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

Lab Record

Software Engineering and Object-Oriented Modeling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

SUPRIYA S

1BM22CS350

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
Mar-June 2024

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Analysis and Design(22CS6PCSEO) laboratory has been carried out by SUPRIYA S (1BM22CS350) during the 5th Semester Oct24-Jan2025.

Signature of the Faculty Incharge:

NAME OF THE FACULTY:

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

Table of Contents

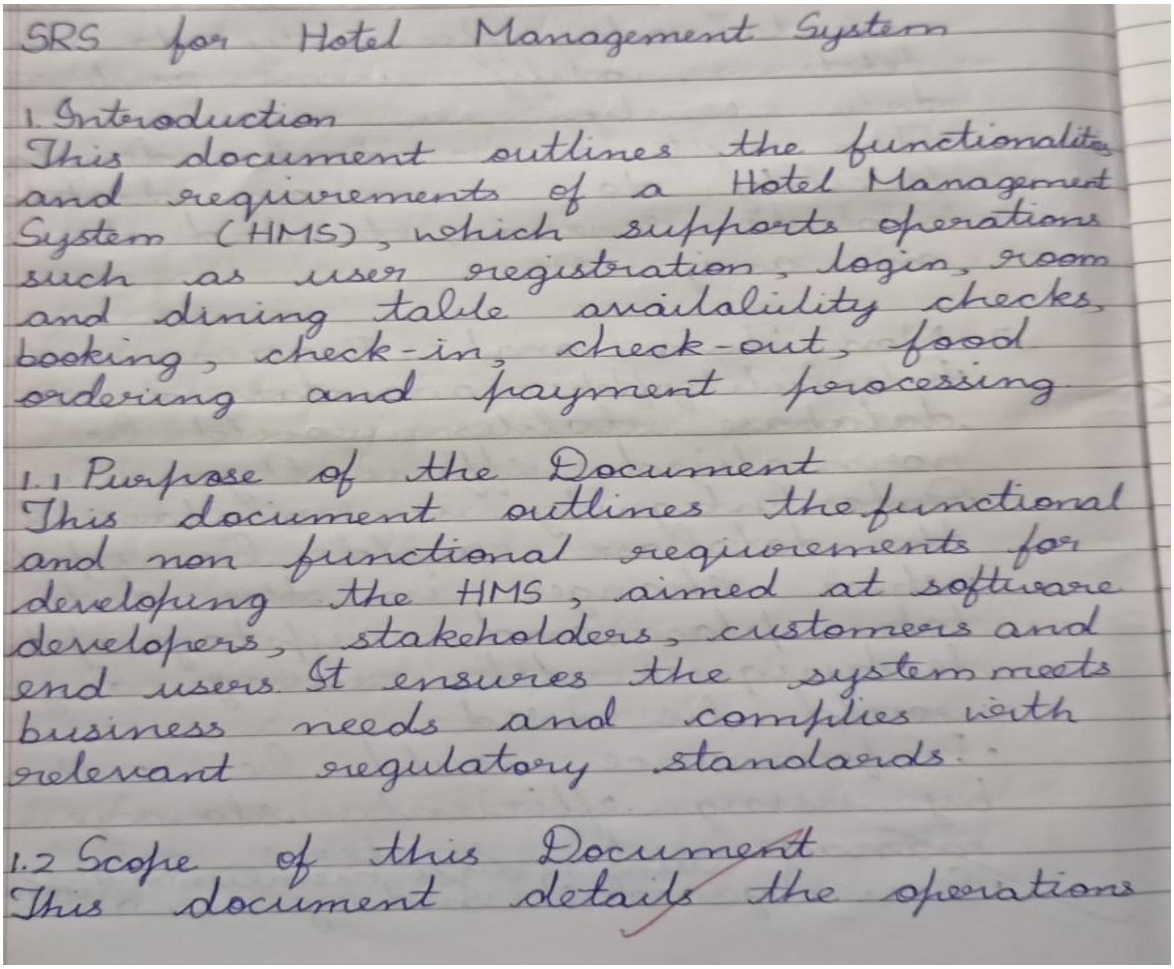
GitHub Link: https://github.com/SupriyaS26/oomd_1BM22CS350.git

1. Hotel Management System

Problem Statement

A hotel needs a system to manage its daily operations efficiently. The system should handle room bookings, check-ins, and check-outs, while also keeping track of guest details and payments. Additionally, it should allow staff to update room availability and view booking records. The goal is to create a simple, user-friendly application that helps streamline hotel operations and provides a smooth experience for both staff and guests. This system will use object-oriented concepts to ensure easy development, maintenance, and scalability.

SRS-Software Requirements Specification


SRS for Hotel Management System

1. Introduction
This document outlines the functionalities and requirements of a Hotel Management System (HMS), which supports operations such as user registration, login, room and dining table availability checks, booking, check-in, check-out, food ordering and payment processing.

1.1 Purpose of the Document
This document outlines the functional and non functional requirements for developing the HMS, aimed at software developers, stakeholders, customers and end users. It ensures the system meets business needs and complies with relevant regulatory standards.

1.2 Scope of this Document
This document details the operations

Fig 1.1

provided by the HMS, including user registration, login, room and dining table availability, booking of rooms or dining tables, food order menu, check-in, check-out and payment processing.

1.3 Overview

The Hotel Management System (HMS) provides core functionalities such as user authentication, booking management, food ordering, check-in/out processes, and payment handling.

2. General Description

HMS provides the following operations:

1) User authentication: is done during registration process and login operation.
2) Availability of rooms or dining tables: by keeping track of rooms and dining tables.
3) Food ordering from the menu
4) User login, Logout: users will use their login credentials to login and logout when session expires or they require.
5) Payments can be processed via multiple banking services.

3. Functional Requirements

1) User registration and login:
Users should be able to register

Fig 1.2

with basic personal details and set a password. Use these credentials during login.
2) Availability of rooms or dining tables: by keeping track of rooms and dining tables.
3) Food ordering by menu.
4) Payment processing: by any banking services including UPI, credit cards, debit cards, etc.
5) Check-in and check-out:
- Users will be able to check in at their reserved time and the system will track their stay duration.
- Upon check-out, the system will generate an invoice for the final payment.

4. Interface Requirements
1) Web interface: users can use the HMS website.
2) The system will be responsive and usable across devices of various screen sizes.

5. Performance Requirements
1) Response time: should be under 1-2 seconds.
2) The system should have scalable storage options to accomodate data growth over time.

Fig 1.3

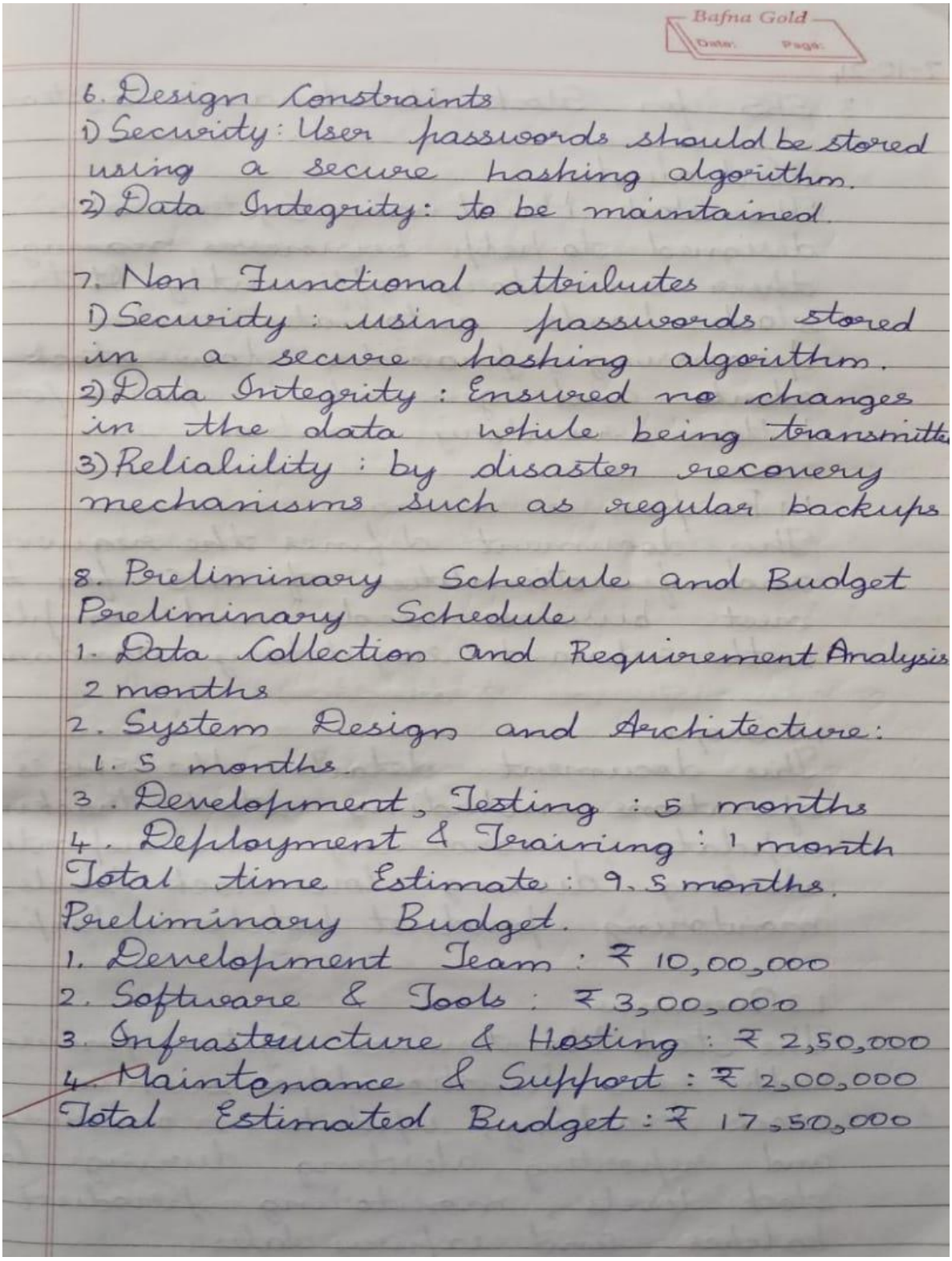

Bafna Gold
Date:
Page:
6. Design Constraints
1) Security: User passwords should be stored using a secure hashing algorithm.
2) Data Integrity: to be maintained.
7. Non Functional attributes
1) Security: using passwords stored in a secure hashing algorithm.
2) Data Integrity: Ensured no changes in the data while being transmitte
3) Reliability: by disaster recovery mechanisms such as regular backups
8. Preliminary Schedule and Budget
Preliminary Schedule
1. Data Collection and Requirement Analysis 2 months
2. System Design and Architecture: 1. 5 months.
3. Development, Testing : 5 months
4. Deployment & Training : 1 month
Total time Estimate : 9.5 months.
Preliminary Budget.
1. Development Team : ₹ 10,00,000
2. Software & Tools : ₹ 3,00,000
3. Infrastructure & Hosting : ₹ 2,50,000
4. Maintenance & Support : ₹ 2,00,000
Total Estimated Budget : ₹ 17,50,000

Fig 1.4

Class Diagram

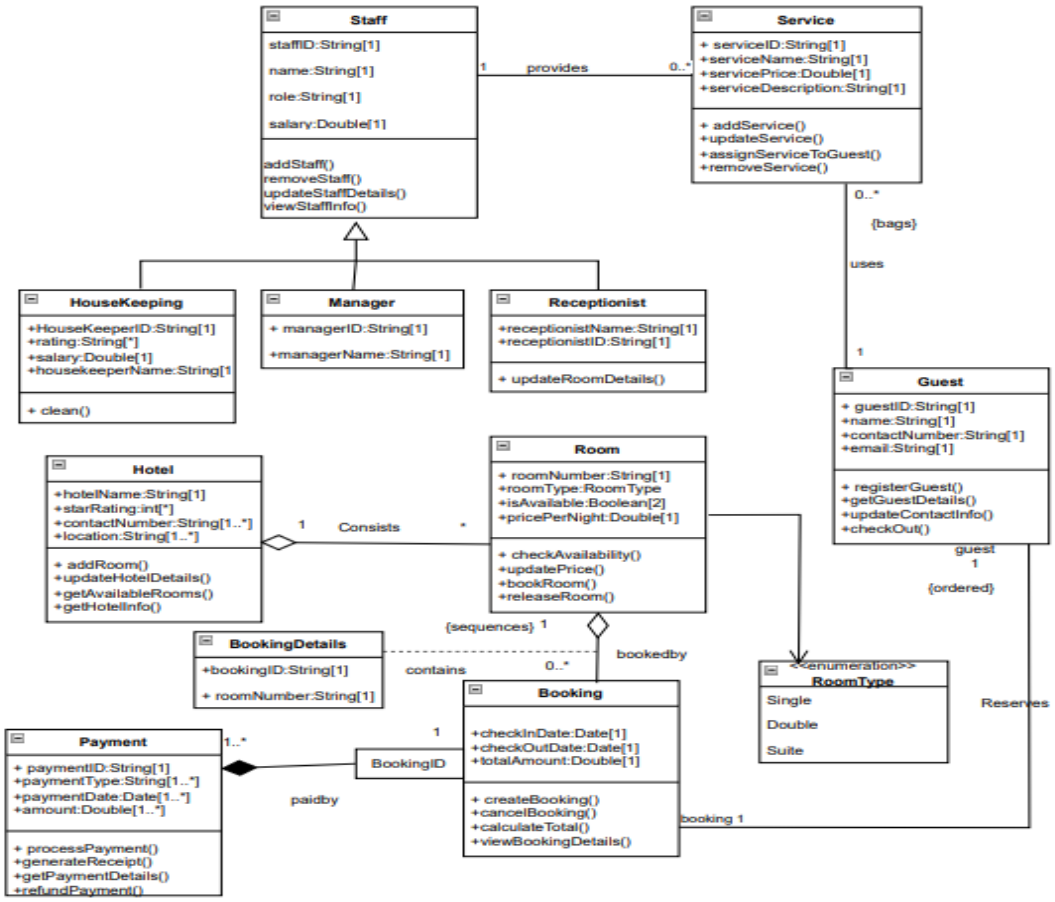


Fig 1.5

The Hotel Management System class diagram shows how different parts of the system work together. The **Hotel** manages multiple **Rooms**, which guests can book through the **Booking** class. Each booking is linked to a guest and payment, ensuring the process is organized. Guests can also use **Services** like room service, provided by **Staff**, who have different roles like managers or receptionists. The system includes a **RoomType** enumeration for types like single or suite, and it ensures a room cannot exist without a hotel (composition). The **BookingDetails** class acts as an association class, linking rooms and booking class.

State Diagram

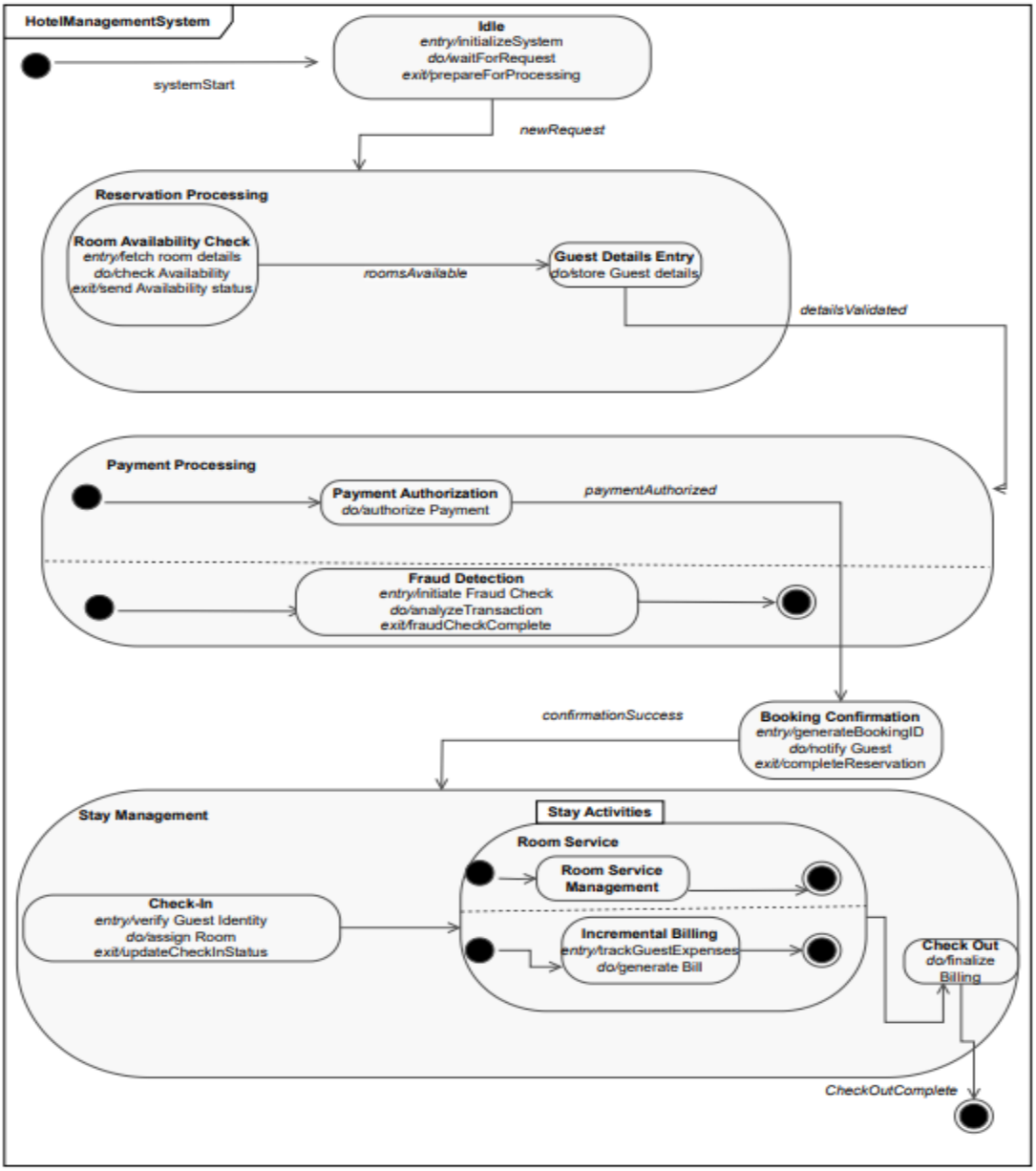


Fig 1.6

This state diagram shows how a Hotel Management System works step by step. It starts in the **Idle** state, waiting for a new request. When a request comes in, the system checks room availability and saves guest details. Next, it processes payments and ensures there are no fraud issues. Once the payment is confirmed, the booking is completed. During the guest's stay, the system manages check-in, room service, and billing. Finally, the guest checks out, and the process ends after the billing is finalized.

Use Case Diagram

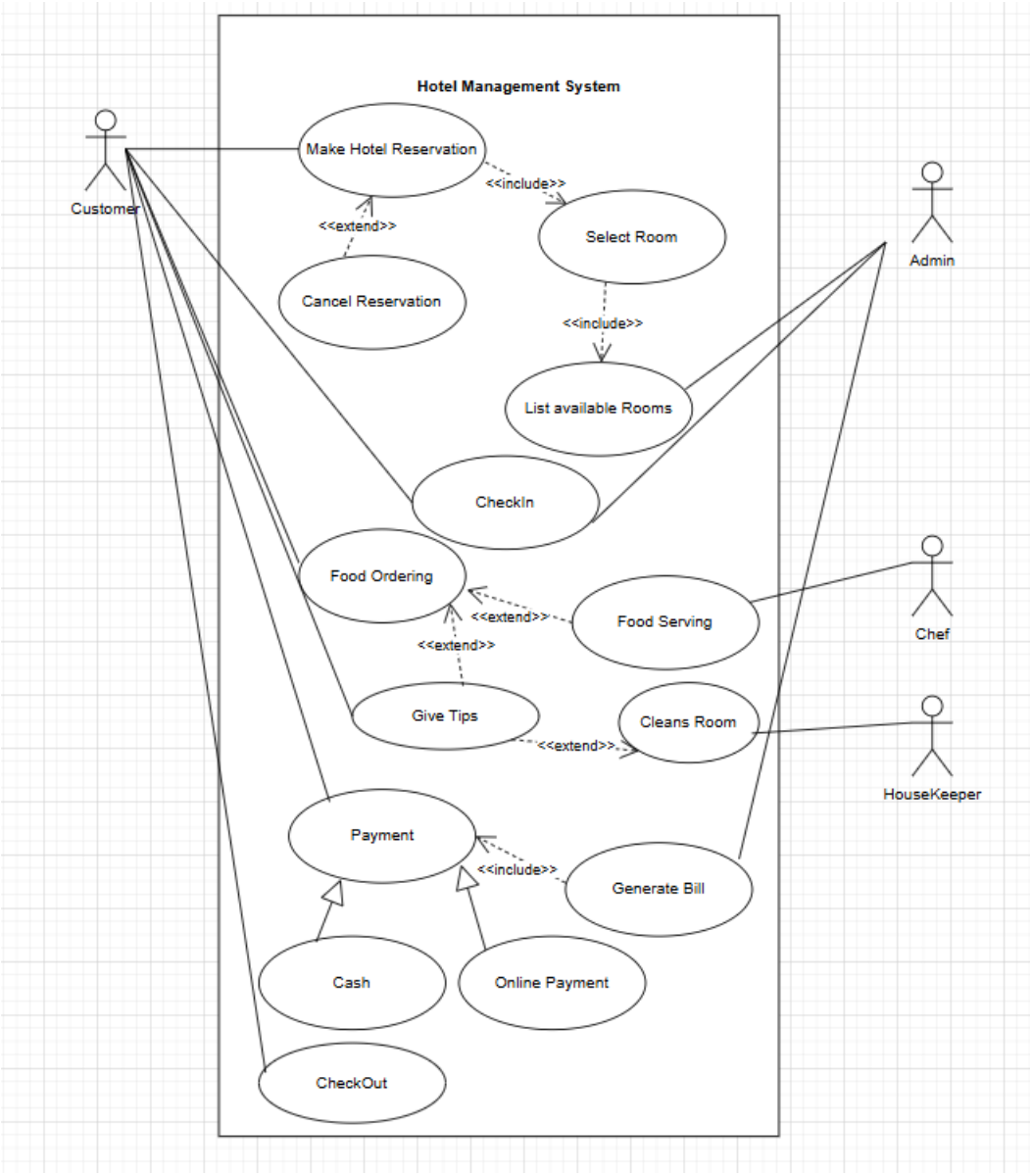


Fig 1.7

Use Case: Hotel Management System

Summary: This use case describes the process by which customers interact with the hotel system for reservations, check-ins, food orders, payments, and check-out, as well as the administrative functions for managing room listings.

