B.M.S. COLLEGE OF ENGINEERING BENGALURU

Autonomous Institute, Affiliated to VTU

Lab Record

Object Oriented Modelling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

SETU MISHRA

1BM22CS250

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 Modelling (23CS5PCOOM) laboratory has been carried out by Setu Mishra (1BM22CS250) during the 5th Semester Oct24-Jan2025.

Signature of the Faculty In charge:
Sunayana S
Assistant Professor
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

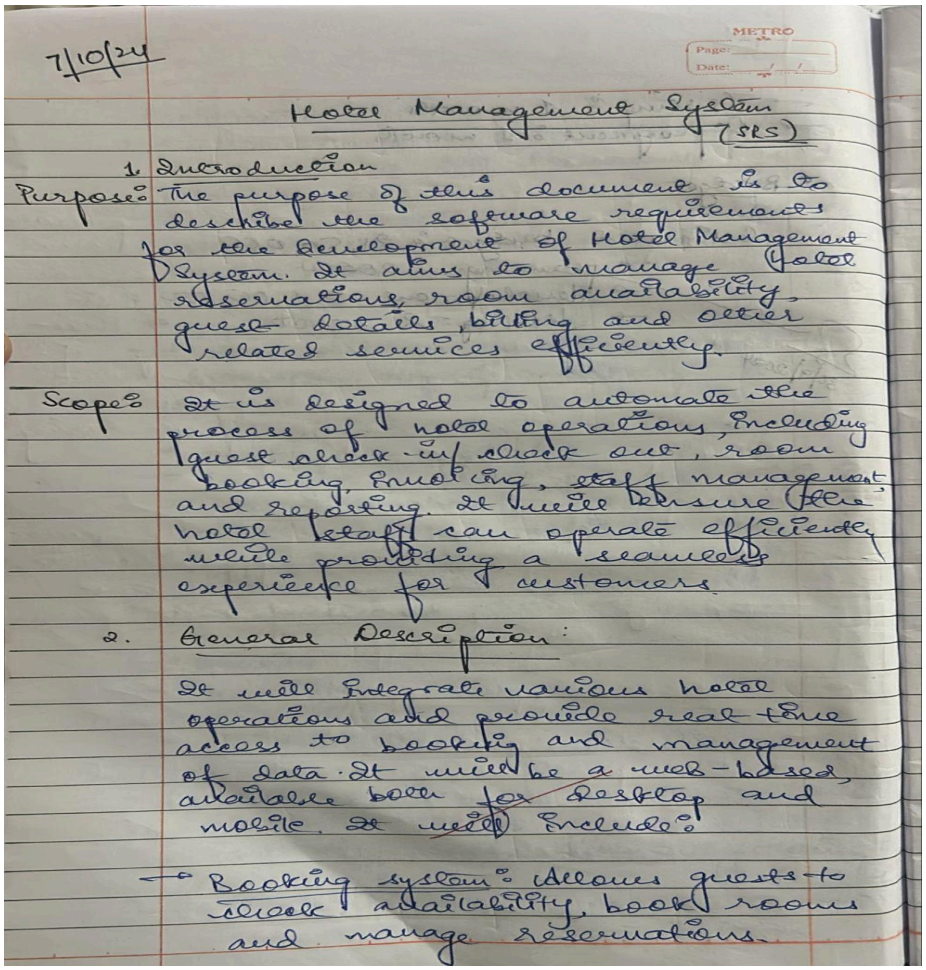
Table of Contents

1. Hotel Management System

1.1 Problem Statement:

The hotel management system aims to streamline the operations of a hotel, focusing on reservation, guest management, and resource allocation. It enables efficient booking of rooms, allowing both online and offline reservations while managing room availability and preventing double bookings. The system handles guest information, tracks check-ins and check-outs, and supports personalized services. Billing and payments are processed smoothly, with invoicing for rooms and additional services. Staff management is also integrated, ensuring smooth coordination between departments like housekeeping and the front desk. Additionally, the system tracks maintenance requests, helping ensure that all facilities are in good working order. It generates real-time reports for management to monitor room occupancy, revenue, and staff performance, ultimately improving operational efficiency and guest satisfaction.

1.2 SRS-Software Requirements Specification:



7/10/24

METRO

Page:

Date: / /

Hotel Management System (SRS)

1. Introduction

Purpose: The purpose of this document is to describe the software requirements for the development of Hotel Management System. It aims to manage hotel reservations, room availability, guest details, billing and other related services efficiently.

Scope: It is designed to automate the process of hotel operations, including guest check-in/ check out, room booking, invoicing, staff management and reporting. It will ensure the hotel staff can operate efficiently while providing a seamless experience for customers.

2. General Description:

It will integrate various hotel operations and provide real-time access to booking and management of data. It will be a web-based, available both for desktop and mobile. It will include:

- Booking system: Allows guests to check availability, book rooms and manage reservations.

→ Check in/check out system: Supports guest check-in and check-out with payment processing.

→ Room service management: Allows guests to place room service requests and track status.

