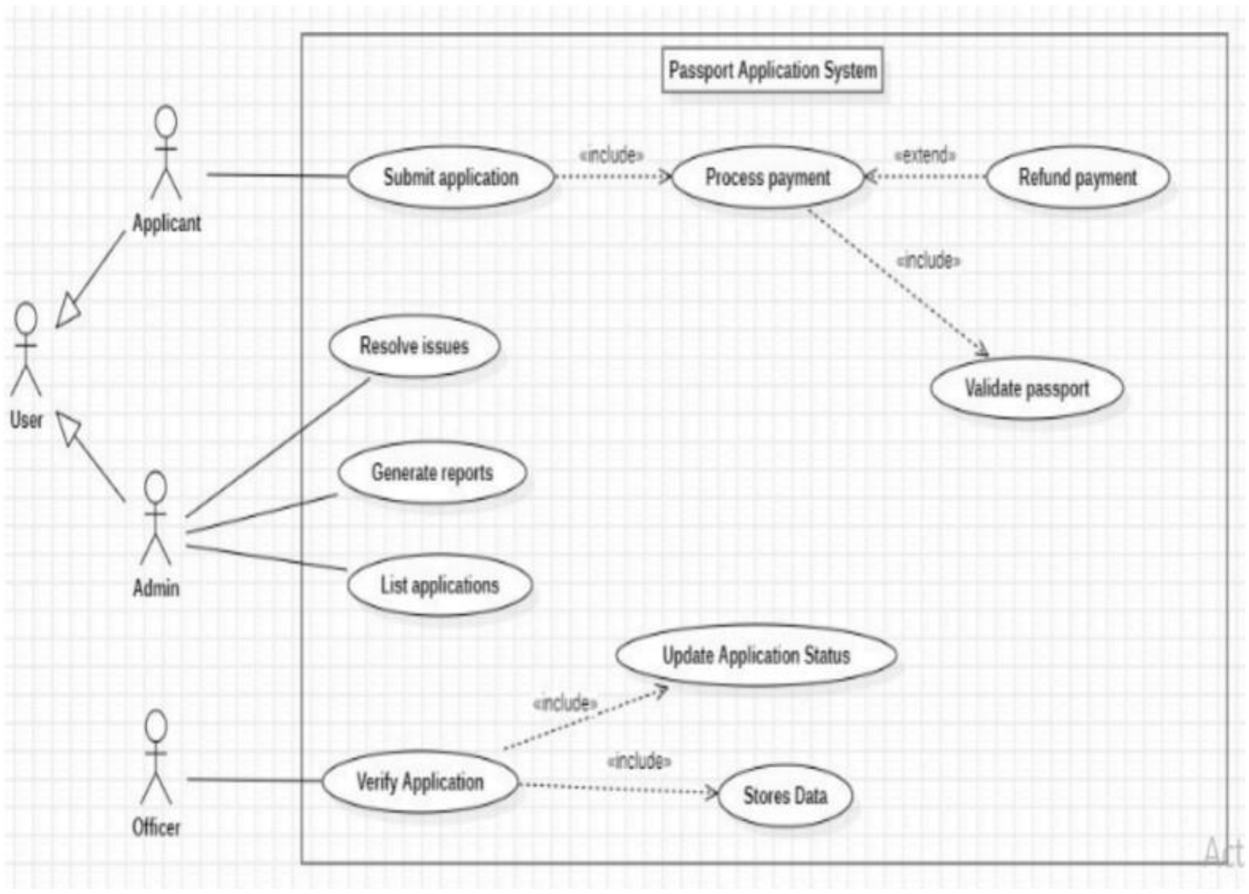


Fig 5.3

Brief Description:

The use case diagram highlights key interactions:
● Actors:
○ Applicants, Officials, Police Department.
● Use Cases:
○ Submit Application, Upload Documents, Pay Fees, Track Status, Verify Documents, Manage Police Clearance, Issue Passport.