Actors: Customer, Admin, Chef, Housekeeper

Preconditions:

The hotel system is operational, with the customer registered or providing necessary details, available rooms listed and up-to-date, and the admin having access to manage room availability and reservations.

Description:

Customers can make or cancel reservations, select rooms, check in, order food, and make payments using cash or online methods. They can also tip and check out. **Admins** manage room listings, while **Chefs** handle food orders and serving. **Housekeepers** are responsible for cleaning rooms. Key activities like generating bills and listing available rooms are included as part of the system's workflow.

Exceptions:

- **No Available Rooms**: The system fails to find available rooms for the selected date or room type, leading to a failed reservation attempt.
- **Incomplete Information**: If the customer fails to provide required details for reservation, payment, or check-in.
- **Failed Payment**: If the customer's payment fails due to insufficient funds, invalid method, or transaction error.
- **Cancellation Restrictions**: If a reservation cannot be canceled due to time restrictions (e.g., within 24 hours before check-in).
- **Room Not Ready**: If a room is not available or properly cleaned at check-in time.
- **Food Order Issue**: If food items are out of stock or incorrectly ordered, leading to delays or a failed food delivery.

Postconditions:

The system processes all customer activities (reservation, check-in, food ordering, payment, and check-out), updates room availability and statuses, records payments with receipts, and allows the admin to manage room listings and view customer reservations.

Sequence Diagram:

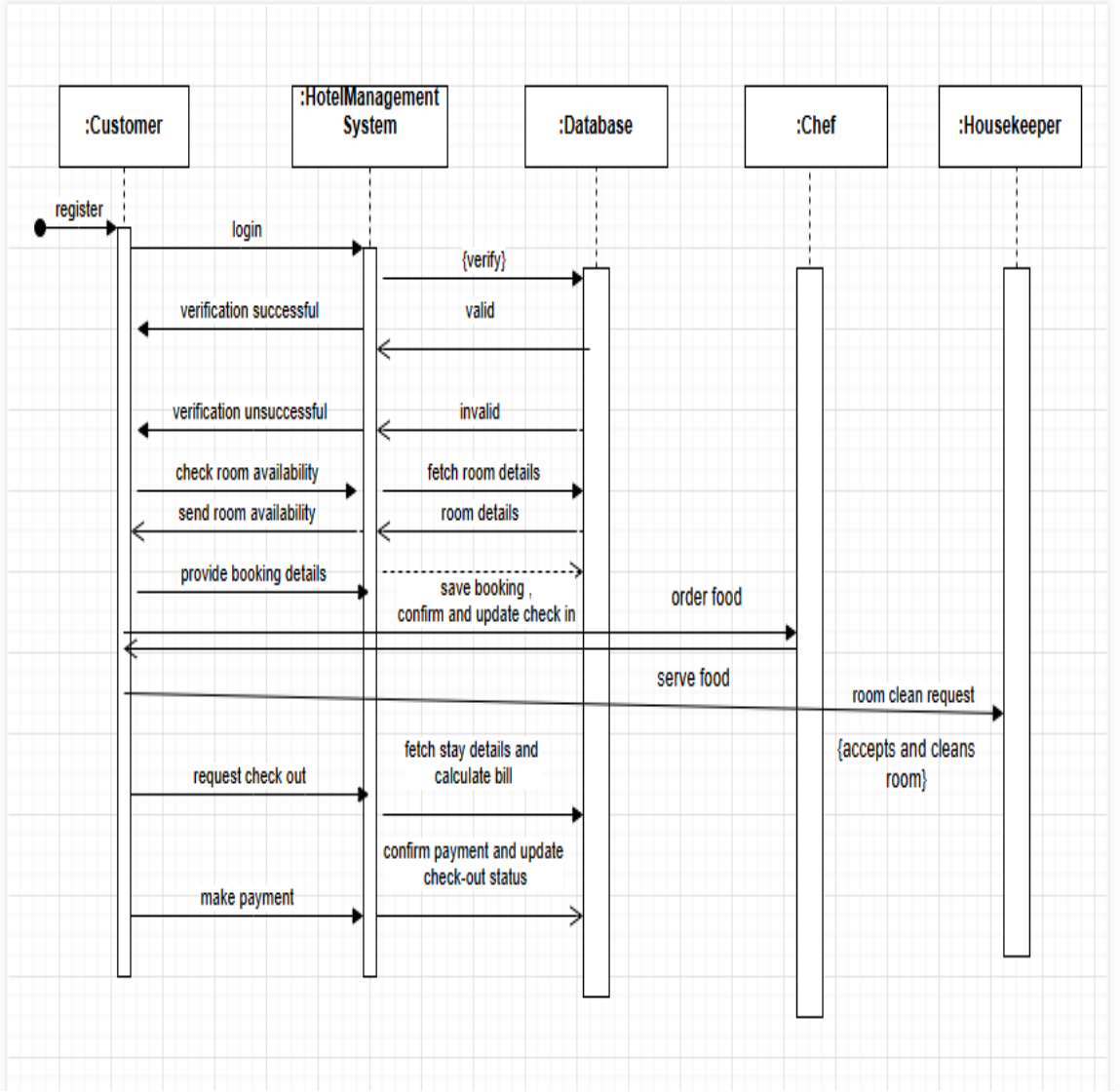


Fig 1.8: Simple Sequence Diagram

The diagram represents a sequence of interactions in a hotel management system. It shows how a customer registers and logs in, with the system verifying their credentials. After logging in, the system checks room availability and confirms bookings. The customer can request services such as food (handled by a chef) or room cleaning (handled by a housekeeper). When checking out, the system calculates the bill, processes payment, and updates the checkout status. The database is used to fetch and update information throughout the process.

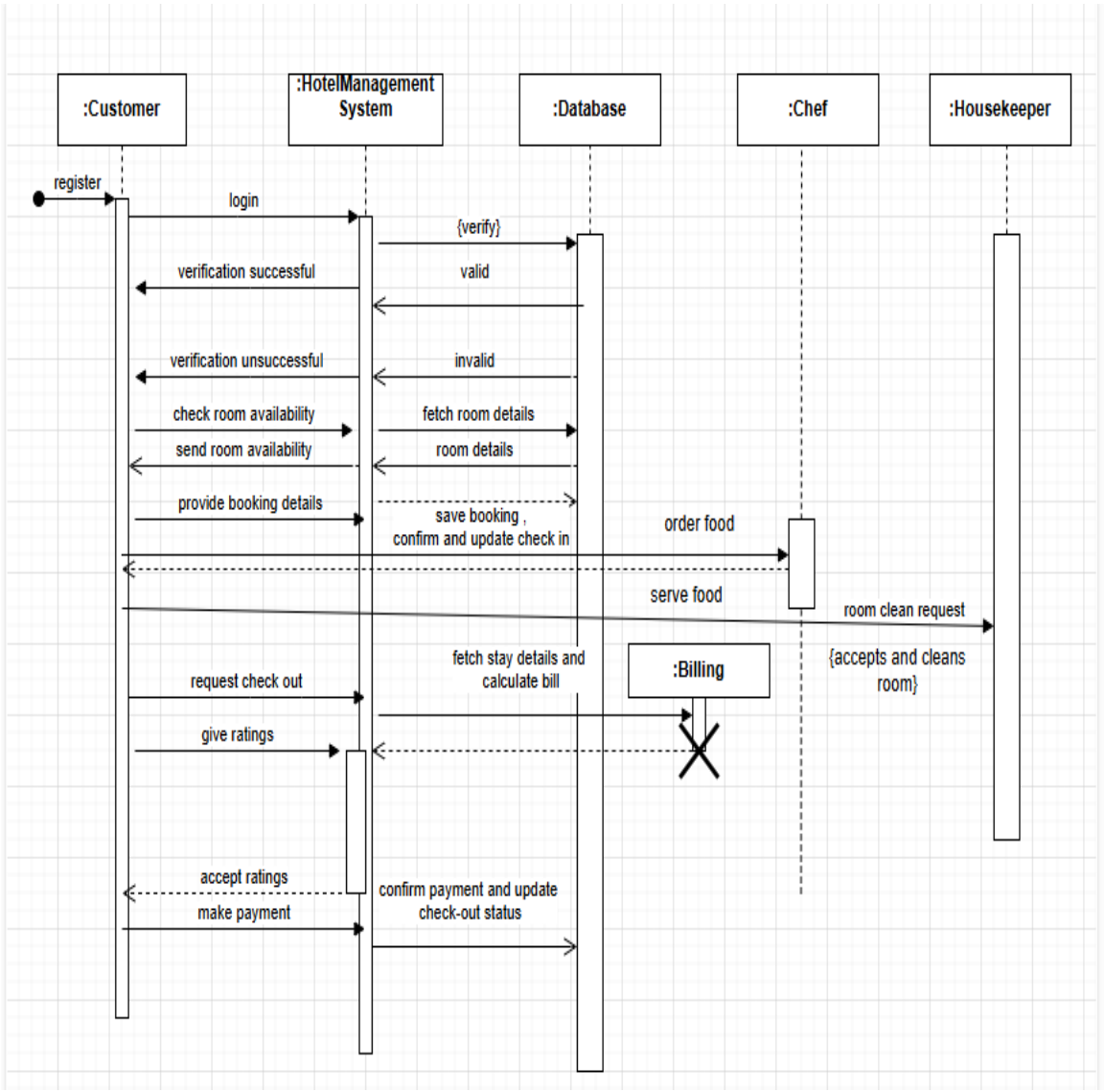


Fig 1.9: Advance Sequence Diagram

This updated sequence diagram includes both passive and transient objects. The **passive object**: Billing represents the billing process and is used during checkout to fetch stay details and calculate the bill. It is a temporary object that is created during the checkout process and destroyed once the transaction is complete, indicated by the X on its lifeline. Additional customer interactions, such as providing ratings and their acceptance, have also been incorporated into the flow for a more detailed representation.

Activity Diagram

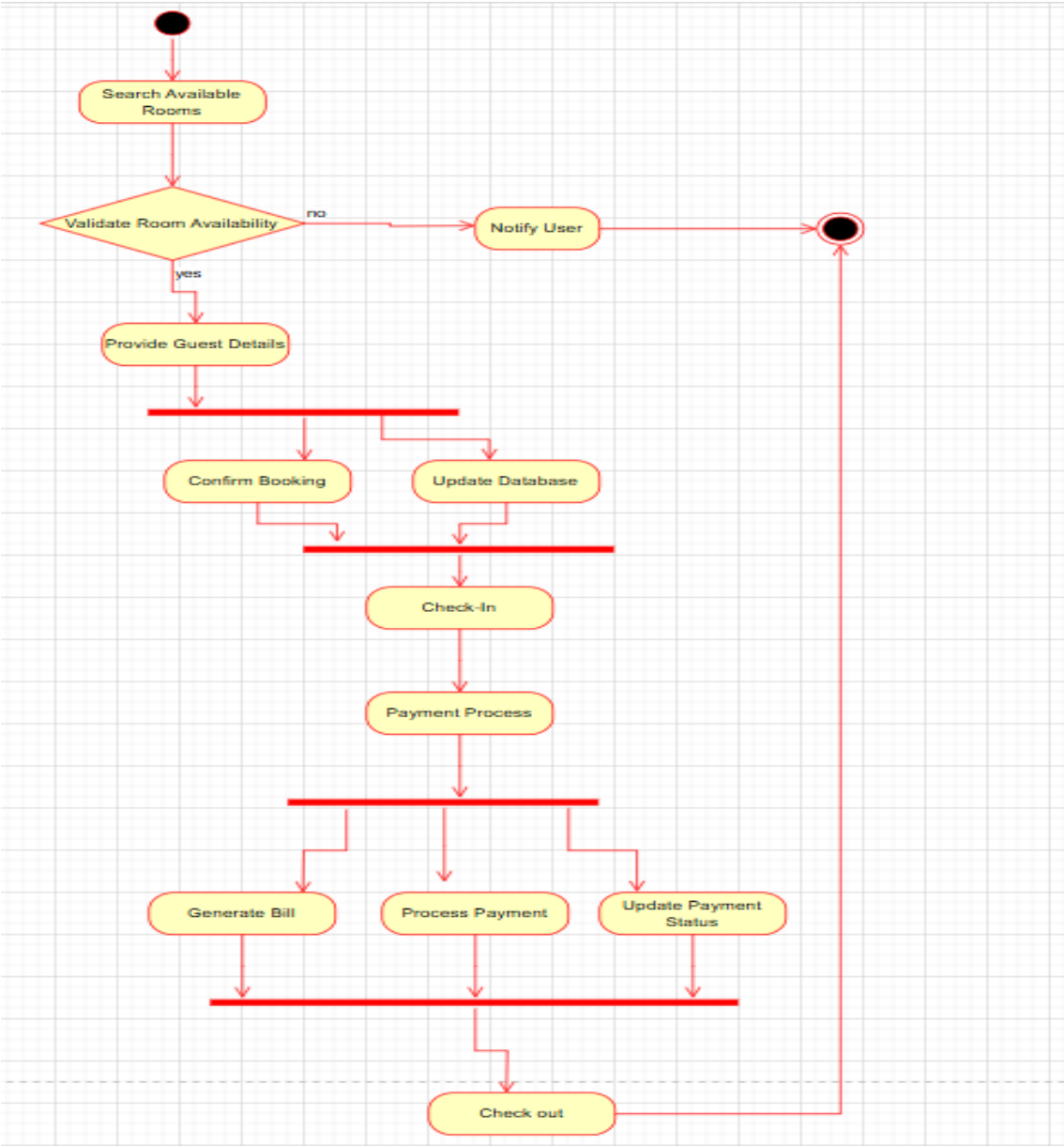


Fig 1.10: Simple Activity Diagram

This simple activity diagram represents the process of booking a hotel room and checking in. The process begins with the guest searching for available rooms. If a suitable room is found, the guest selects and confirms the booking. The system checks the room's availability and, if available, reserves the room and updates its status. The payment system then processes the guest's payment, and if successful, sends a confirmation.

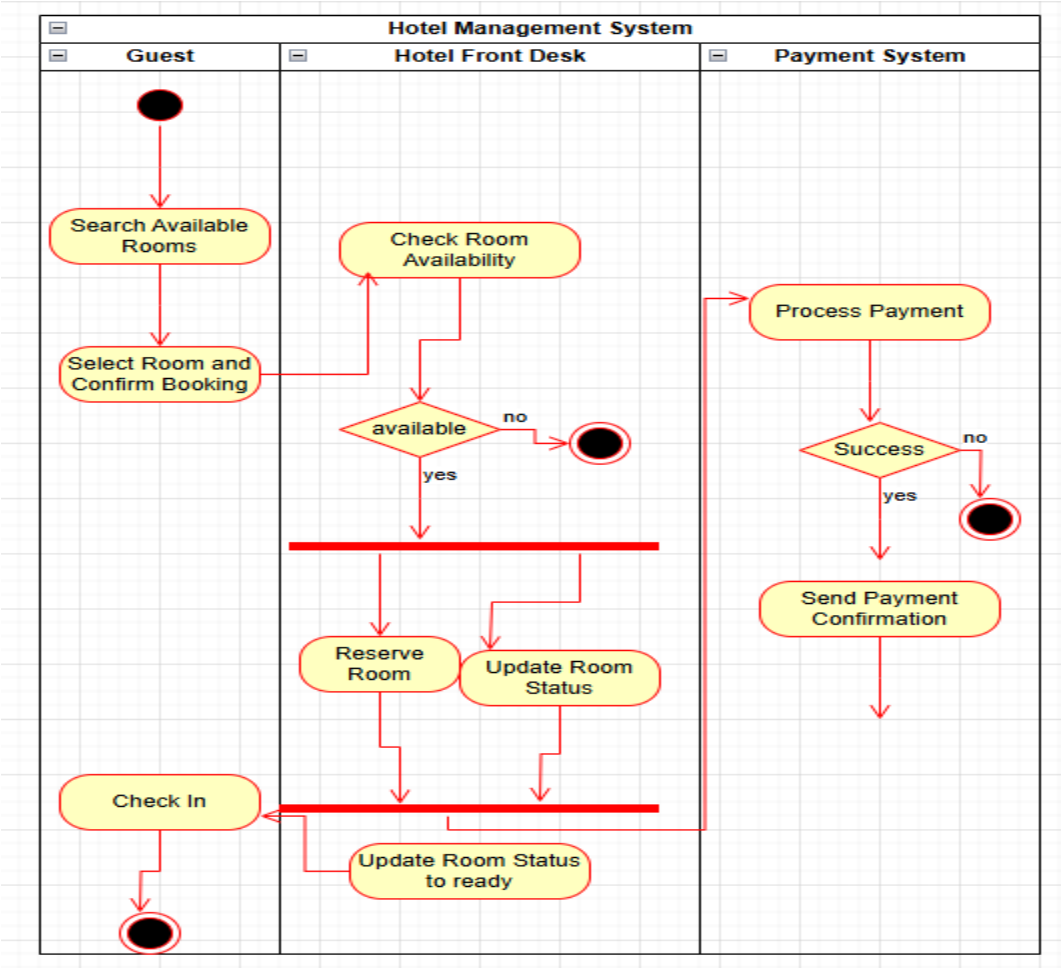


Fig 1.11: Advanced Activity Diagram

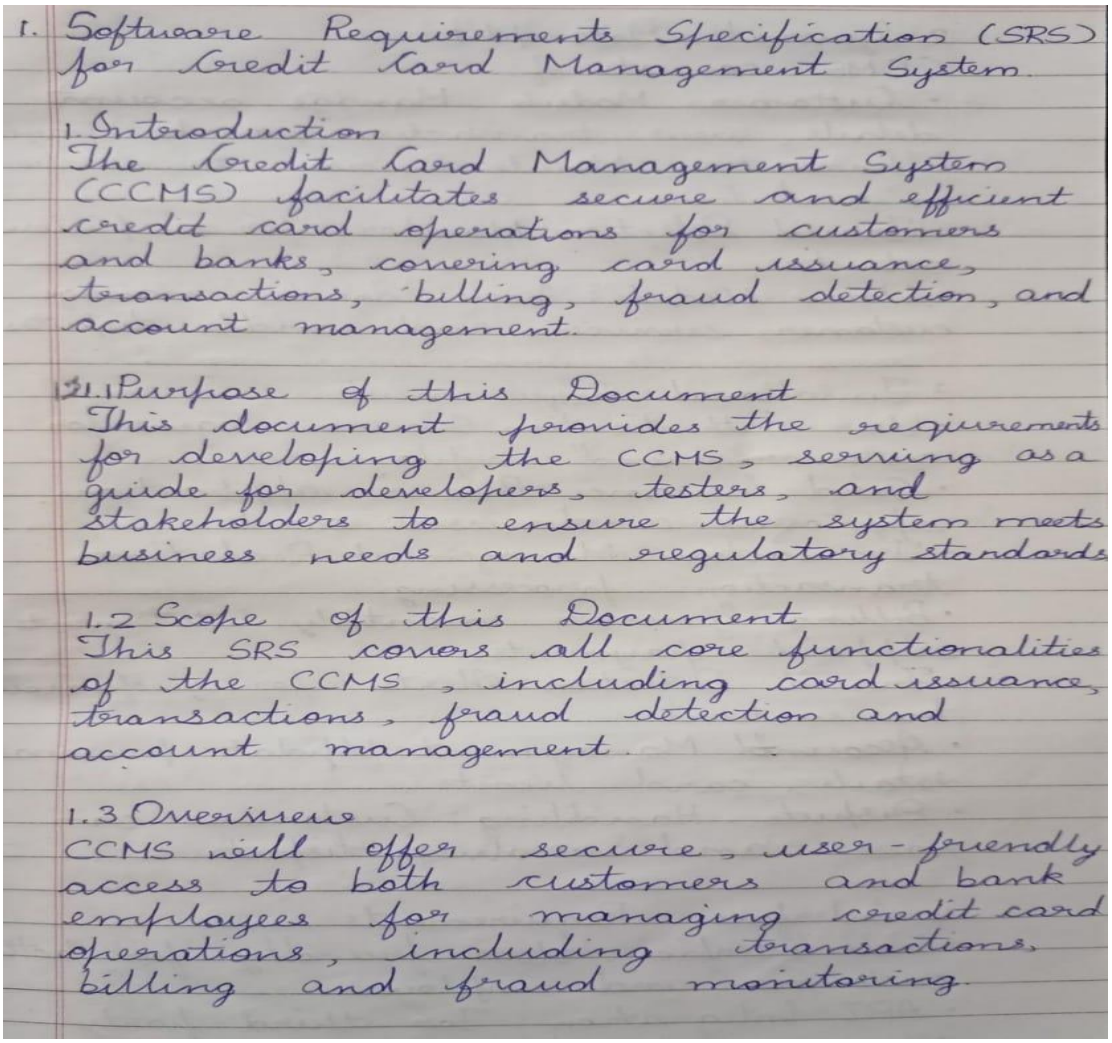
This activity diagram outlines the flow of the hotel booking and check-in process involving three main entities: the **Guest**, **Hotel Front Desk**, and **Payment System**. The process begins with the guest searching for available rooms, selecting a room, and confirming the booking. The hotel front desk checks room availability and, if available, reserves the room and updates its status to reflect readiness for check-in. Simultaneously, the payment system processes the payment and, upon successful completion, sends a payment confirmation. The process concludes with the guest checking in after the room status is updated to "ready." Decision points in the diagram ensure proper handling of scenarios such as room availability and payment success, ensuring a smooth flow.