→ Billing system: Automated invoicing for guest services, room fees and other charges.

→ Admin dashboard: Allows hotel staff to manage rooms, bookings and customer feedback.

3. Functional Requirements:

→ Room Booking: Users can check availability, book rooms and cancel reservations.

→ Payment processing: Integration with payment gateways for secure online transactions.

→ Room Service: Guests can request additional services and view the status of their requests.

→ User Management: Admins can

be able to handle 10000 concurrent users without performance degradation

→ Error Rate: Should not exceed 0.01% of all operations

→ Resource Usage: The software must efficiently use memory and CPU resources.

7. Design Constraints:

→ Software constraints - The system must be built using Linux, Mysql etc.

→ Hardware Constraints - Should support deployment on cloud infrastructure

8. Schedule :- → Phase 1: 1 month (Requirement Gathering and Analysis)

Phase 2: 4 months (Design and Development)

Phase 3: 1 month (Testing)

Phase 4: 15 days (Deployment)

9. Budget: Estimated budget: Rs 1,50,000

planning: ₹ 20,000

development: ₹ 1,00,000

testing: ₹ 30,000

be able to handle 10000 concurrent users without performance degradation

→ Error Rate: Should not exceed 0.01% of all operations

→ Resource Usage: The software must efficiently use memory and CPU resources.

7. Design Constraints:

→ Software constraints - The system must be built using Linux, Mysql etc.

→ Hardware Constraints - Should support deployment on cloud infrastructure

8. Schedule: → Phase 1: 1 month (Requirement Gathering and Analysis)

Phase 2: 4 months (Design and Development)

Phase 3: 1 month (Testing)

Phase 4: 15 days (Deployment)

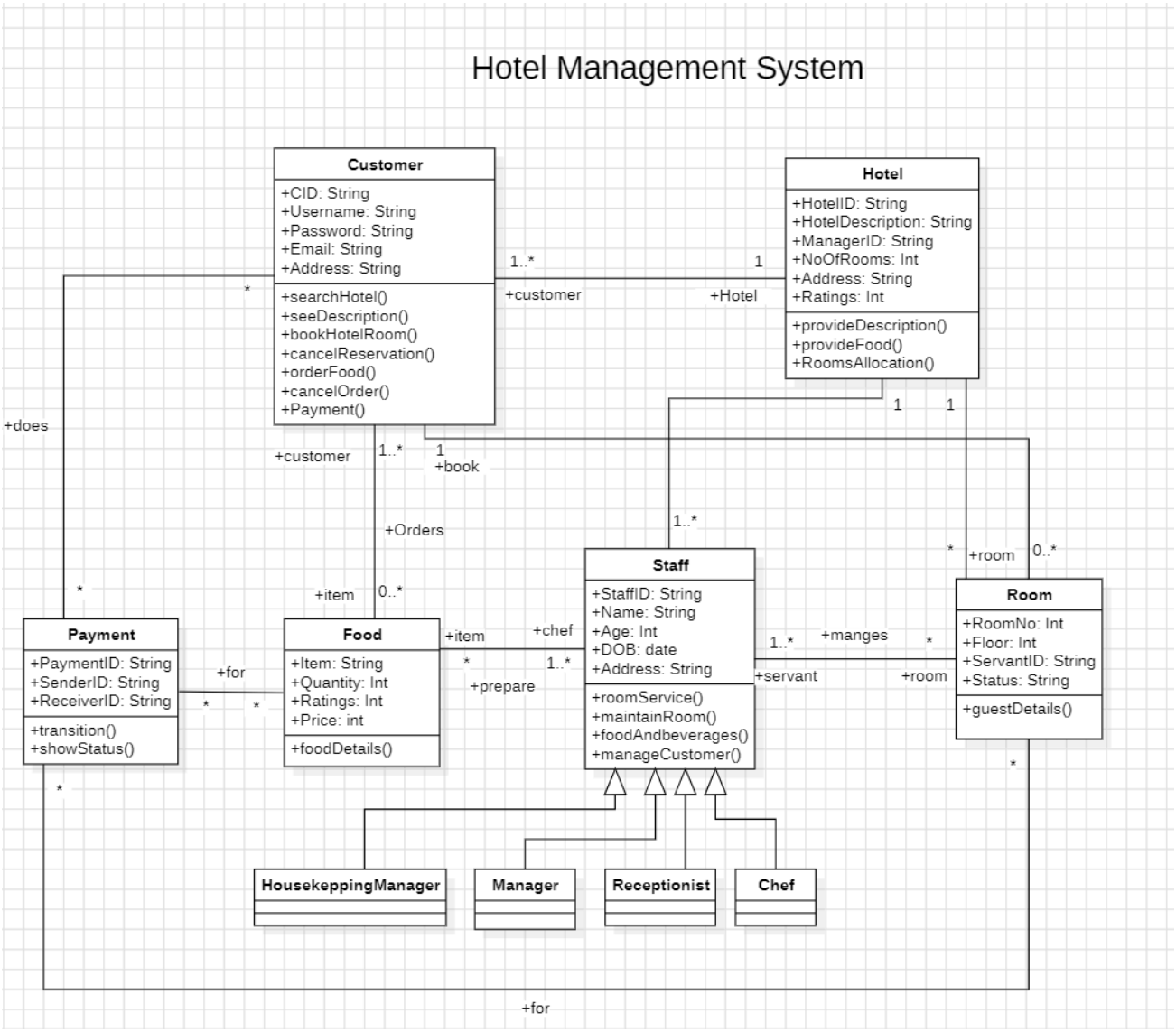
9. Budget: Estimated budget: Rs 1,50,000