5.6 Sequence Diagrams:

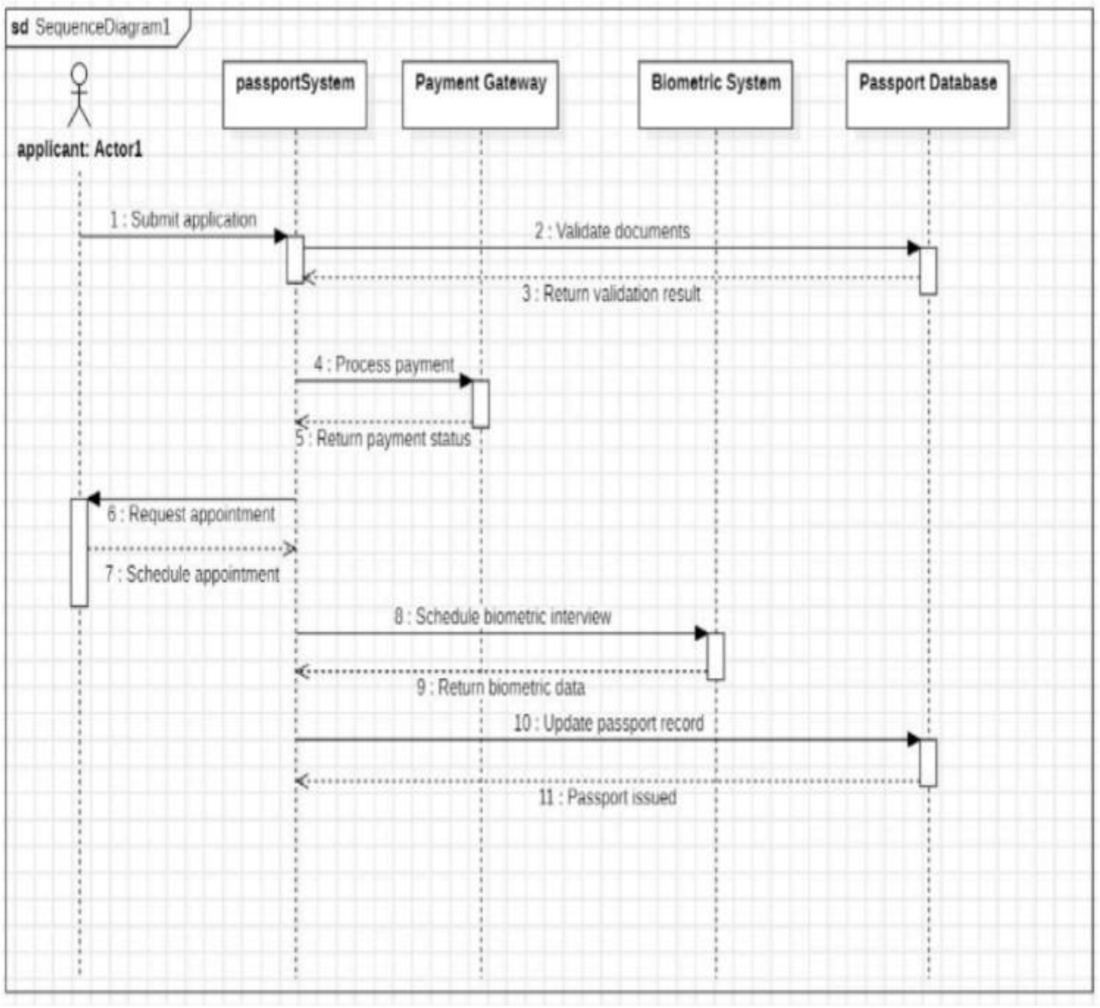


Fig 5.4

Brief Description:

The sequence diagram illustrates the process of applying for a passport:1. Applicant registers and logs in to the system.2. Applicants fill out the application form and upload documents.3. System validates and stores the data.4. Applicants pay the application fee.5. System notifies officials for document verification.

6. Officials verify documents and initiate police clearance.7. After clearance, the system updates the status to "Passport Issued".