2. Credit Card Processing System

Problem Statement

The problem in credit card processing lies in ensuring secure, efficient, and accurate handling of transactions between customers, merchants, and banks. Challenges include preventing fraud, ensuring data privacy, verifying cardholder details, and processing payments in real-time while maintaining system reliability. The solution requires a robust system that validates transactions, securely communicates with banks, and handles errors or declined payments effectively, ensuring a seamless experience for all parties involved.

SRS-Software Requirements Specification



1. Software Requirements Specification (SRS) for Credit Card Management System.

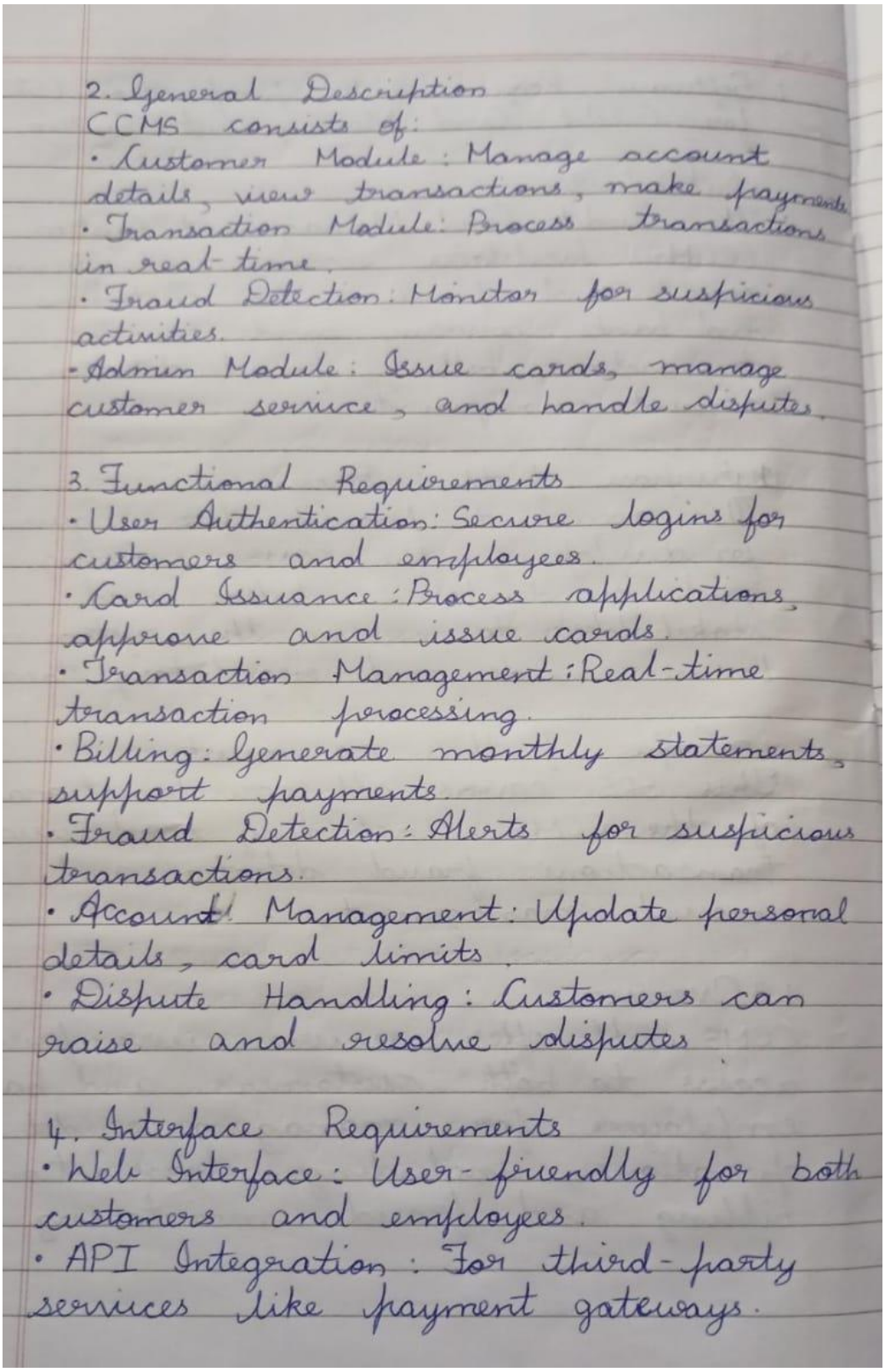
1. Introduction
The Credit Card Management System (CCMS) facilitates secure and efficient credit card operations for customers and banks, covering card issuance, transactions, billing, fraud detection, and account management.

1.1 Purpose of this Document
This document provides the requirements for developing the CCMS, serving as a guide for developers, testers, and stakeholders to ensure the system meets business needs and regulatory standards.

1.2 Scope of this Document
This SRS covers all core functionalities of the CCMS, including card issuance, transactions, fraud detection and account management.

1.3 Overview
CCMS will offer secure, user-friendly access to both customers and bank employees for managing credit card operations, including transactions, billing and fraud monitoring.

Fig 2.1



2. General Description
CCMS consists of:
• Customer Module: Manage account details, view transactions, make payments.
• Transaction Module: Process transactions in real-time.
• Fraud Detection: Monitor for suspicious activities.
• Admin Module: Issue cards, manage customer service, and handle disputes.

3. Functional Requirements
• User Authentication: Secure logins for customers and employees.
• Card Issuance: Process applications, approve and issue cards.
• Transaction Management: Real-time transaction processing.
• Billing: Generate monthly statements, support payments.
• Fraud Detection: Alerts for suspicious transactions.
• Account Management: Update personal details, card limits.
• Dispute Handling: Customers can raise and resolve disputes.

4. Interface Requirements
• Web Interface: User-friendly for both customers and employees.
• API Integration: For third-party services like payment gateways.

Fig 2.2

- Mobile Compatibility: Optimized for mobile access.

5. Performance Requirements
- Response Time: Transactions should process within 2 seconds.
- Scalability: Support up to 1 million concurrent users
- Availability: 99.9% uptime is required

6. Design Constraints
- Security: ~~Must comply~~ By using a 16 digit number along with CVV.
- Compliance: Adhere to banking regulations and privacy laws.
- Data Integrity: Ensure accurate and secure transaction data.

7. Non - Functional Attributes
- Security: Encryption for data storage and transmission.
- Reliability: System must be able to recover from failures.
- Usability: Intuitive interface for customers and employees.
- Maintainability: Modular design for easy updates.

8. Preliminary Schedule and Budget
- Requirements Gathering: 1 month
- Design and Development: 6 months
- Testing and Deployment: 2 months.

Total Estimated Budget: ₹ 34,900,000

Fig 2.3

Class Diagram

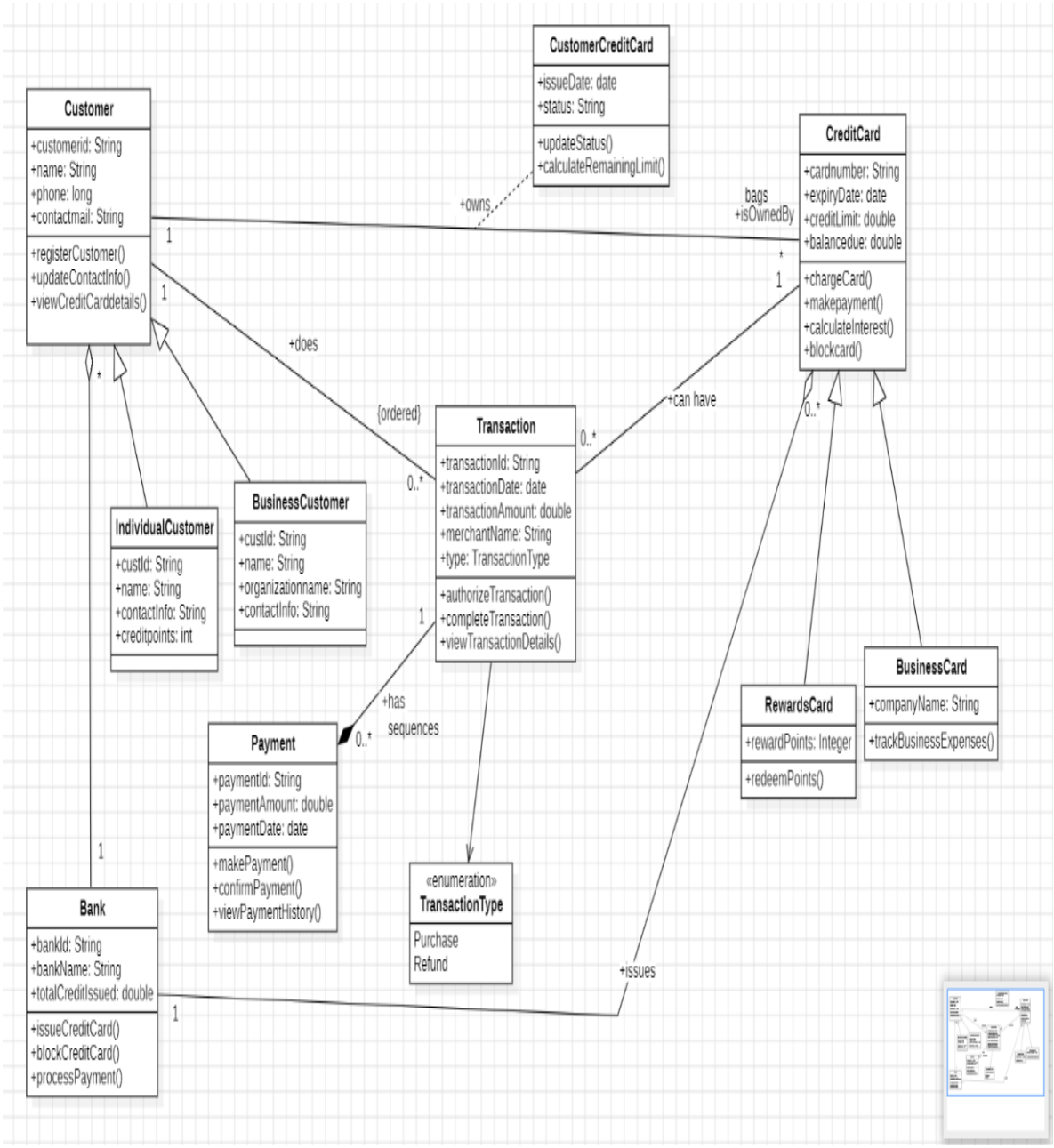


Fig 2.4

This class diagram models a credit card processing system. It includes entities like customers (individual and business), credit cards, transactions, payments, and banks. Customers can own credit cards and make transactions. Transactions can be authorized, completed, or refunded. Payments can be made to settle transaction balances. The system also supports rewards cards for business customers.

State Diagram

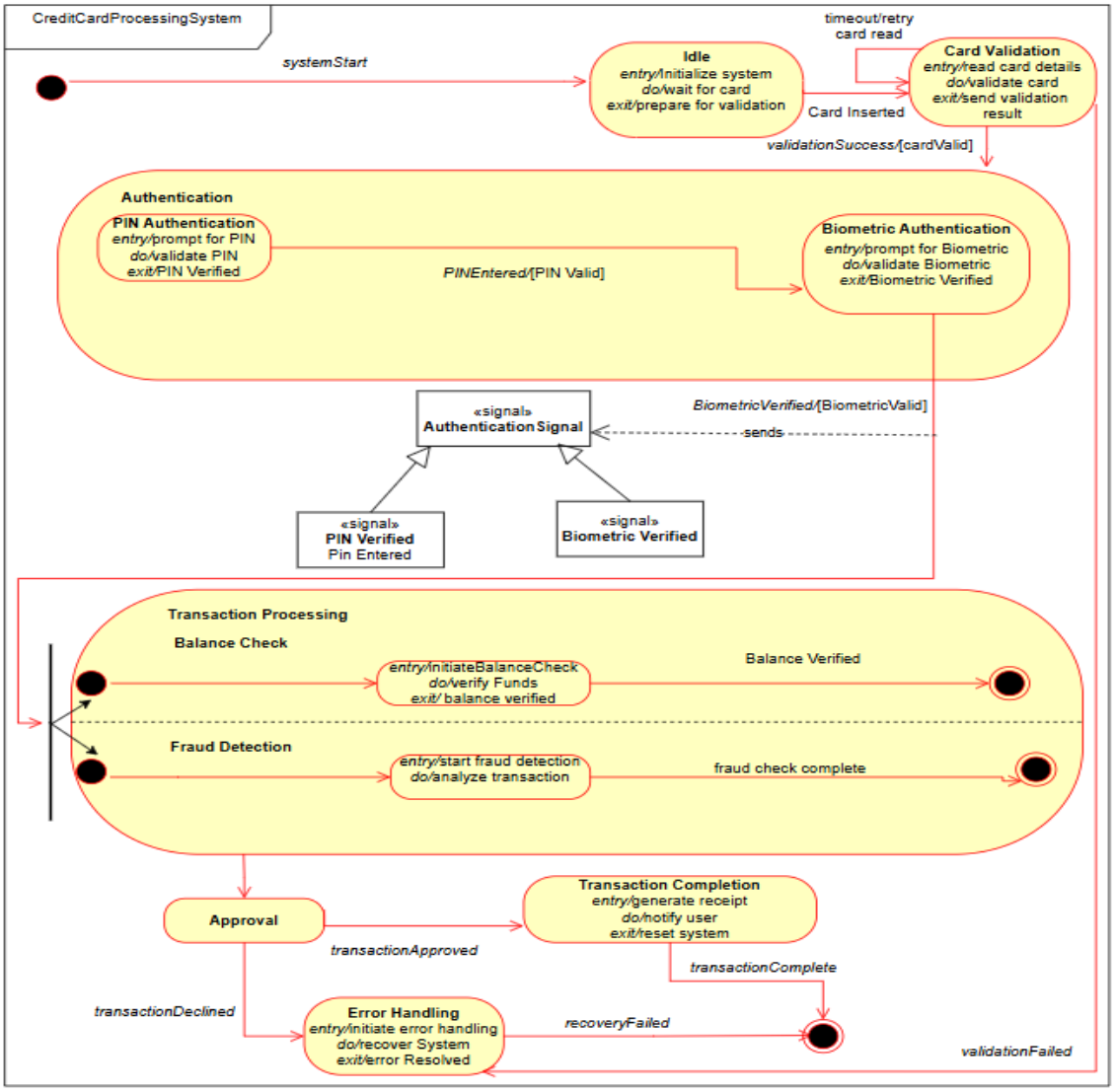


Fig 2.5

This state diagram illustrates the process of a credit card processing system. It starts with the card being inserted and then goes through validation. If the card is valid, the system moves to the authentication step, where the user can either enter their PIN or use biometric authentication. After successful authentication, the system checks the available balance and runs fraud detection checks. If everything is clear, the transaction is approved and completed. If any errors occur, the system enters an error handling state to try and resolve the issue.

Use Case Diagram

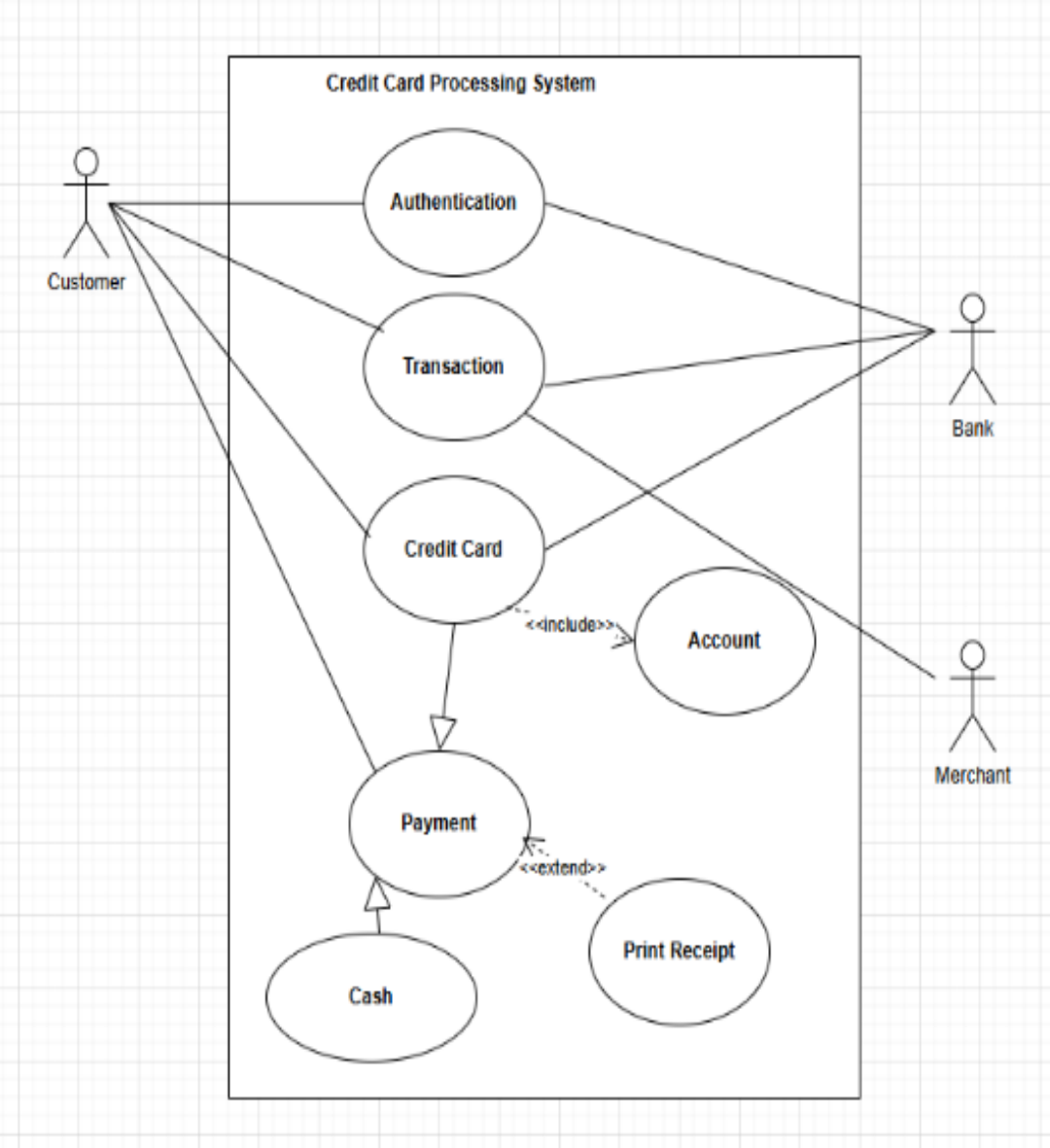


Fig 2.6

Use Case: Credit Card Processing System

Summary: This use case outlines the steps involved in processing a credit card transaction, from the customer's perspective.

Actors: Customer, Bank, Merchant

Precondition:

The customer has a valid credit card. The merchant has a valid merchant account with the bank.

Description:

The customer presents their credit card to the merchant for payment. The merchant enters the transaction amount and any relevant information (e.g., item description). The merchant verifies the customer's identity by requesting them to enter their PIN or provide biometric authentication. The bank's system checks the customer's available credit and approves or declines the transaction. If approved, the bank authorizes the transaction and sends a confirmation to the merchant. The merchant then completes the transaction and provides the customer with a receipt.

Exception:

- **Insufficient Funds:** If the customer's credit limit is insufficient, the transaction is declined.
- **Card Declined:** If the card is invalid or reported as stolen, the transaction is declined.
- **Network Error:** If there is a communication error between the merchant and the bank, the transaction may fail.

Post-condition:

- The transaction is successfully processed and recorded in the bank's system.
- The customer's account balance is updated to reflect the transaction.
- The merchant receives payment for the transaction.
- The customer receives a receipt as proof of purchase.