planning: ₹ 20,000

development: ₹ 1,00,000

testing: ₹ 30,000

1.3 Class Diagram:



Entities:

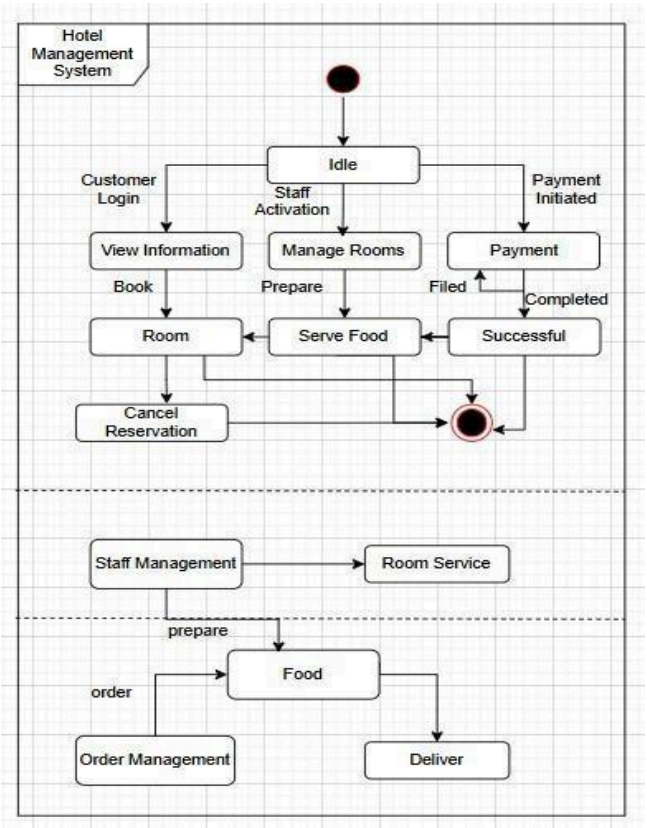
- Customer: Represents guests with attributes like ID, username, password, email, and address. They can perform actions such as searching hotels, booking rooms, cancelling reservations, and ordering food.
- Hotel: Represents individual hotels with attributes like ID, description, manager ID, number of rooms, address, and ratings. Hotels provide descriptions, food services, and manage room allocations.
- Room: Represents individual rooms within a hotel with attributes like room number, floor, servant ID, status, and guest details.

- Staff: Represents hotel staff members with attributes like ID, name, age, date of birth, address, and roles like chef, housekeeper, manager, or receptionist.
- Payment: Represents payment transactions with attributes like payment ID, sender and receiver IDs, and methods for transition and status display.
- Food: Represents food items with attributes like item name, quantity, ratings, price, and a method to retrieve detailed information.

Relationships:

- A customer can book one or many rooms, and a room can be booked by one or many customers.
- A hotel has many rooms, and a room belongs to a hotel.
- A hotel can be managed by a manager, and a manager can manage one hotel.
- Staff can be assigned to rooms (servants), prepare food (chefs), manage housekeeping, or act as receptionists.
- A customer can make one or many payments, and a payment can be made by one customer.
- Food can be ordered for a customer, and a customer can order one or many food items.

1.4 State Diagram:



Customer States:

- Customer Login: The customer logs in to the system.
- View Information: The customer views information related to rooms, prices, etc.
- Book: The customer books a room.
- Room: The customer is assigned a room.
- Cancel Reservation: The customer cancels their reservation.

Hotel Staff States:

- Staff Activation: The staff member activates their account.
- Idle: The staff member is waiting for a task.
- Manage Rooms: The staff member manages the availability and reservation status of rooms.
- Prepare: The staff member prepares the room for the customer.
- Serve Food: The staff member serves food to the customer.

Payment States:

- Payment Initiated: The payment process has begun.
- Payment Completed: The payment has been completed successfully.