5.7 Activity Diagram:

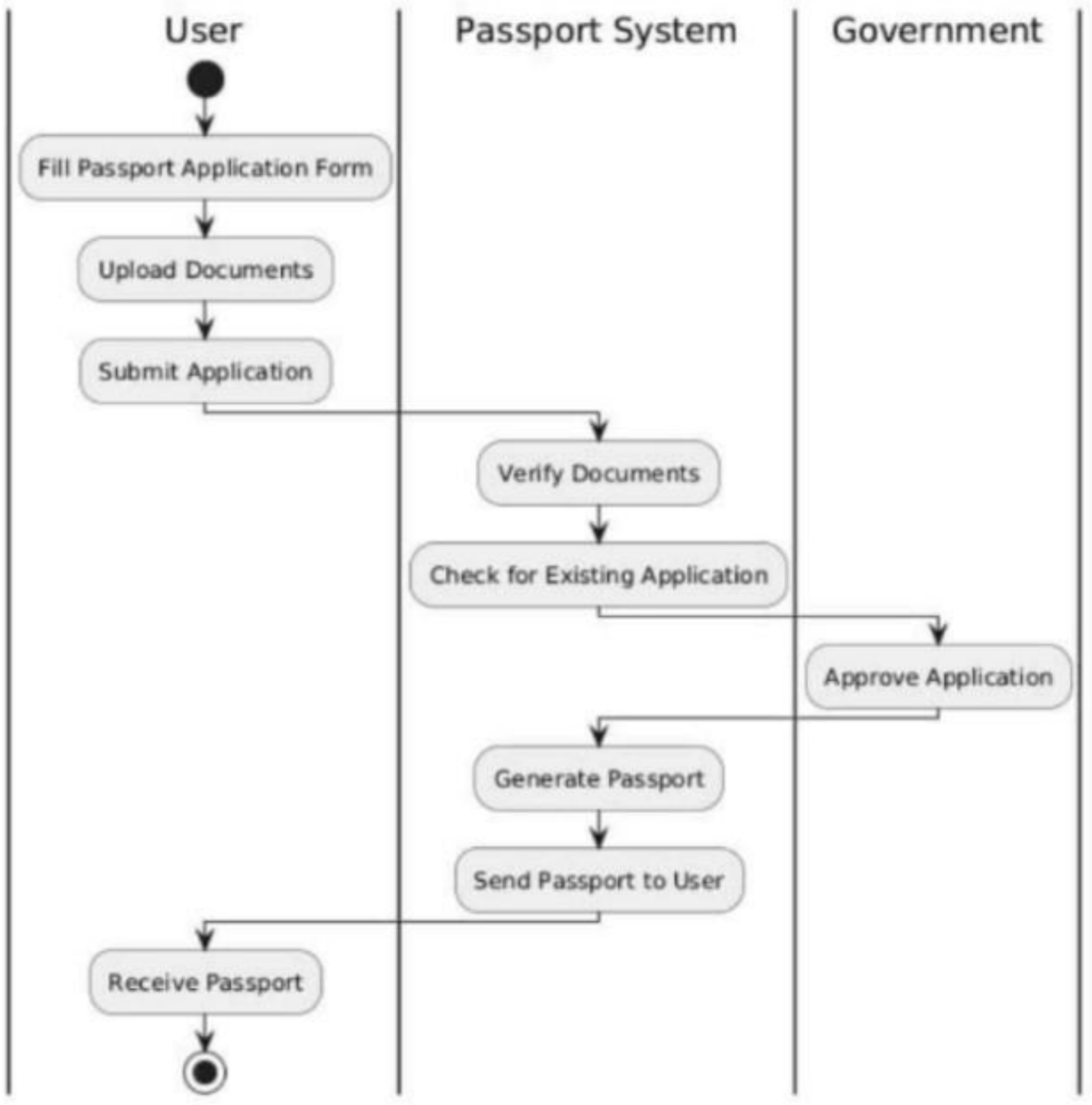


Fig 5.5

Brief Description:

The activity diagram describes the workflow of the verification process:

1. Applicant submits the application.

2. System forwards the application to officials for document verification.

3. Officials verify documents and approve/reject the application.

4. For approved applications, initiate police clearance.

5. Police department sends a clearance status.

6. Officials update the application status to "Approved".

7. System generates and issues the passport.