Sequence Diagram:

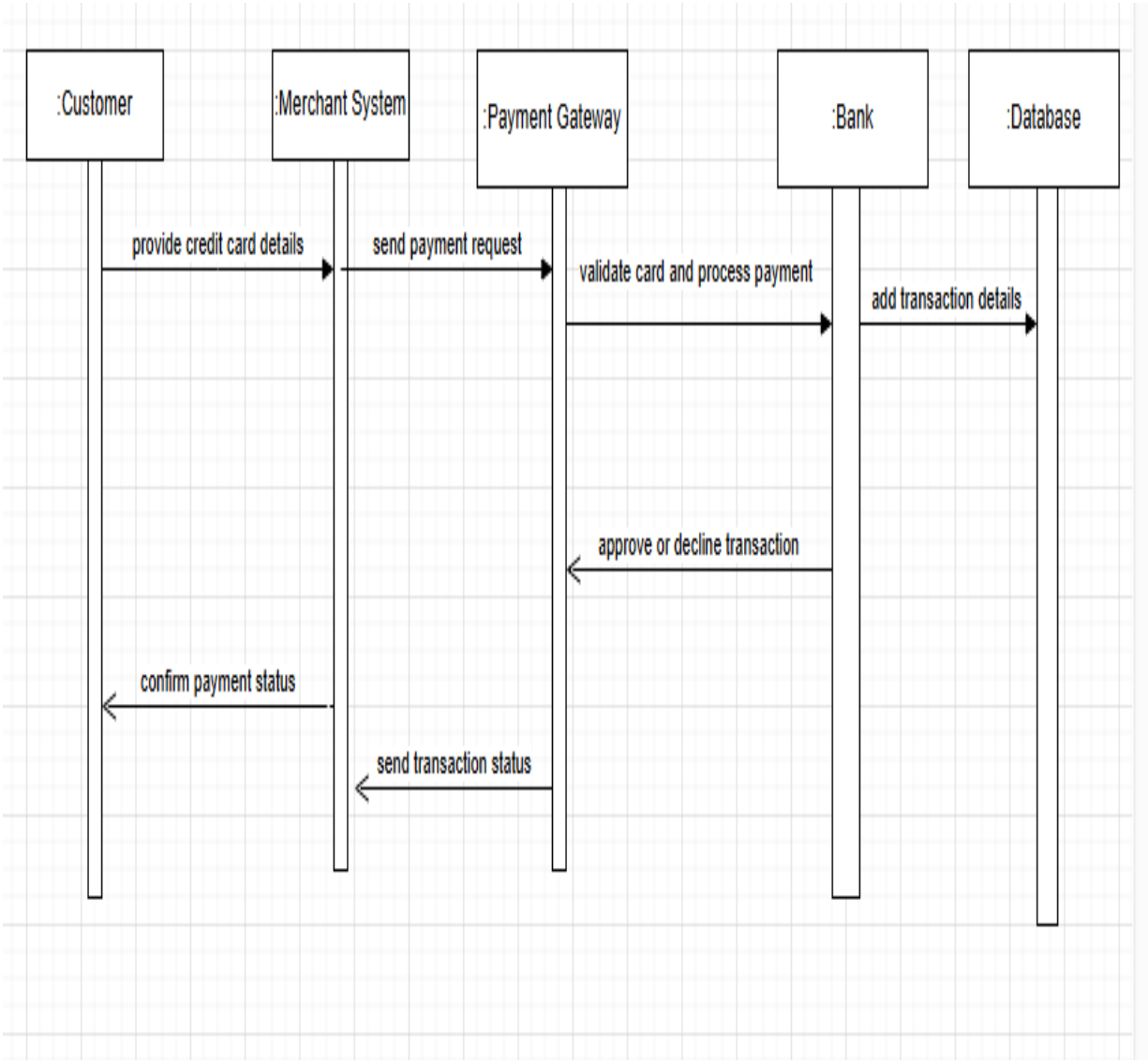


Fig 2.7: Simple Sequence Diagram

This diagram shows how a credit card transaction is processed. The Customer provides their card details to the Merchant System. The Merchant System then sends a payment request to the Payment Gateway. The Payment Gateway validates the card and processes the payment, communicating with the Bank to authorize the transaction. The Bank then approves or declines the transaction. Finally, the Payment Gateway sends the transaction status back to the Merchant System, who then confirms the status with the Customer.

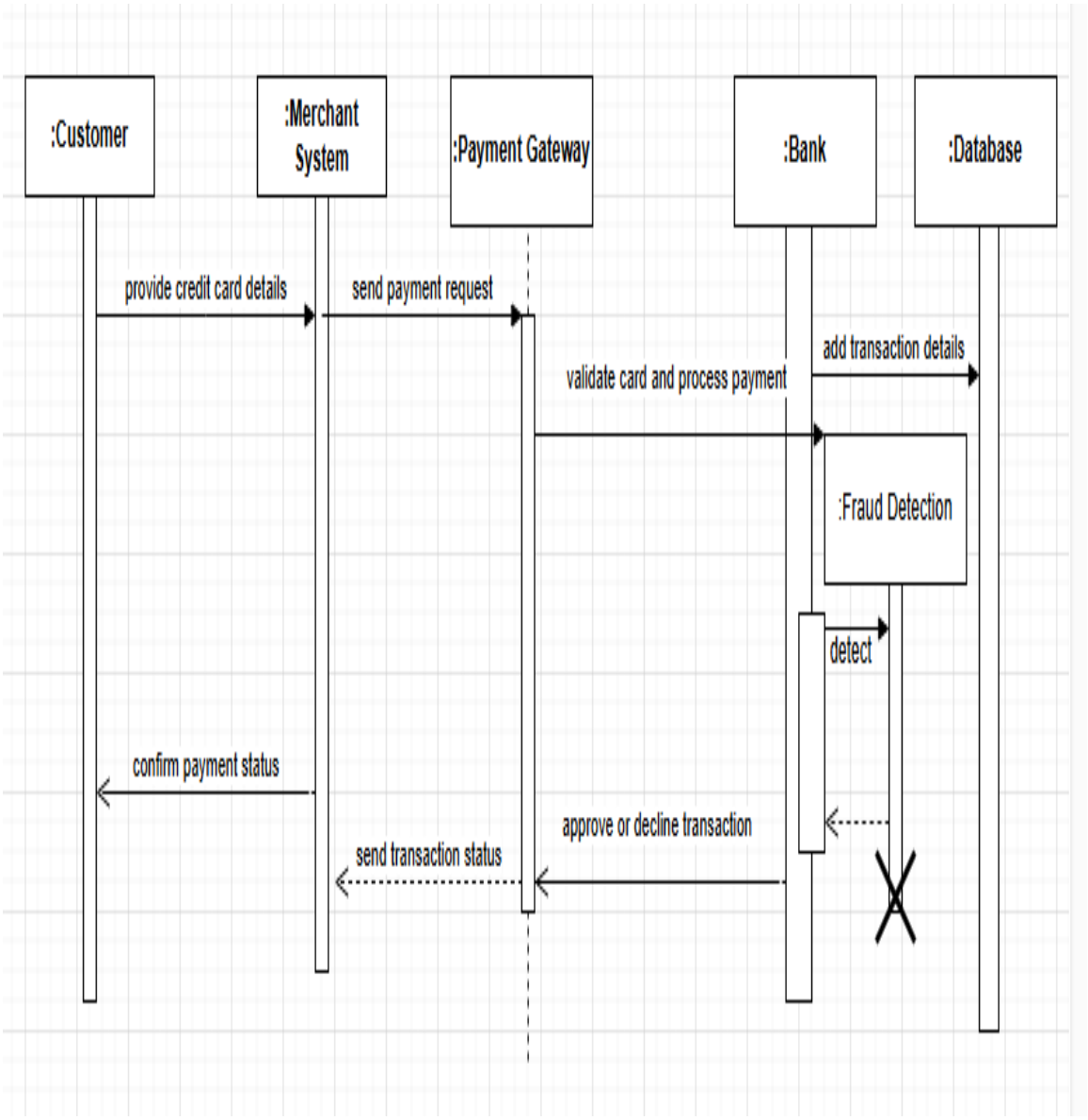


Fig 2.8: Advanced Sequence Diagram

This diagram illustrates the process of credit card transactions. The Customer initiates the transaction by providing their credit card details to the Merchant System. The Merchant System then sends a payment request to the Payment Gateway. The Payment Gateway validates the card and sends the transaction details to the Bank for processing. The Bank checks the transaction for potential fraud by sending the details to the Fraud Detection system. After the fraud check, the Bank approves or declines the transaction. Finally, the Payment Gateway sends the transaction status back to the Merchant System, who then confirms the status with the Customer.

Activity Diagram

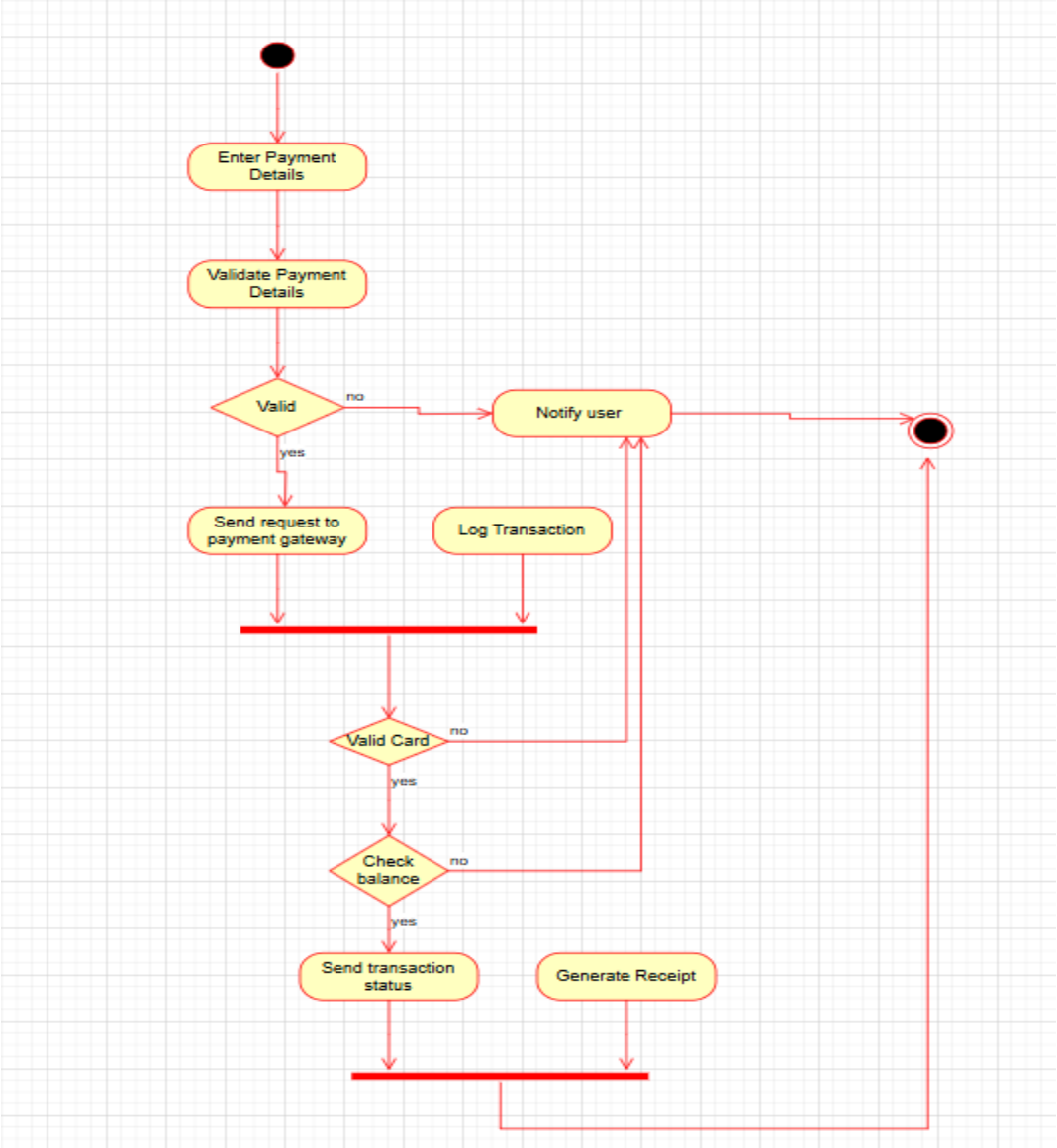


Fig 2.9: Simple Activity Diagram

This activity diagram illustrates the process of a credit card transaction. It starts with entering payment details, which are then validated. If the details are valid, a request is sent to the payment gateway. The gateway checks for a valid card and sufficient balance. If both are verified, the transaction status is sent, a receipt is generated, and the transaction is logged. If any step fails, the user is notified.

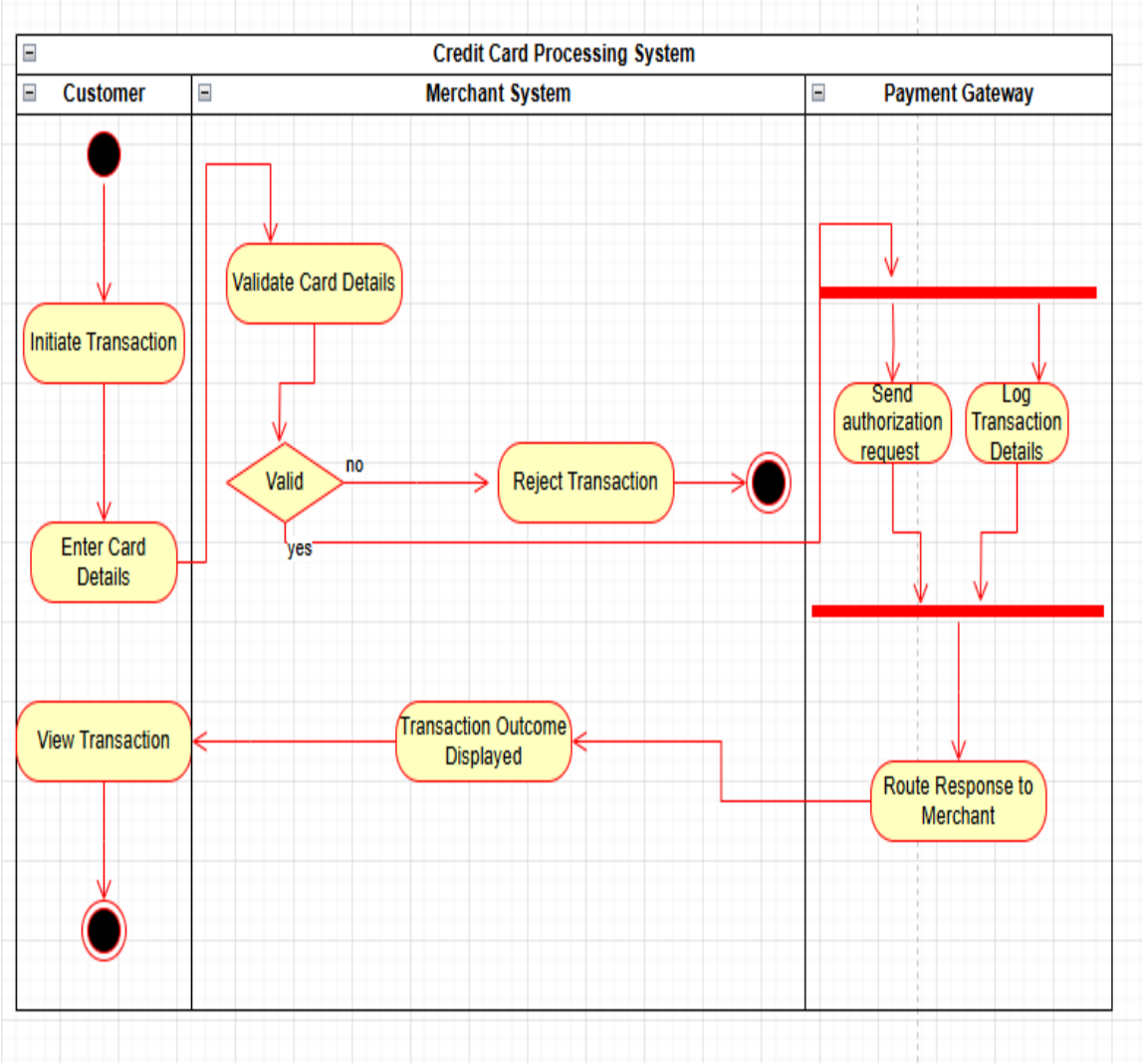


Fig 2.10: Advanced Activity Diagram

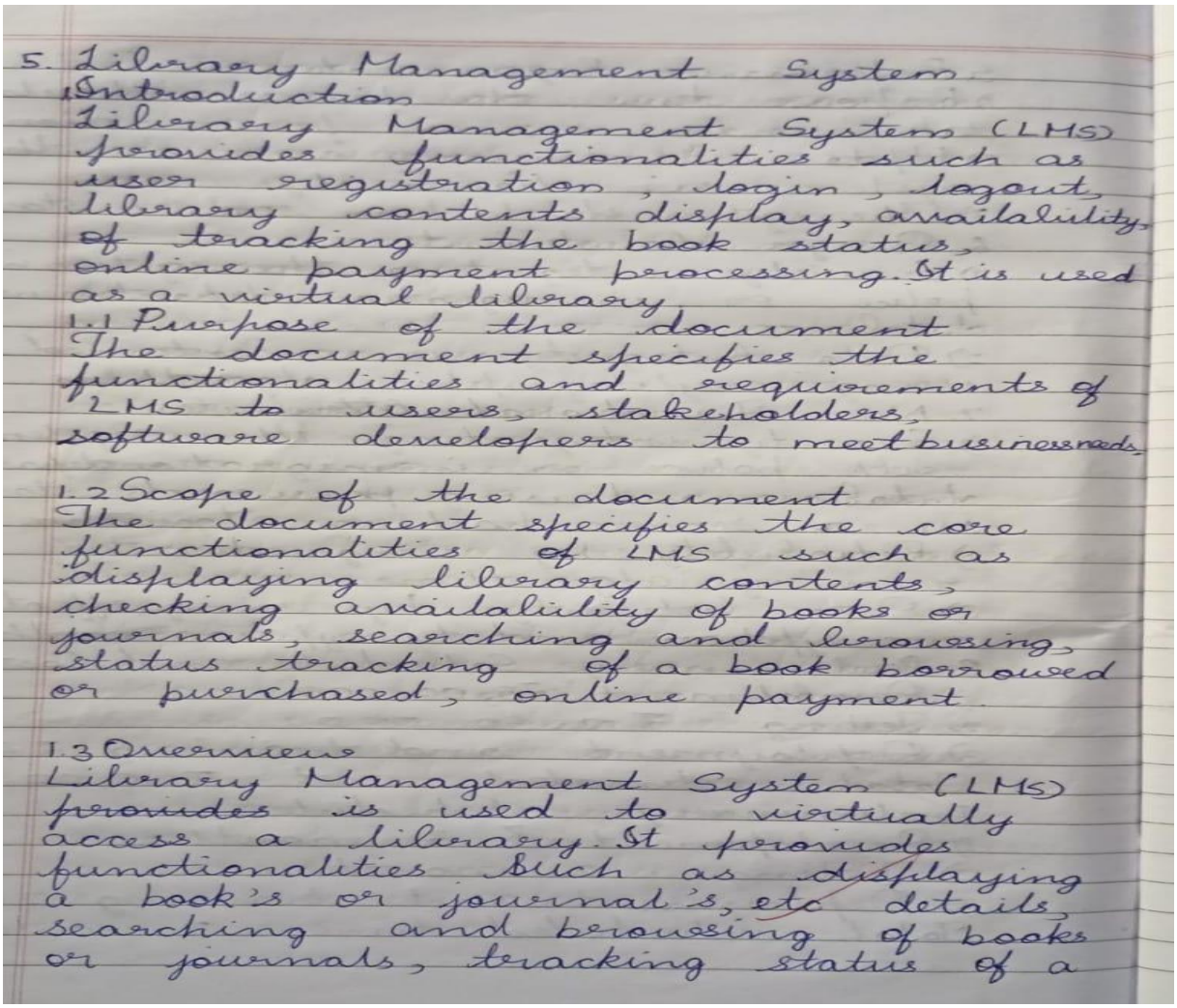
This diagram illustrates the credit card processing workflow. It starts with the Customer initiating a transaction by entering card details. The Merchant System then validates these details. If valid, an authorization request is sent to the Payment Gateway, and the transaction details are logged. The Payment Gateway processes the request and routes the response back to the Merchant System. Finally, the Merchant System displays the transaction outcome to the Customer. If the card details are invalid, the transaction is rejected.

3. Library Management System

Problem Statement

The problem in library management lies in efficiently managing book inventory, tracking borrowed and returned items, and maintaining accurate records of members and transactions. Manual processes can lead to errors, delays, and difficulty in locating books or monitoring availability. The solution requires a digital system that automates tasks like cataloging, issuing, and returning books, sends notifications for due dates, and provides easy access to information for both librarians and members, ensuring smooth and organized library operations.

SRS-Software Requirements Specification



5. Library Management System

Introduction

Library Management System (LMS) provides functionalities such as user registration, login, logout, library contents display, availability, of tracking the book status, online payment processing. It is used as a virtual library.

1.1 Purpose of the document

The document specifies the functionalities and requirements of LMS to users, stakeholders, software developers to meet businessneeds.

1.2 Scope of the document

The document specifies the core functionalities of LMS such as displaying library contents, checking availability of books or journals, searching and browsing, status tracking of a book borrowed or purchased, online payment.

1.3 Overviews

Library Management System (LMS) provides is used to virtually access a library. It provides functionalities such as displaying a book's or journal's, etc details, searching and browsing of books or journals, tracking status of a

Fig 3.1

borrowed or purchased book or journal, online payment processing.