Other States:

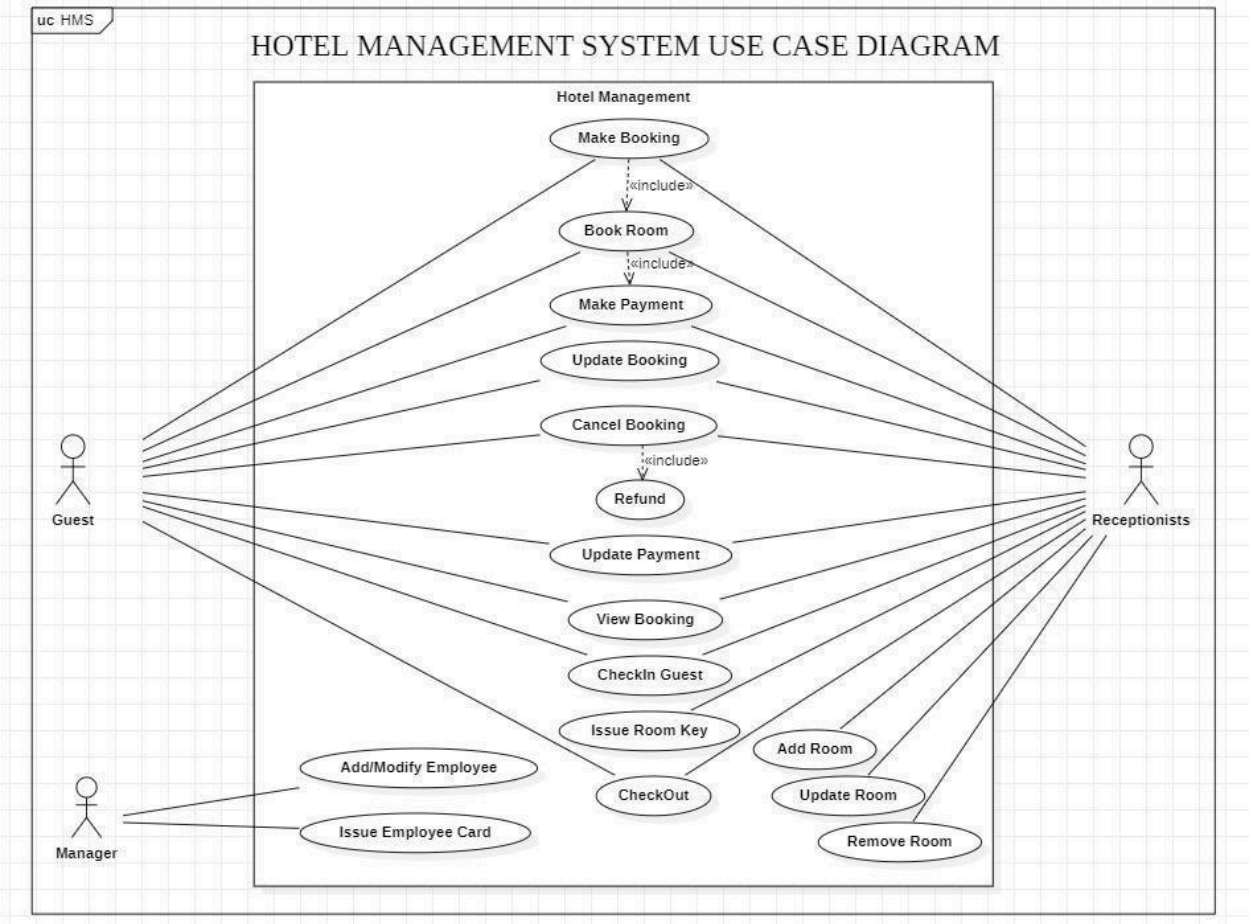
- Filed: The request is filed.
- Successful: The process was successful.
- Order Management: The staff manages the order for room service.
- Room Service: The staff prepares and delivers room service.
- Prepare: The staff prepares the food for room service.
- Food: The food is ready to be delivered.
- Deliver: The food is delivered to the customer.

1.5 Use-Case Diagram:

Actors:

- Guest: Guests interact with the system to make bookings, make payments, update bookings, cancel bookings, and receive refunds.

- Manager: Managers have access to features such as adding/modifying employee records, issuing employee cards, and managing rooms.
- Receptionists: Receptionists perform actions like checking in guests, issuing room keys, checking out guests, updating room information, and removing rooms.



Use Cases:

- Hotel Management: This encompasses a range of activities, including making bookings, updating bookings, and managing payments.
- Make Booking: Guests can make bookings through the system.
- Book Room: This is a specific function within booking, allowing guests to reserve a room.
- Make Payment: Guests can make payments for their bookings.
- Update Booking: Guests can update their bookings, such as changing dates or room preferences.
- Cancel Booking: Guests can cancel their bookings.
- Refund: The system facilitates refunds for canceled bookings.

- Update Payment: Receptionists can update payment information.
- View Booking: Receptionists can view booking details.
- Check in Guest: Receptionists check in guests upon arrival.
- Issue Room Key: Receptionists provide room keys to guests.
- Checkout: Receptionists process guest checkouts.
- Add Room: Managers can add new rooms to the hotel.
- Add/Modify Employee: Managers can add or modify employee records.
- Issue Employee Card: Managers can issue employee cards.
- Update Room: Receptionists can update room information.
- Remove Room: Receptionists can remove rooms from the system.