2. General Description

LMS provides the operations such as:

1) ~~User~~ Displaying the library content
2) Searching and Browsing
3) Displaying status of a book or journal.
4) ~~Tracking~~ & Checking availability of books or journals.
5) Payment Processing: by any online payment methods.

3. Functional Requirements

1) User Registration and login/logout: allows users to register with name, address, contact details and set a password, these credentials will be used while login operation.
2) Display the library contents: subjectwise display of contents.
3) Checking availability: by keeping track of books, stock of books and journals.
4) Status tracking: of borrowed or purchased books or journals
5) Payment Processing: by online payment methods using credit cards, debit cards or UPI, etc.

Fig 3.2

4. Interface Requirements
1) Web Interface: users can use LMS website
2) Responsiveness: LMS can run on any screen.

5. Performance Requirements
1) Storage: system should have scalable storage.
2) Upto 10,000 users can use LMS simultaneously.
3) Response Time: the processes should take time under 1-2 ms.

6. Design constraints
1) Security: using passwords stored in a secure hashing algorithm.
2) Data Integrity: by cryptographic algorithms usage.

7. Non Functional Attributes
1) Reliability: by using disaster recovery algorithms, backup techniques to recover data during calamity.
2) Security: by using passwords
3) Data Integrity: by encryption and decryption techniques.

8. Preliminary Schedule and Cost
- 8 months for development, testing and deployment of system.
- Preliminary cost: ₹10,00,000 for

Fig 3.3

Class Diagram

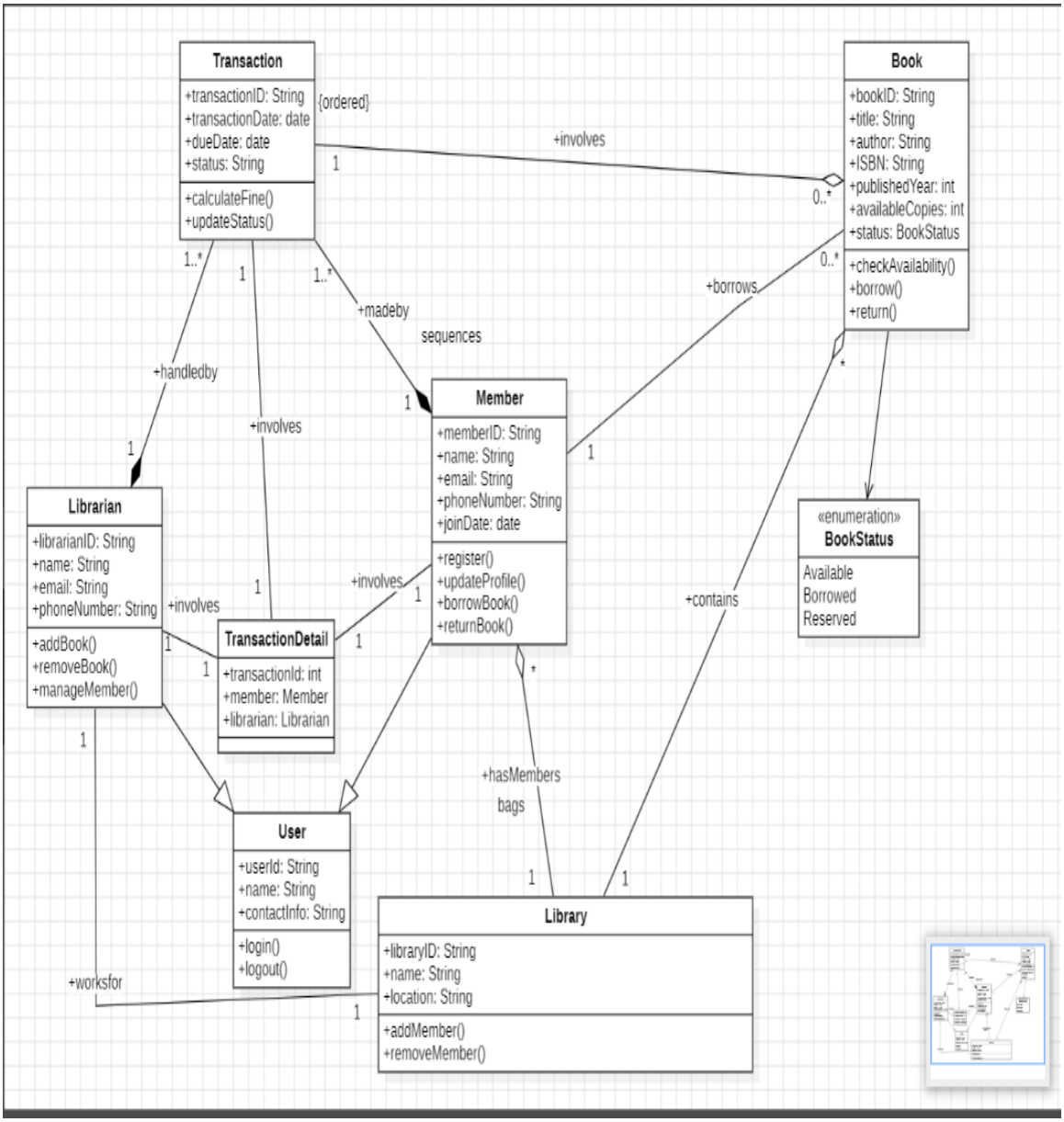


Fig 3.4

This class diagram represents a library system with entities like Users, Members, Librarians, Transactions, Books, and a Library. Users can be Members or Librarians. Members can borrow Books, while Librarians can add, remove, and manage Books and Members. Transactions record borrowing and returning of Books. The Library has a collection of Books and manages its members.

State Diagram

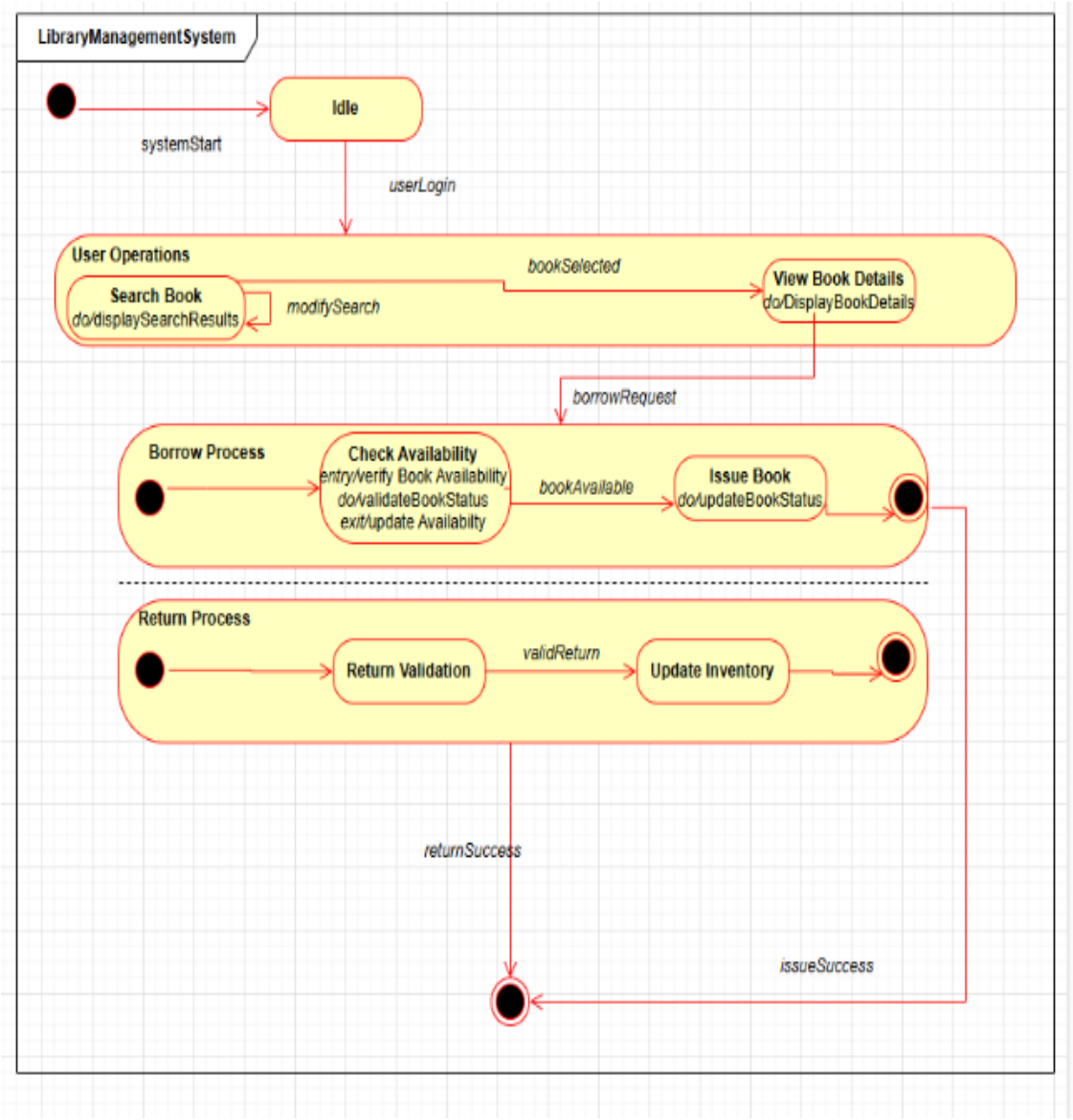


Fig 3.5

This diagram depicts the state diagram of a library management system. It starts with the system being idle. When a user logs in, they can perform user operations like searching for books and viewing book details. The user can then initiate a borrow process, where the system checks book availability and issues the book if available. The return process involves validating the returned book and updating the inventory. Successful issue and return operations lead to the issueSuccess and returnSuccess states, respectively.

Use Case Diagram

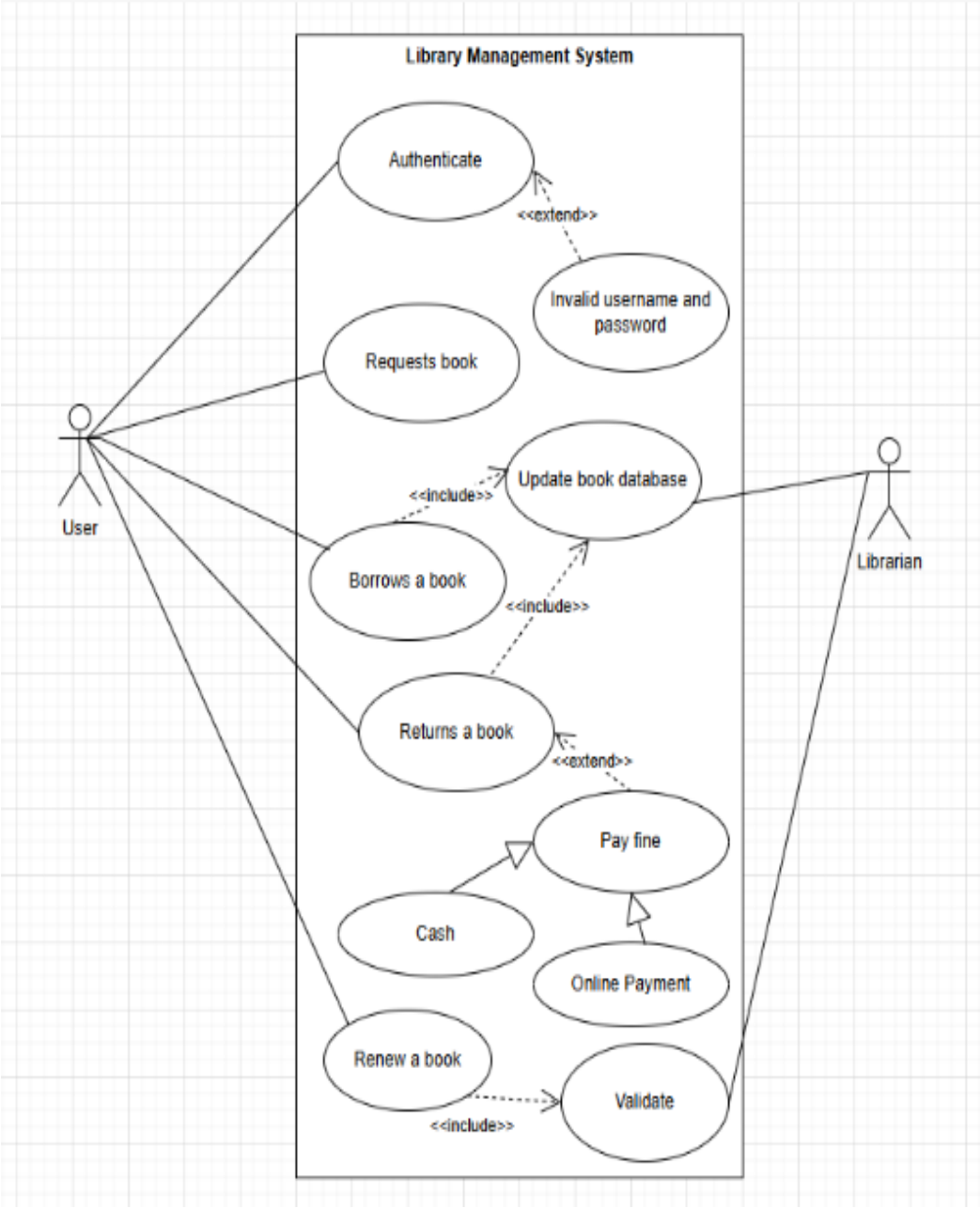


Fig 3.6

Use case: Borrow book

Summary: This use case describes the process of a user borrowing a book from the library.

Actors: User, Librarian

Precondition: The user must be a registered member of the library.

Description:

The user authenticates with the system by providing their credentials (username and password). If the authentication is successful, the user requests to borrow a specific book. The system verifies the book's availability. If available, the system updates the book's status in the database as "borrowed." The user receives the borrowed book. If the book is not available, the system informs the user of the unavailability. If the user returns the book late, a "Pay fine" use case is triggered. The user may renew the book borrowing by requesting a renewal, which involves validation of the renewal request.

Exception:

- **Invalid Credentials:** If the user enters incorrect credentials during authentication, the system will display an error message.
- **Book Unavailable:** If the requested book is not available (already borrowed or lost), the system will inform the user and suggest alternatives.
- **Renewal Denied:** If the renewal request is denied (e.g., due to exceeding the maximum renewal limit), the system will inform the user.

Postcondition:

- The user successfully borrows the book.
- The book's status in the database is updated to "borrowed."
- The user is informed about the borrowing details (e.g., due date).

Sequence Diagram

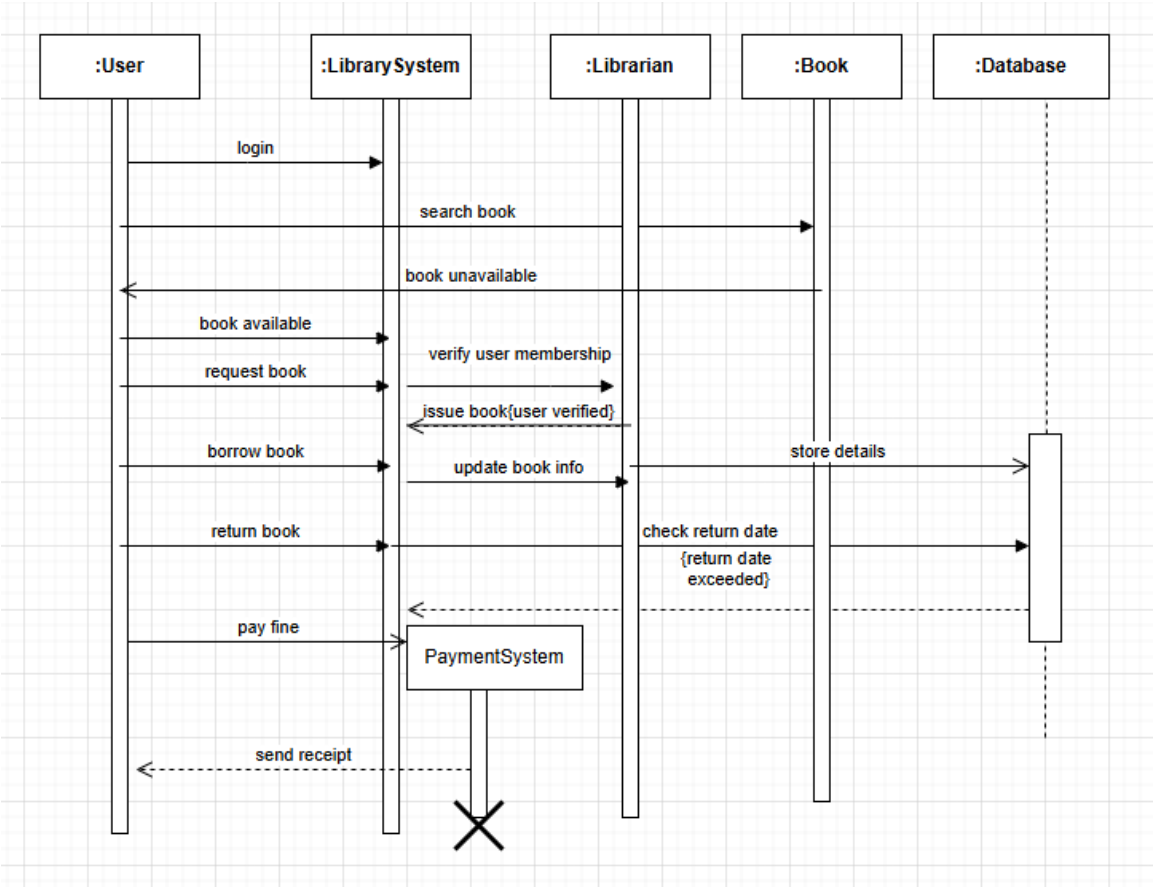


Fig 3.7

This sequence diagram illustrates the interactions between a user, the library system, librarian, books, database and payment system during a book borrowing process. The user starts by logging in and searching for a book. If the book is available, the user requests to borrow it. The system verifies the user's membership and issues the book if eligible. The book's information is updated in the database, and the book is borrowed. When the user returns the book, the system checks the return date. If the book is overdue, a fine is calculated and paid through the payment system. Finally, a receipt is sent to the user.

Activity Diagram

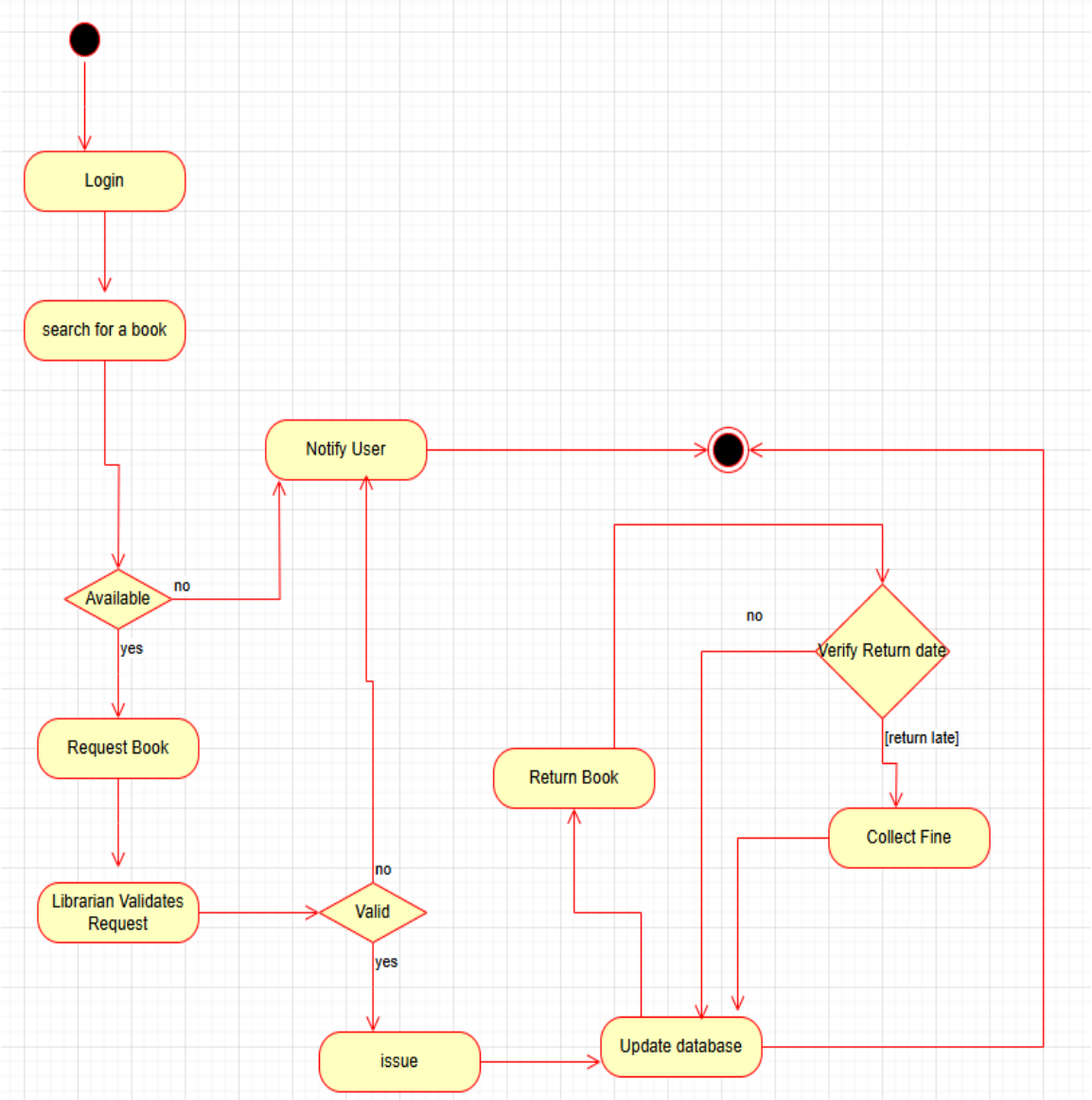


Fig 3.8

This activity diagram illustrates the process of borrowing and returning books in a library. It starts with the "Login" activity. Then, the user searches for a book. If the book is available, the user requests it. If the librarian validates the request, the book is issued, and the database is updated. If the book is not available or the request is invalid, the user is notified. When returning the book, the system verifies the return date. If the book is returned late, a fine is collected.