1.6 Sequence Diagram:

Actors

- Actor

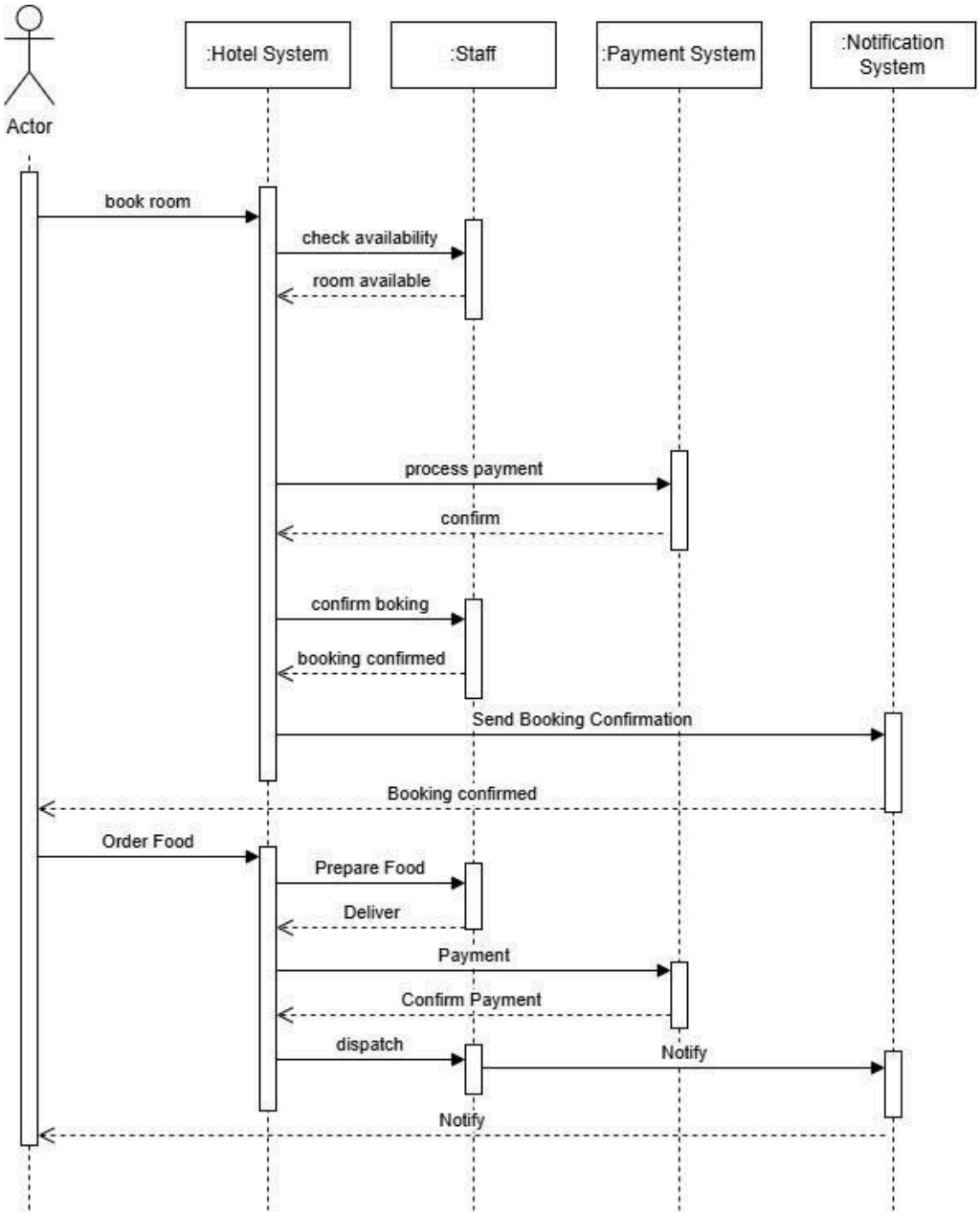
Systems

- :Hotel System
- :Staff
- :Payment System
- :Notification System

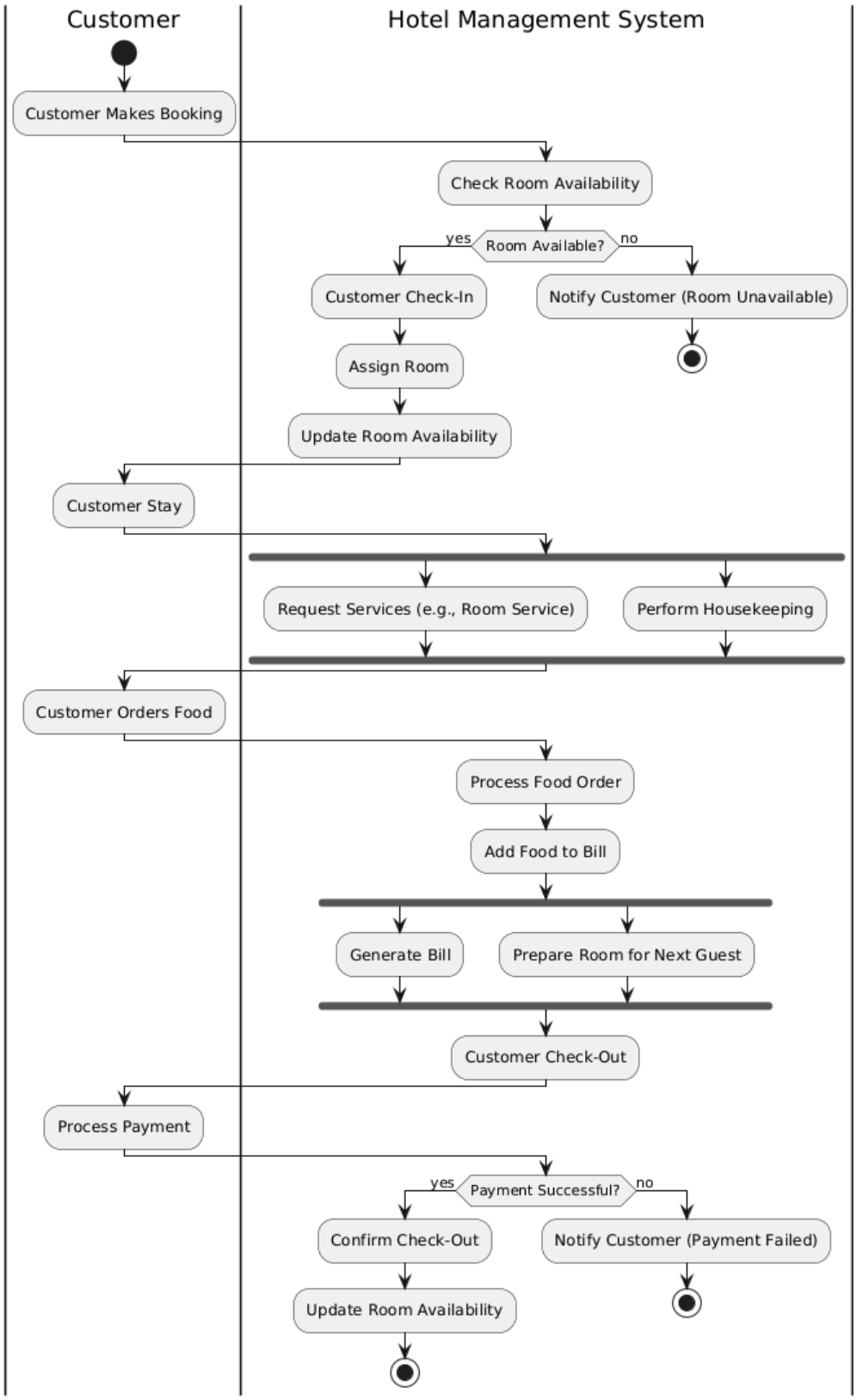
Sequence

- Actor -> :Hotel System : book room
- :Hotel System -> :Staff : check availability
- :Staff --> :Hotel System : room available
- :Hotel System -> :Payment System : process payment
- :Payment System --> :Hotel System : confirm
- :Hotel System -> :Staff : confirm boking
- :Staff --> :Hotel System : booking confirmed
- :Hotel System -> :Notification System : Send Booking Confirmation
- :Notification System --> Actor : Booking confirmed

- Actor -> :Hotel System : Order Food
- :Hotel System -> :Staff : Prepare Food
- :Staff --> :Hotel System : Deliver
- :Hotel System -> :Payment System : Payment
- :Payment System --> :Hotel System : Confirm Payment
- :Hotel System -> :Staff : dispatch
- :Staff --> :Notification System : Notify
- :Notification System --> Actor : Notify



1.7 Activity Diagram:



Customer Interaction:

- Booking: The customer initiates the process by making a booking.
- Check-In: Upon arrival, the customer checks in. The system verifies room availability and assigns a room.
- Stay: The customer can request services during their stay, such as room service or housekeeping.
- Ordering Food: The customer can order food. The system processes the order and adds it to the bill.
- Check-Out: When the customer checks out, the system generates a bill, processes payment, and confirms the check-out.

System Processes:

- Room Availability: The system manages room availability, updating it after check-ins, check-outs, and bookings.
- Housekeeping: The system handles housekeeping requests, preparing rooms for the next guests.
- Food Orders: The system processes food orders and bills them to the customer.
- Payment Processing: The system receives and processes payments from customers.
- Check-Out Confirmation: The system confirms check-outs and updates room availability accordingly.

2. Credit Card Processing

2.1 Problem Statement:

The increasing reliance on digital transactions has highlighted the need for a secure and efficient credit card processing system. Current systems often struggle with security vulnerabilities, slow transaction speeds, and poor user experiences, leading to increased fraud, transaction failures, and customer dissatisfaction. The objective is to develop a robust credit card processing system that complies with Payment Card Industry Data Security Standards (PCI DSS) to protect sensitive information while ensuring quick and seamless transactions for both online and in-store purchases. The system should feature an intuitive interface for merchants and customers, support multiple payment methods, and integrate easily with existing business systems. Additionally, it should include fraud detection mechanisms and provide comprehensive reporting tools for businesses to analyze transactions and customer behavior. By addressing these challenges, the proposed system aims to enhance security, improve customer satisfaction, and ultimately drive revenue growth for businesses.

2.2 SRS-Software Requirements Specification:

30/9/24

SRS for credit card system

1. Introduction

1.1 Purpose of this document

The purpose of this document is to provide a comprehensive outline of the functional and non-functional requirements essential for successful project completion. This document aims to ensure that all stakeholders have a clear understanding of the project scope and objectives, which will facilitate effective communication and collaboration among the people during development of the project.

1.2 Scope of this Document

This document is a user-friendly, secure application that ensures spam detection, transaction handling, user account handling, billing and statement generation, card issuance.

2. Functional Requirements

- Transaction processing
  - like purchase, payment, refund
  - transactions must be secure

- fraud detection
  - Detect suspicious activities and recognise patterns.
  - It must generate alerts.

- User Registration
  - User should be able to register by providing KYC documents and personal details.
  - User information is critical and personal info should be validated and stored securely.
- reporting
  - administrators must have access to transaction reports and fraud detection reports
- credit card application
  - The system must perform eligibility checks and when any user applies for a card by submitting the application.

3. Non-functional attributes
   - security
   - usability
   - reliability
   - maintainability

4. Schedule and budget
   - Schedule
     - Specification and requirement gathering : 1 month
     - Design : 2 months
     - Development : 5 months

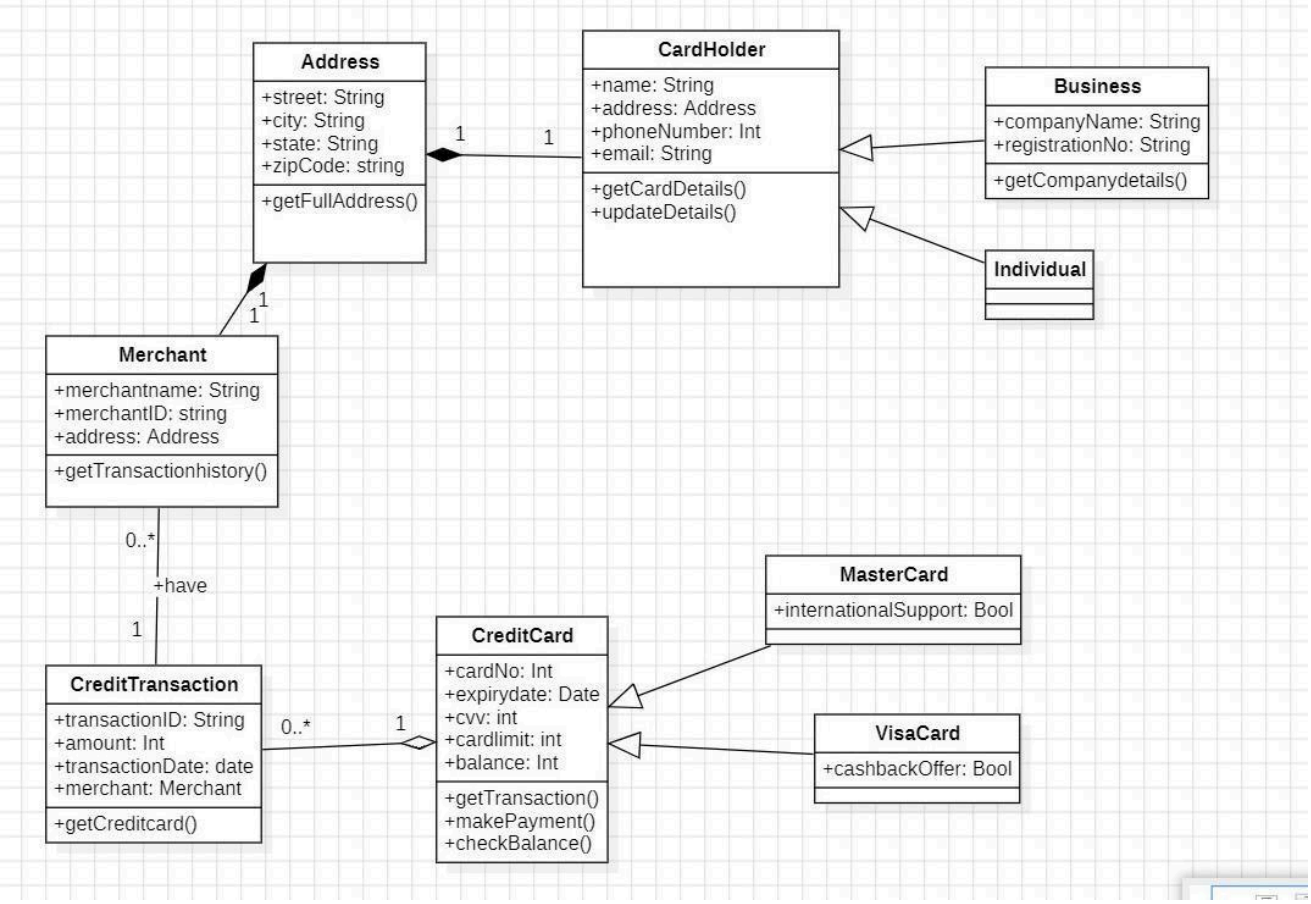
→ Testing : 1 month
→ Deployment : 1 month