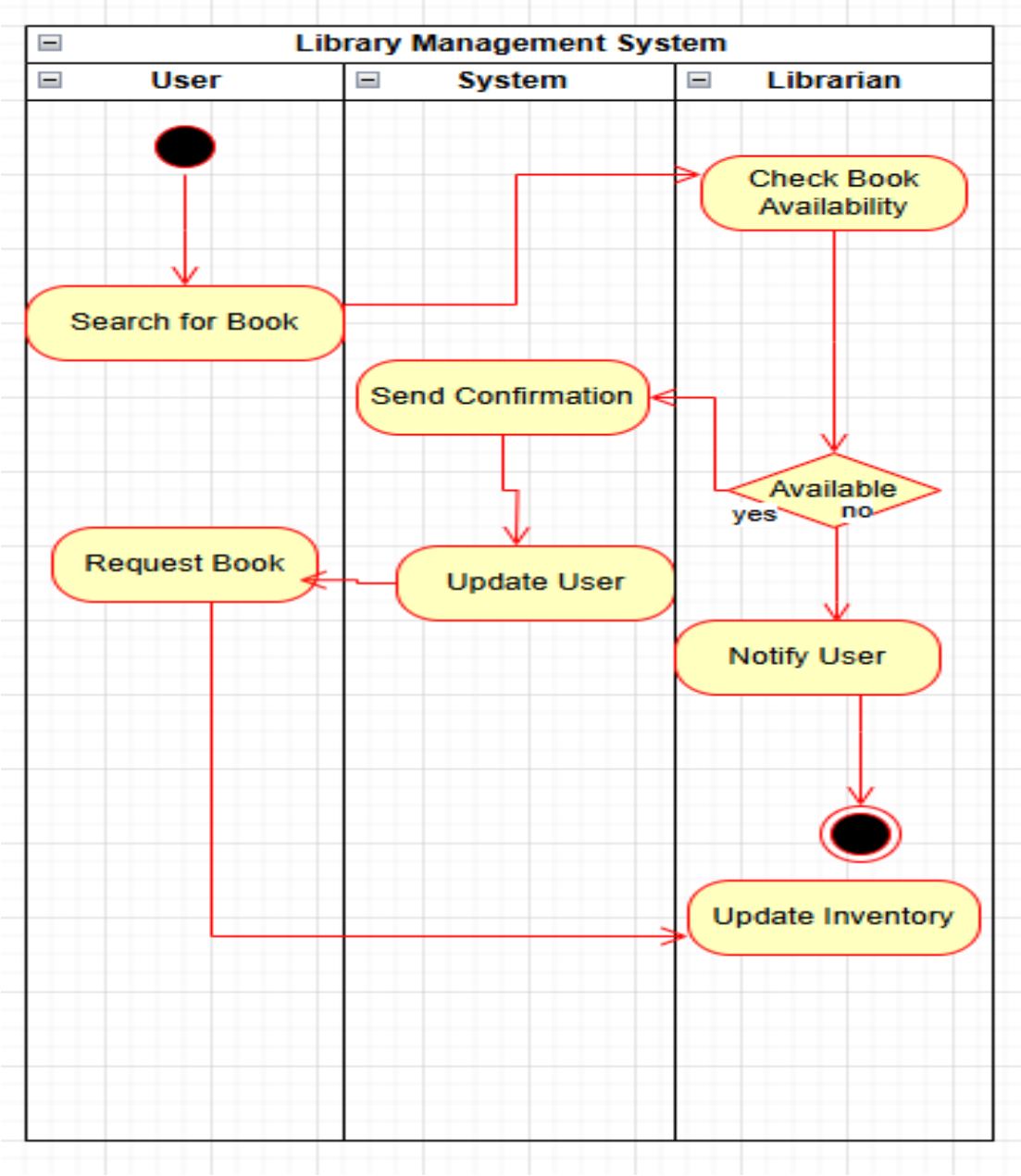


Fig 3.9

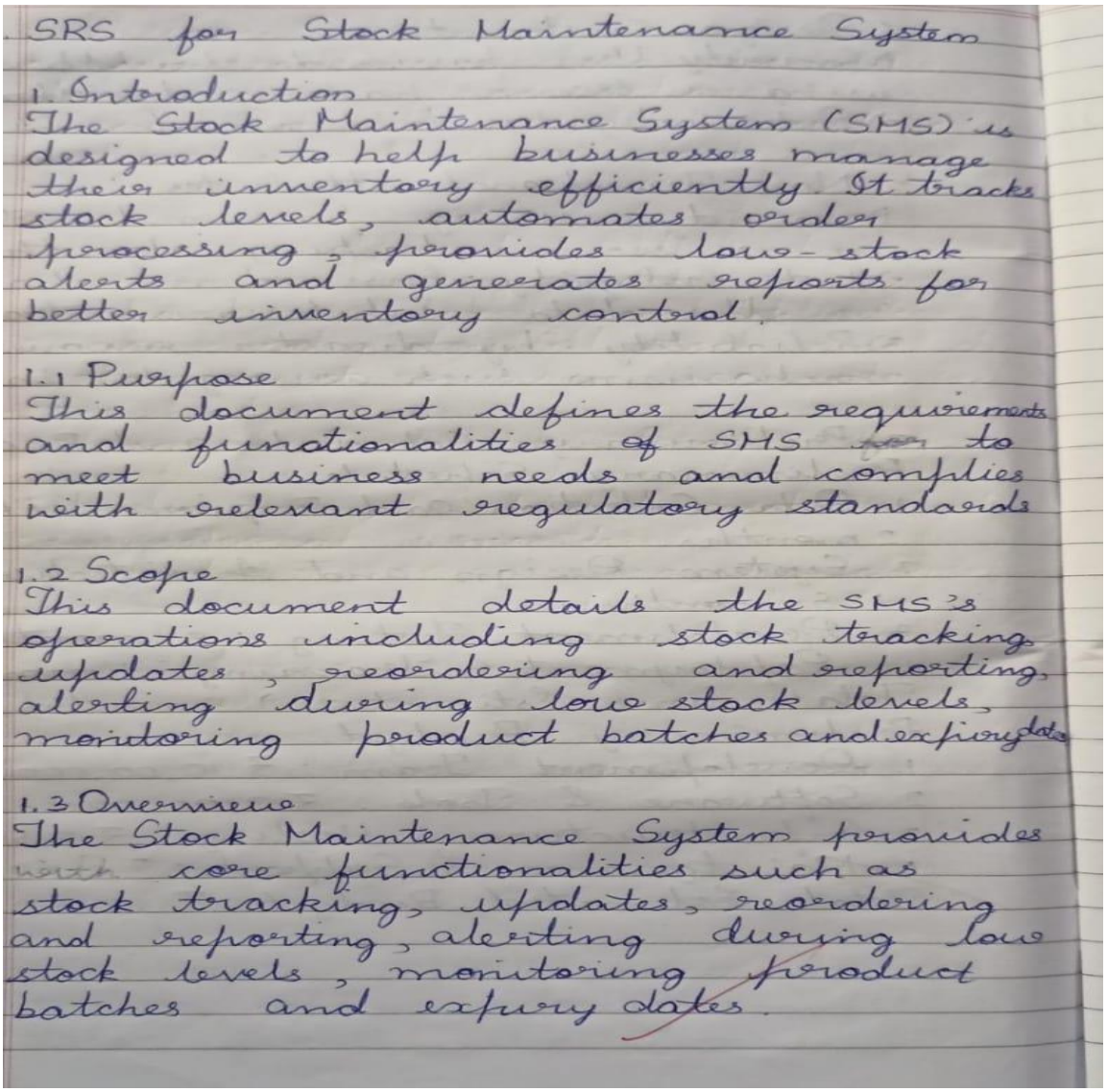
This advanced activity diagram illustrates the process of borrowing a book in a library system. It starts with the User searching for a book. The System then checks the book's availability. If available, the User requests the book, and the System updates the User's information and sends confirmation. If the book is unavailable, the User is notified. Finally, the System updates the inventory.

4.Stock Maintenance System

Problem Statement

A Stock Maintenance System is a software application designed to manage and track inventory within a business or organization. The system aims to streamline the process of monitoring stock levels, managing product orders, updating quantities, and ensuring timely reordering. The primary goal is to ensure smooth stock operations and enhance business efficiency by reducing manual tracking and errors.

SRS-Software Requirements Specification



SRS for Stock Maintenance System

1. Introduction
The Stock Maintenance System (SMS) is designed to help businesses manage their inventory efficiently. It tracks stock levels, automates order processing, provides low-stock alerts and generates reports for better inventory control.

1.1 Purpose
This document defines the requirements and functionalities of SMS to meet business needs and complies with relevant regulatory standards

1.2 Scope
This document details the SMS's operations including stock tracking, updates, reordering and reporting, alerting during low stock levels, monitoring product batches and expiry dates

1.3 Overview
The Stock Maintenance System provides core functionalities such as stock tracking, updates, reordering and reporting, alerting during low stock levels, monitoring product batches and expiry dates.

Fig 4.1

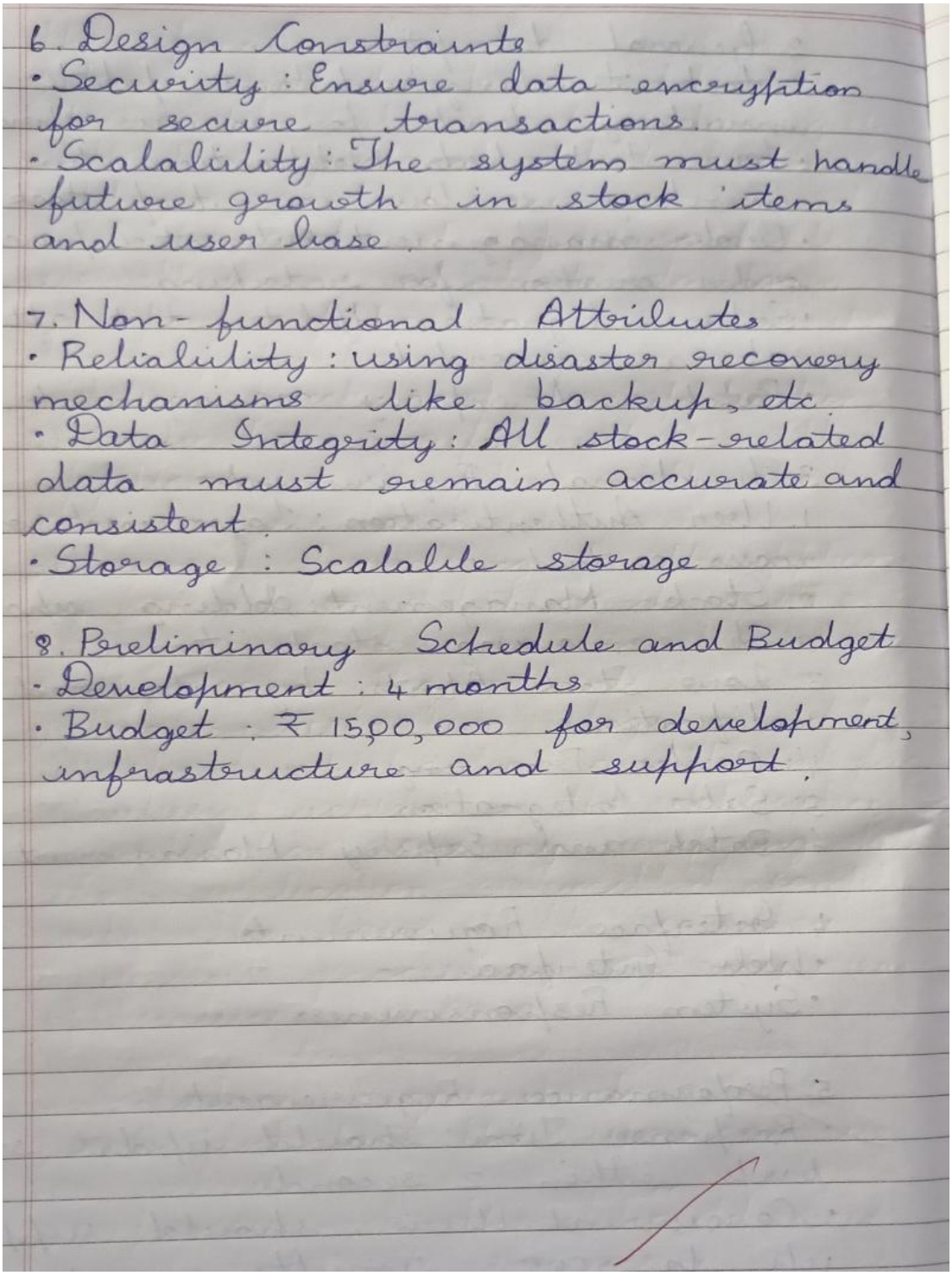
2. General description
- Stock tracking: Real-time updates on inventory levels for various products.
- Low-stock alerts: Notifications when inventory is low or out of stock.
- Order management: Automated purchase order creation for restocking.
- Sales integration: Updates stock after sales are recorded.
- Batch and expiry management.

3. Functional Requirements
1. User Authentication: Login and role-based access control.
2. Stock Management: Adding, updating and removing stock items.
3. Low Stock Alerts: by automated notifications.
4. Order Processing
5. Sales Integration
6. Batch and Expiry Management.

4. Interface Requirements
- Web Interface
- System Responsiveness.

5. Performance Requirements
- Response Time: should update stock levels within 2 seconds.
- Concurrent Users: should support up to 5,000 simultaneous users.

Fig 4.2


6. Design Constraints
- Security : Ensure data encryption for secure transactions.
- Scalability : The system must handle future growth in stock items and user base.

7. Non-functional Attributes
- Reliability : using disaster recovery mechanisms like backup, etc.
- Data Integrity : All stock-related data must remain accurate and consistent.
- Storage : Scalable storage

8. Preliminary Schedule and Budget
- Development : 4 months
- Budget : ₹ 15,00,000 for development, infrastructure and support.

Fig 4.3

Class Diagram

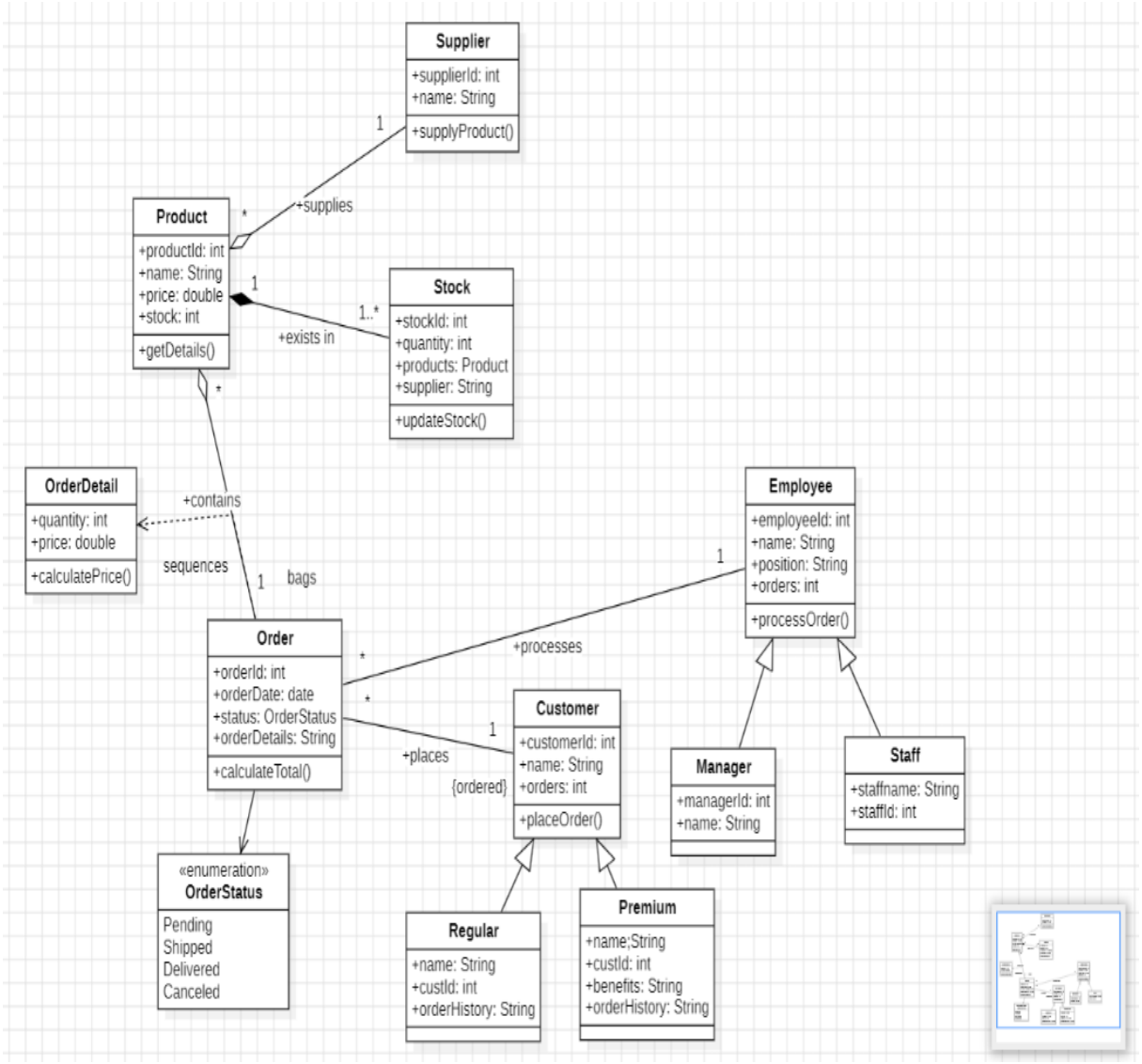


Fig 4.4

This class diagram illustrates a system for managing orders and inventory in a retail context. It includes entities like Customers, Employees (including Managers and Staff), Products, Suppliers, Stock, and Orders. Customers can place orders, which are processed by Employees. Orders consist of OrderDetails, each representing a specific product and quantity. The system tracks product inventory, which is managed by Stock objects that are linked to Products and Suppliers. The diagram also includes attributes and methods for each class, such as order status, order calculation, and stock updates.

State Diagram

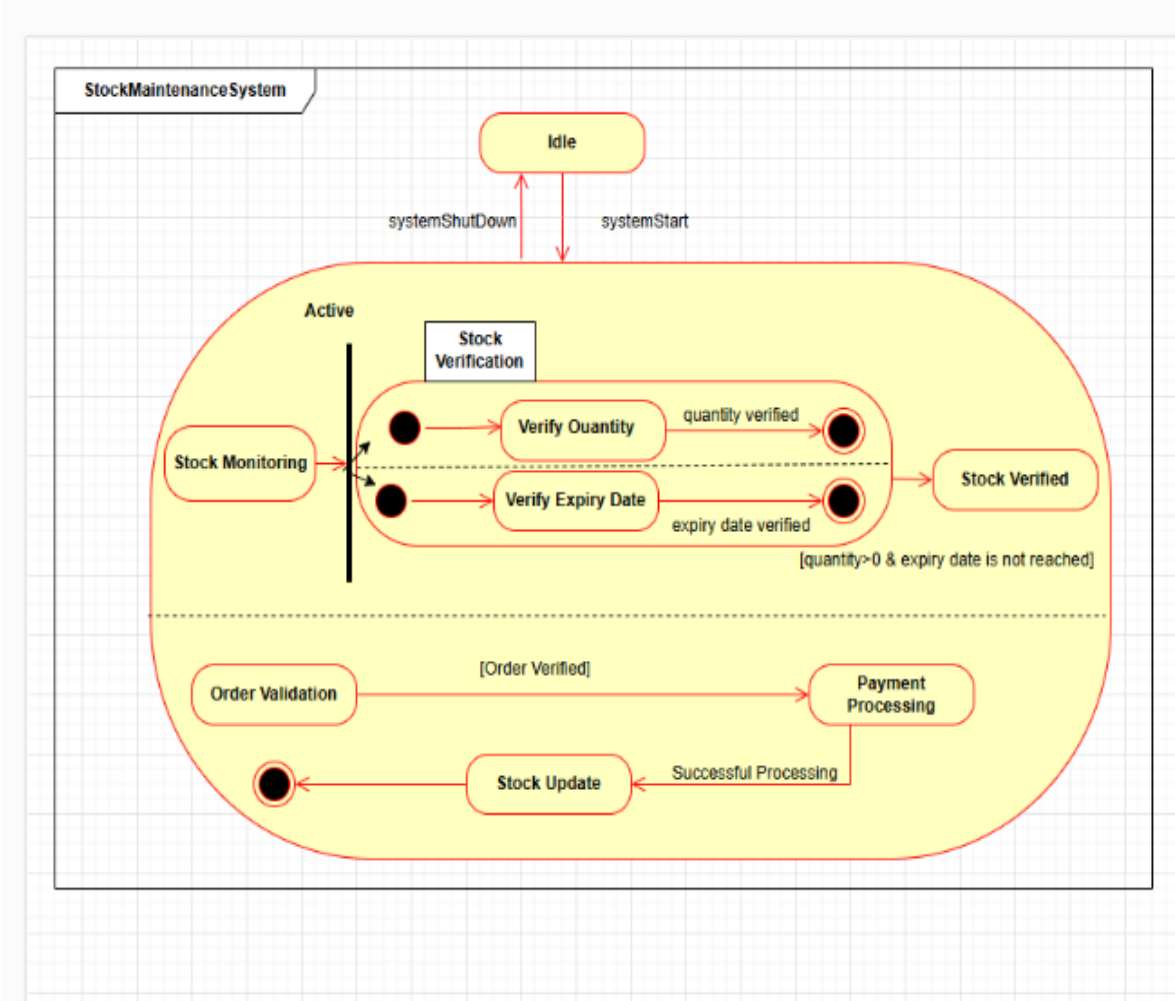


Fig 4.5

This state diagram depicts the workflow of a Stock Maintenance System. The system starts in the "Idle" state and transitions to the "Active" state upon system startup. In the "Active" state, the system enters the "Stock Monitoring" state. Within this state, it verifies the quantity and expiry date of the stock. If both verifications are successful and the stock quantity is greater than zero and the expiry date has not been reached, the system proceeds to the "Order Validation" state. After successful order validation, the system enters the "Payment Processing" state. Finally, upon successful payment processing, the system updates the stock and returns to the "Stock Monitoring" state. The system can transition back to the "Idle" state upon system shutdown.

Use Case Diagram

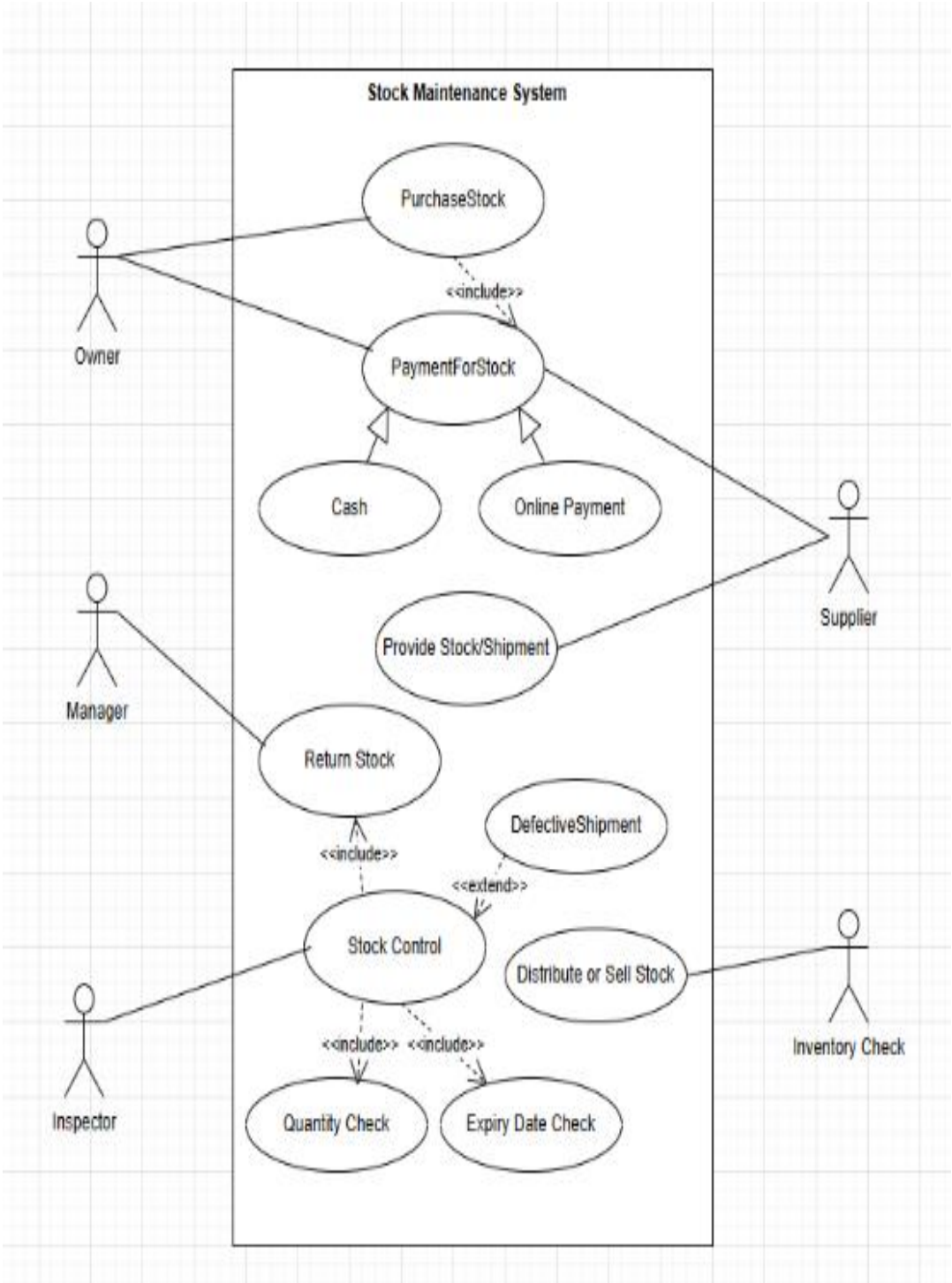


Fig 4.6

Use Case: Stock Maintenance System

Summary: This use case describes the process of acquiring stock for the business.

Actors: Owner, Manager, Supplier, Inventory Checker, Inspector

Precondition: The business has identified a need to purchase new stock.

Description:

The process begins when the Owner initiates the purchase of stock. The Owner selects the required stock items and identifies a suitable Supplier. The Manager then negotiates the purchase order with the Supplier, including details like quantity, price, and delivery date. The Supplier provides the stock/shipment as per the agreed-upon terms. The Inventory Checker verifies the quantity and quality of the received stock, while the Inspector checks for any defects or damage. If any defects are found, the "Defective Shipment" use case is triggered. The payment for the stock is made either in cash or through online payment. Once the payment is processed, the stock is added to the inventory and is now available for distribution or sale.

Exception:

- The Supplier may not be able to fulfil the order due to stock unavailability or other reasons.
- The received stock may have defects or damage.
- The payment for the stock may be declined.

Post-condition:

- The business has acquired the required stock.
- The inventory is updated with the new stock.
- Payment for the stock is made successfully.

Sequence Diagram:

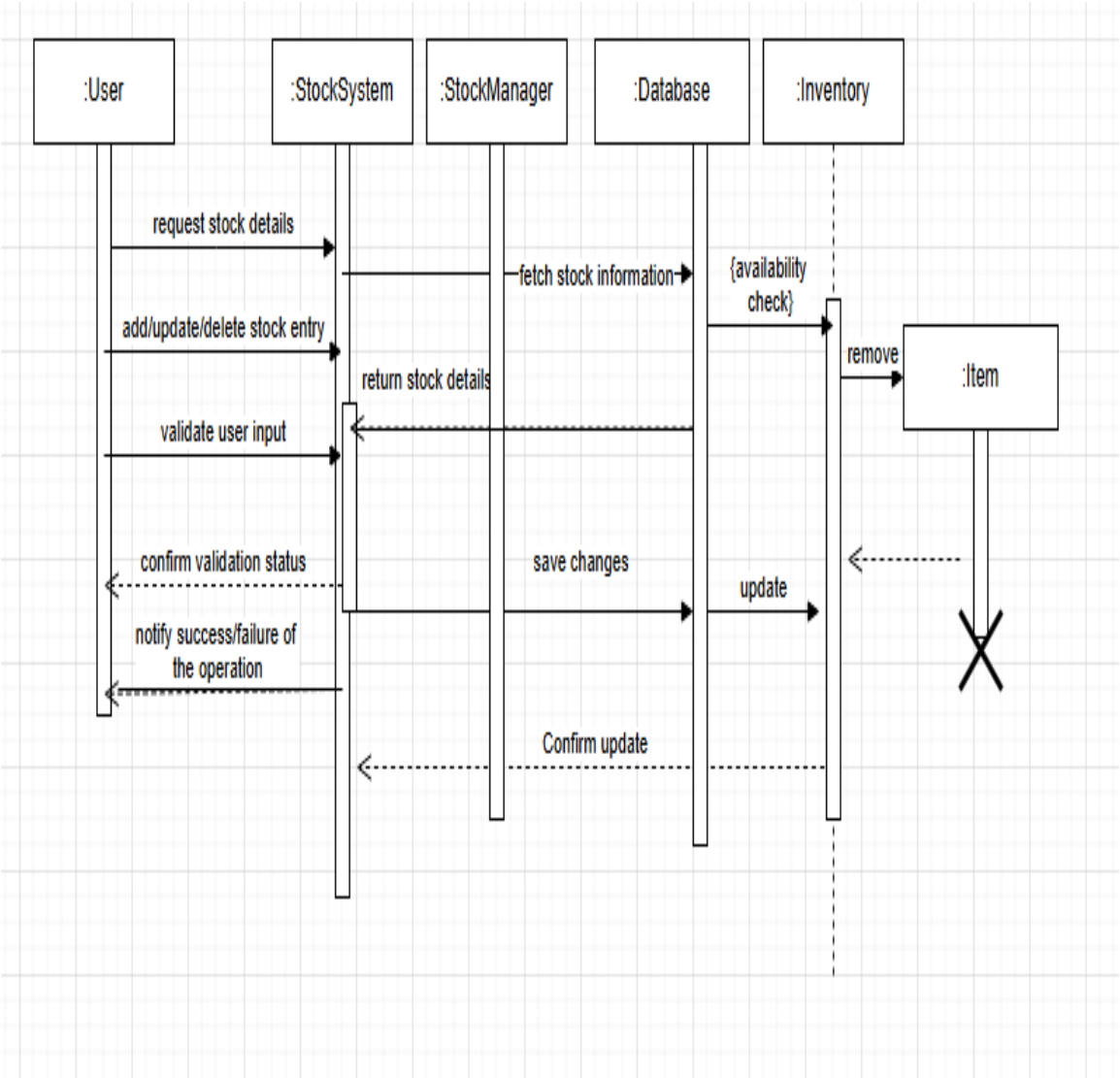


Fig 4.7

This sequence diagram depicts the process of managing stock in a system. A User requests stock details, and the Stock System fetches the information from the Database, including availability checks. The Stock Manager can then add, update, or delete stock entries. The system validates the user input and notifies the user of the success or failure of the operation. Changes are saved to the Database, and the Inventory is updated accordingly.

Activity Diagram:

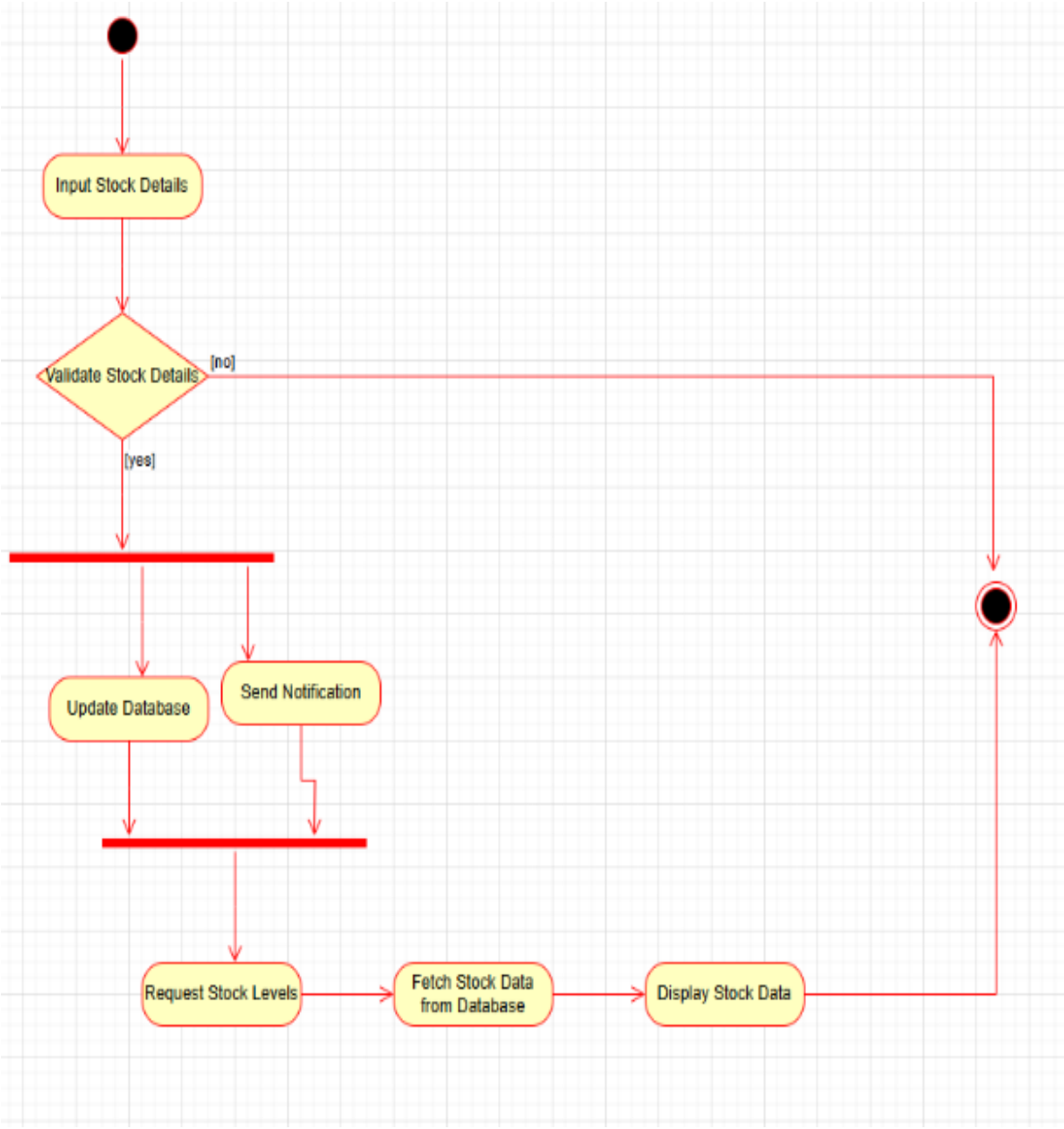


Fig 4.8

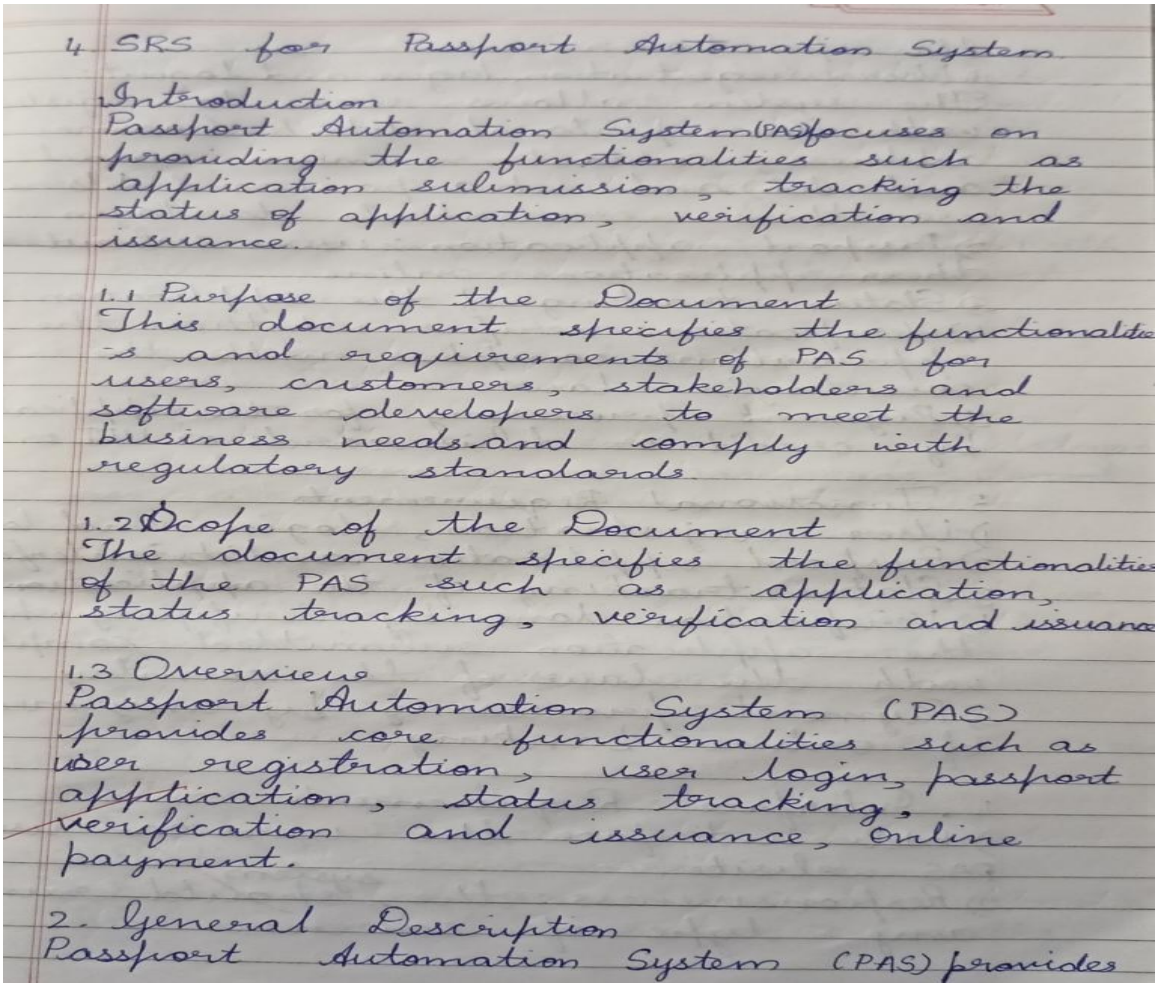
This activity diagram illustrates the process of managing stock levels in a system. It starts with the input of stock details, which are then validated. If the details are valid, the database is updated, and a notification is sent. The user can then request stock levels, and the system fetches the data from the database and displays it to the user.

5.Passport Automation System

Problem Statement

A Passport Automation System is designed to streamline the application, processing, and issuance of passports. The system aims to reduce manual intervention by automating the entire workflow, including application submission, document verification, payment processing, and status tracking. It should ensure efficient handling of personal data, improve processing speed, minimize errors, and provide applicants with real-time updates on their application status. By integrating with government databases and verification systems, the solution should enhance security and accuracy while offering a user-friendly interface for applicants to apply, track, and receive their passports.

SRS-Software Requirements Specifications



4 SRS for Passport Automation System

Introduction
Passport Automation System(PAS)focuses on providing the functionalities such as application submission, tracking the status of application, verification and issuance.

1.1 Purpose of the Document
This document specifies the functionalities and requirements of PAS for users, customers, stakeholders and software developers to meet the business needs and comply with regulatory standards.

1.2 Scope of the Document
The document specifies the functionalities of the PAS such as application, status tracking, verification and issuance

1.3 Overview
Passport Automation System (PAS) provides core functionalities such as user registration, user login, passport application, status tracking, verification and issuance, online payment.

2. General Description
Passport Automation System (PAS) provides

Fig 5.1

the operations:
1) User registration, login and logout: The system allows users to register with basic personal details (name, address, contact number, email and dob) and set a password. Users use these credentials to login.
2) Passport application: users submit their applications online.
3) Status tracking of the application
4) Verification: verifies whether the application complies with the country's laws.
5) Payment processing: for the application by credit or debit cards or UPI or etc.

3. Functional Requirements
1) User registration, login and logout
2) Passport application and its interface
3) Status tracking of the application
4) Verification: done by verifying whether the application submitted complies with the laws of the country.
5) Payment processing: by any means of online banking.

4. Interface Requirements
1) Web interface: the users can use PAS website.
2) Responsiveness: the system should run on any type of screens.

Fig 5.2

5. Performance Requirements
1) Response time : the processes should take under 1-2 ms for execution.
2) Upto 10,000 users can use the system simultaneously.
3) The system should have scalable storage.
6. Design Constraints
1) Security : integrating the system with police or immigration department databases
2) Data Integrity : is maintained by using cryptographic algorithms.
7. Non Functional Attributes
1) Security : integrating the system with police or immigration department databases and using passwords stored in a secure hashing algorithm.
2) Reliability : by using disaster recovery algorithms.
3) Data Integrity : by using effective encryption and decryption methods.
8. Preliminary Schedule and Cost.
1) Development : ₹5,00,000
2) Testing : ₹ 2,50,000
3) Deployment and training : ₹ 2,50,00
Total estimated cost : ₹ 10,00,000
Preliminary Schedule
1) Development : 3 months
2) Testing : 2 months
3) Deployment and training : 2 month
Total scheduled time = 7 months.

Fig 5.3

Class Diagram

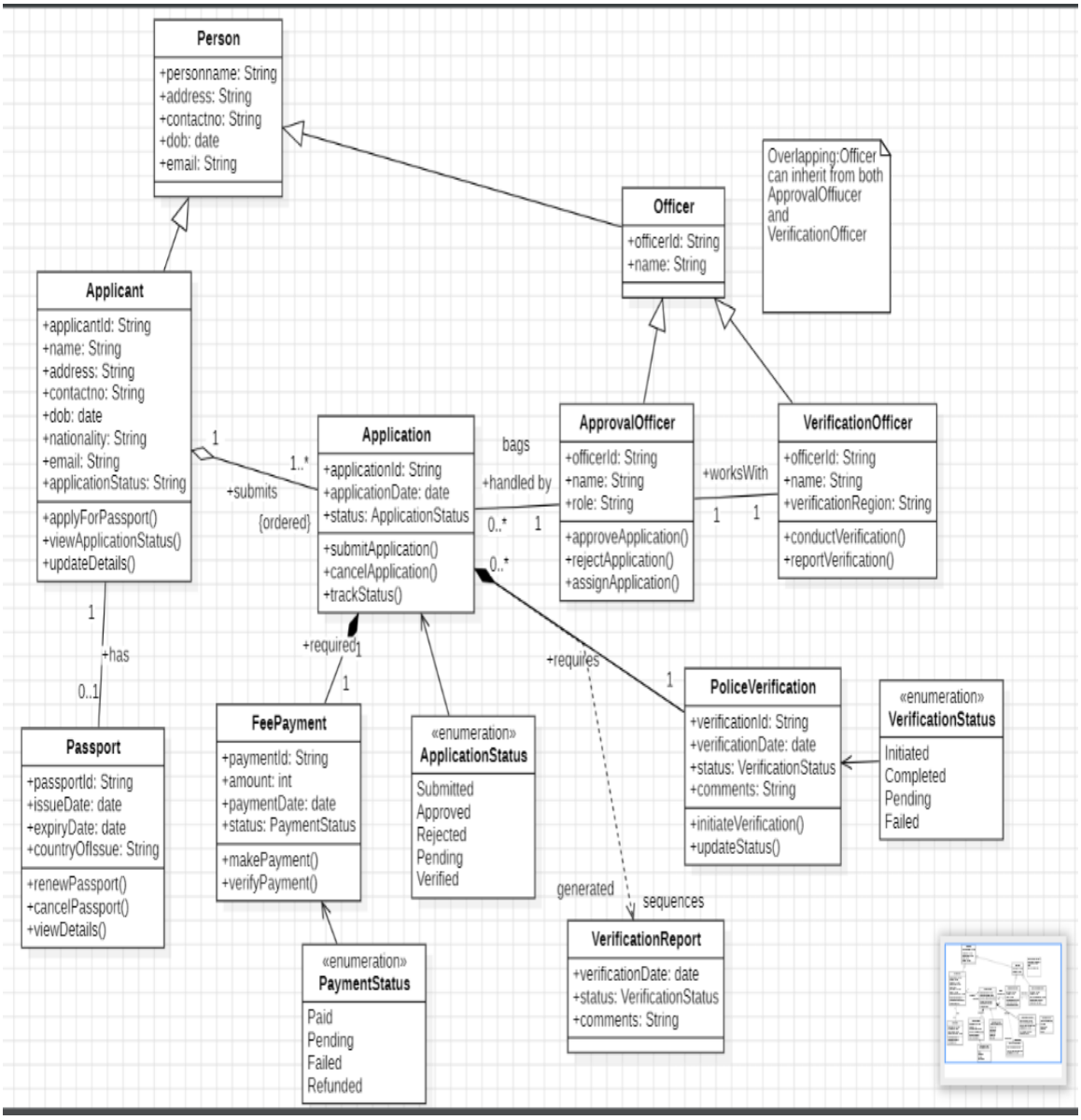


Fig 5.4

This diagram models a passport application system. It includes entities like Person, Applicant, Officer, ApprovalOfficer, VerificationOfficer, Application, Passport, FeePayment, and VerificationReport. Applicants submit applications for passports, which are then processed by Officers. ApprovalOfficers review and approve applications, while VerificationOfficers conduct background checks. The system tracks the status of applications and generates reports on the verification process.

State Diagram

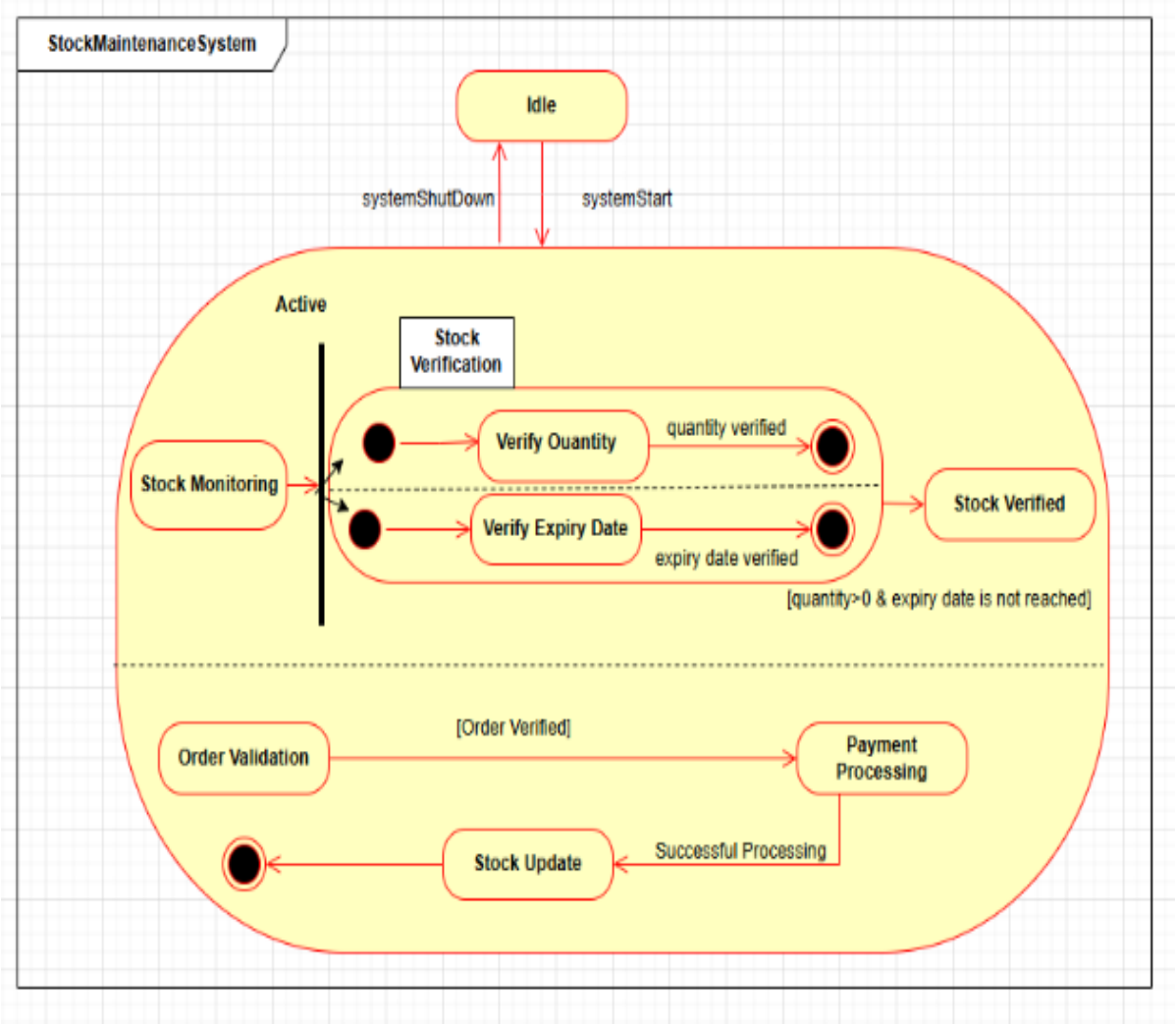


Fig 5.6

This state diagram depicts the workflow of a Stock Maintenance System. The system starts in the "Idle" state and transitions to the "Active" state upon system startup. In the "Active" state, the system enters the "Stock Monitoring" state. Within this state, it verifies the quantity and expiry date of the stock. If both verifications are successful and the stock quantity is greater than zero and the expiry date has not been reached, the system proceeds to the "Order Validation" state. After successful order validation, the system enters the "Payment Processing" state. Finally, upon successful payment processing, the system updates the stock and returns to the "Stock Monitoring" state. The system can transition back to the "Idle" state upon system shutdown.

Use Case Diagram

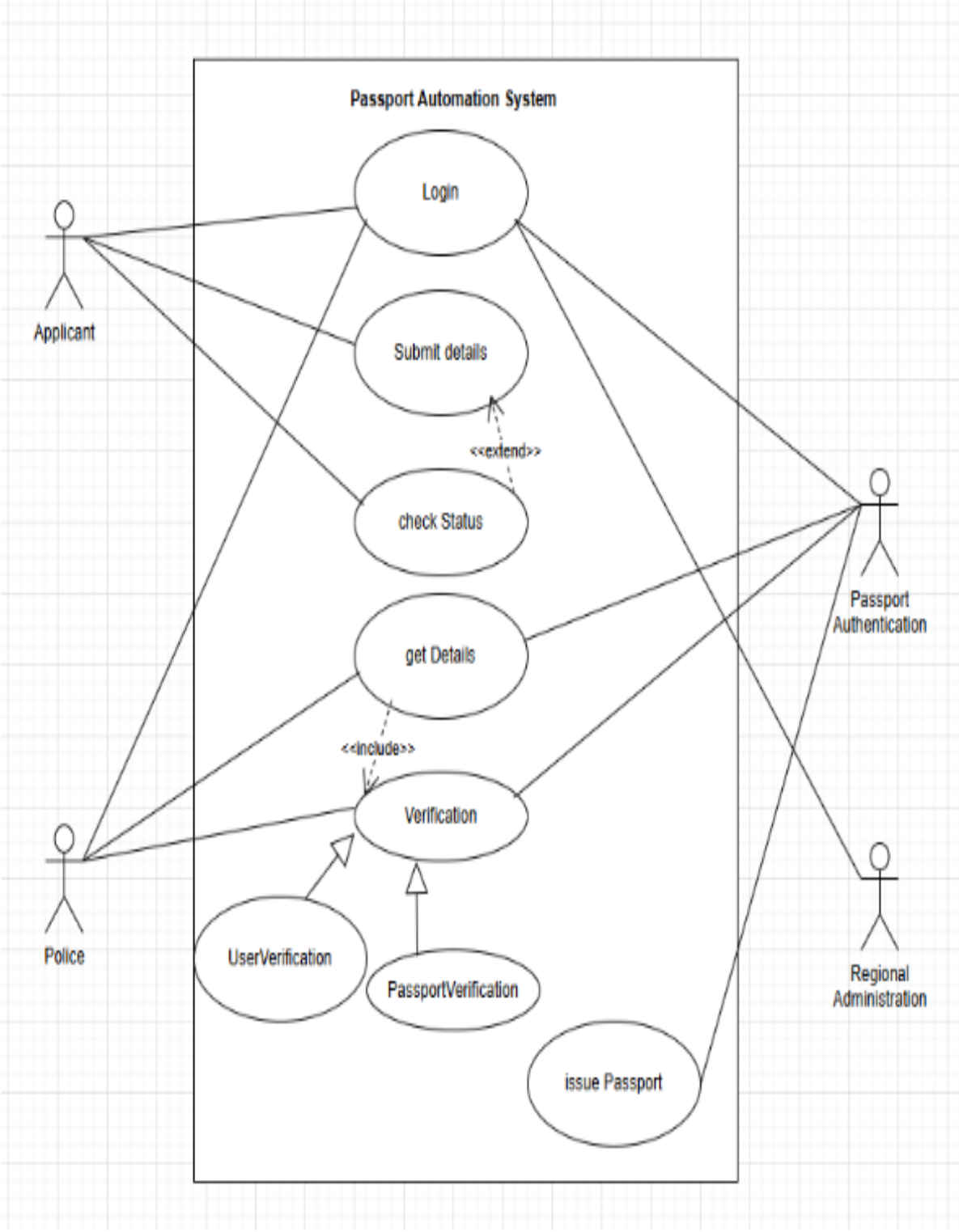


Fig 5.7

Use Case: Passport Automation System

Summary: This use case describes the process of an applicant applying for a passport.

Actors: Applicant, Passport Authentication, Police, Regional Administration

Precondition: The applicant is eligible to apply for a passport.

Description:

The process begins when the Applicant logs in to the system. The Applicant then submits their personal details and supporting documents for the passport application. The system validates the submitted information.

The "Verification" use case is then initiated. This includes "UserVerification" and "PassportVerification". In UserVerification, the Police conduct background checks on the applicant. In PassportVerification, the Regional Administration verifies the authenticity of the applicant's documents.

If the verification process is successful, the system issues the passport. The applicant can then check the status of their application at any time using the "check Status" use case.

Exception:

- If the submitted information is incomplete or invalid, the system will display an error message and prompt the applicant to correct the information.
- If the verification process reveals any discrepancies or issues, the application may be rejected.

Postcondition:

- The applicant's passport application is processed successfully.
- The passport is issued to the applicant.

Sequence Diagram

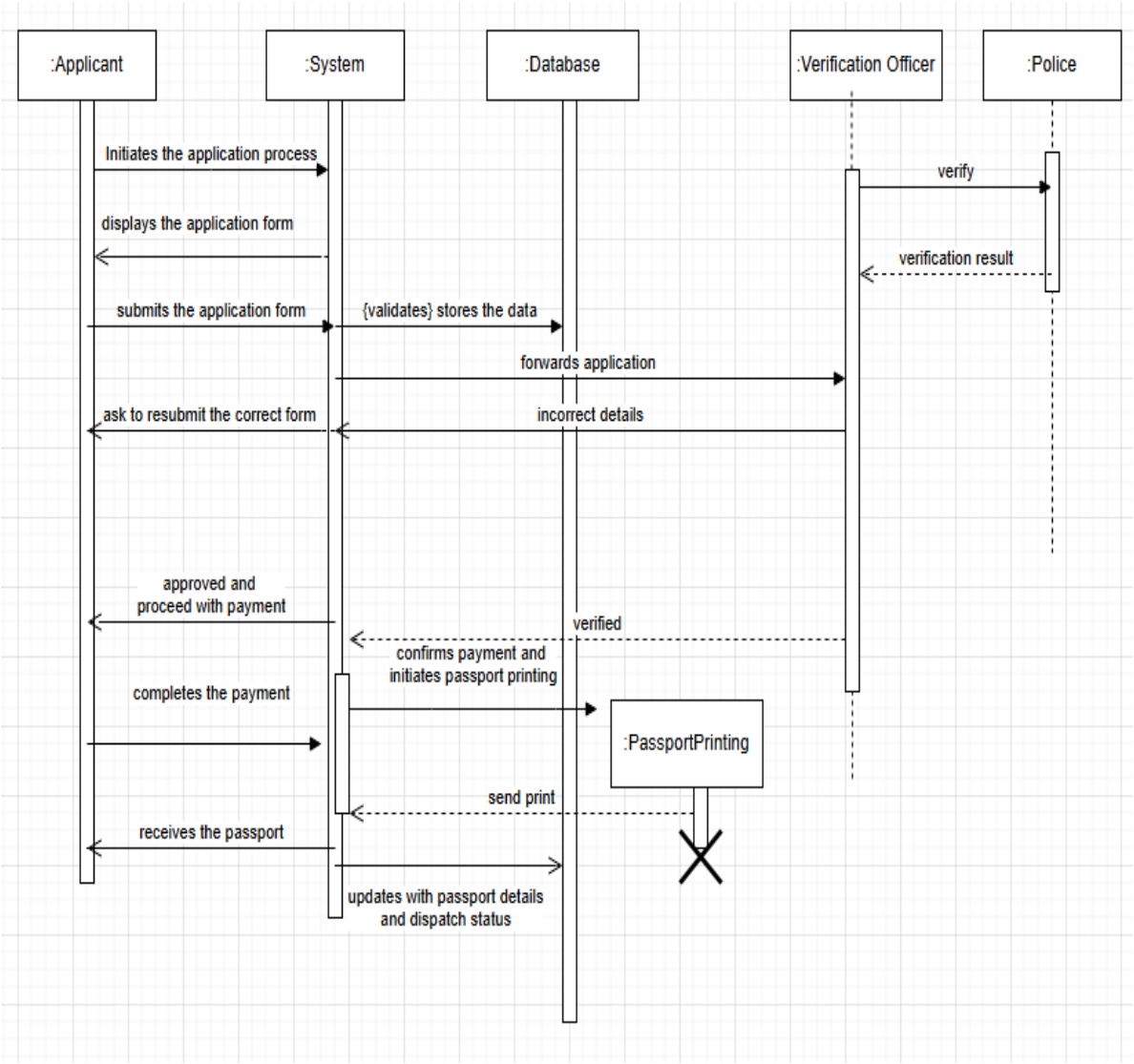


Fig 5.8

This sequence diagram illustrates the process of a passport application. An Applicant initiates the process by requesting an application form. The System displays the form, and the Applicant submits the completed form. The System validates the data and stores it in the Database. The System then forwards the application to the Verification Officer, who verifies the information. If the details are incorrect, the Applicant is asked to resubmit. Once verified, the System confirms payment and initiates passport printing. The Passport Printing process completes, and the System updates the Database with passport details and dispatch status. Finally, the Applicant receives the passport.

Activity Diagram

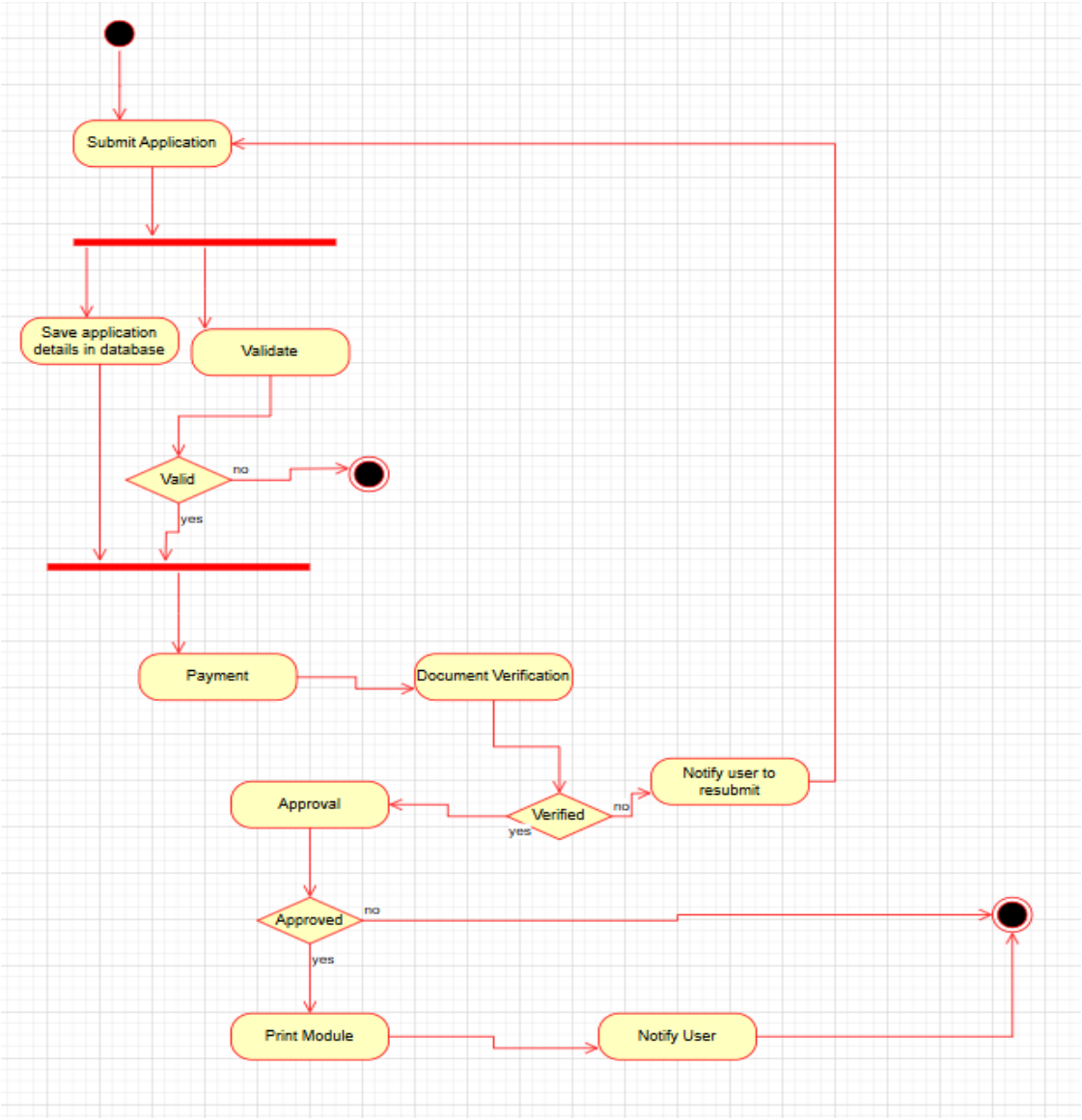


Fig 5.9

This diagram illustrates the process of a passport application. It starts with the submission of the application, followed by saving the details in the database and validation. If the application is valid, payment is processed, and then the document verification takes place. After successful verification and approval, the passport is printed, and the user is notified. If any stage fails, the user is notified to resubmit or take necessary actions.