- **Budget**

Estimated budget : ₹1,50,000
→ planning = ₹20000
→ development = ₹ 1,00,000
→ testing = ₹ 30,000

30/9/2024

2.3 Class Diagram:



Entities:

- Address: Represents an address with attributes like street, city, state, and zip code.
- CardHolder: Represents a cardholder with attributes like name, address, email, and phoneNumber.
- Merchant: Represents a merchant with attributes like merchantName, merchantID, and address.
- CreditCard: Represents a credit card with attributes like cardNo, expiryDate, cvv, cardLimit, and balance.
- CreditTransaction: Represents a credit transaction with attributes like transactionID, amount, transactionDate, and merchant.

Relationships:

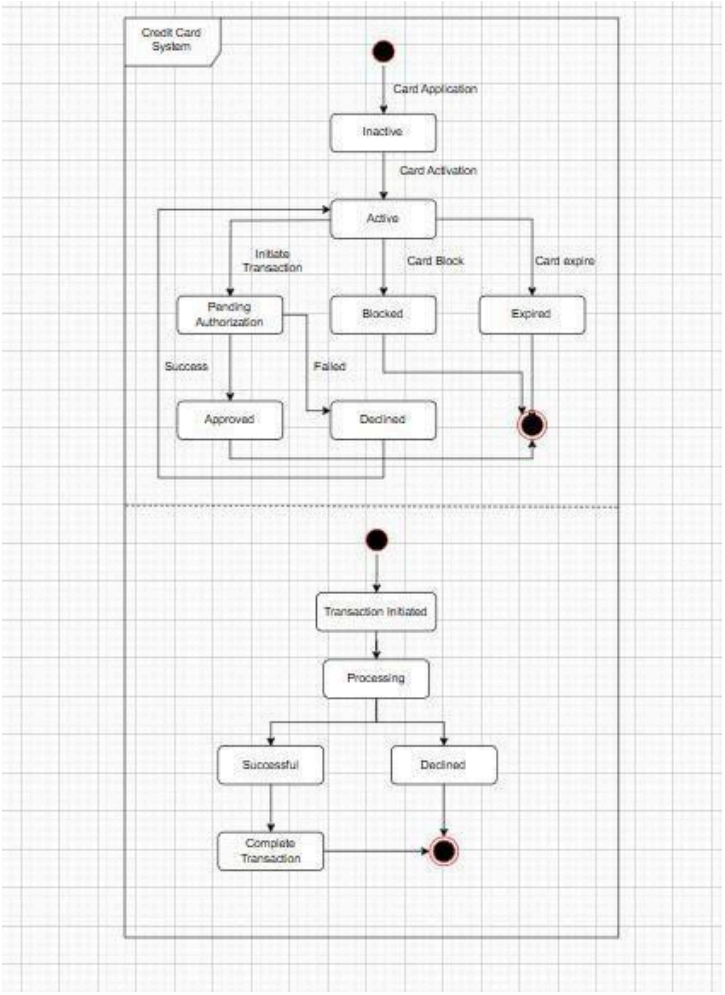
- A CardHolder can have multiple CreditCards.

- A CreditCard belongs to a CardHolder.
- A CreditTransaction is associated with a CreditCard.
- A Merchant can process multiple CreditTransactions.
- A CreditTransaction is associated with a Merchant.

Inheritance:

- MasterCard and VisaCard inherit from CreditCard.

2.4 State Diagram:



States:

- Inactive: The initial state of a newly issued credit card.
- Active: The state after the card has been activated by the cardholder.

- Blocked: The state when a card is temporarily or permanently blocked, typically due to security reasons or non-payment.
- Expired: The state when a card reaches its expiration date and becomes unusable.

Transitions:

- Card Application: Transition from no card to an inactive state when a card application is submitted and approved.
- Card Activation: Transition from inactive to active state when the cardholder activates the card.
- Card Block: Transition from active to blocked state when the card is blocked due to security concerns or non-payment.
- Card Expire: Transition from active or blocked to expired state when the card reaches its expiration date.
- Initiate Transaction: Transition from active to pending authorization state when a transaction is initiated.
- Pending Authorization: The state while a transaction is being authorized.
- Success: Transition from pending authorization to approved state if the authorization is successful.
- Failed: Transition from pending authorization to declined state if the authorization fails.
- Transaction Initiated: Transition from approved to processing state when a transaction is initiated.
- Processing: The state while a transaction is being processed.
- Successful: Transition from processing to complete state if the transaction is successful.
- Declined: Transition from processing to complete state if the transaction is declined.

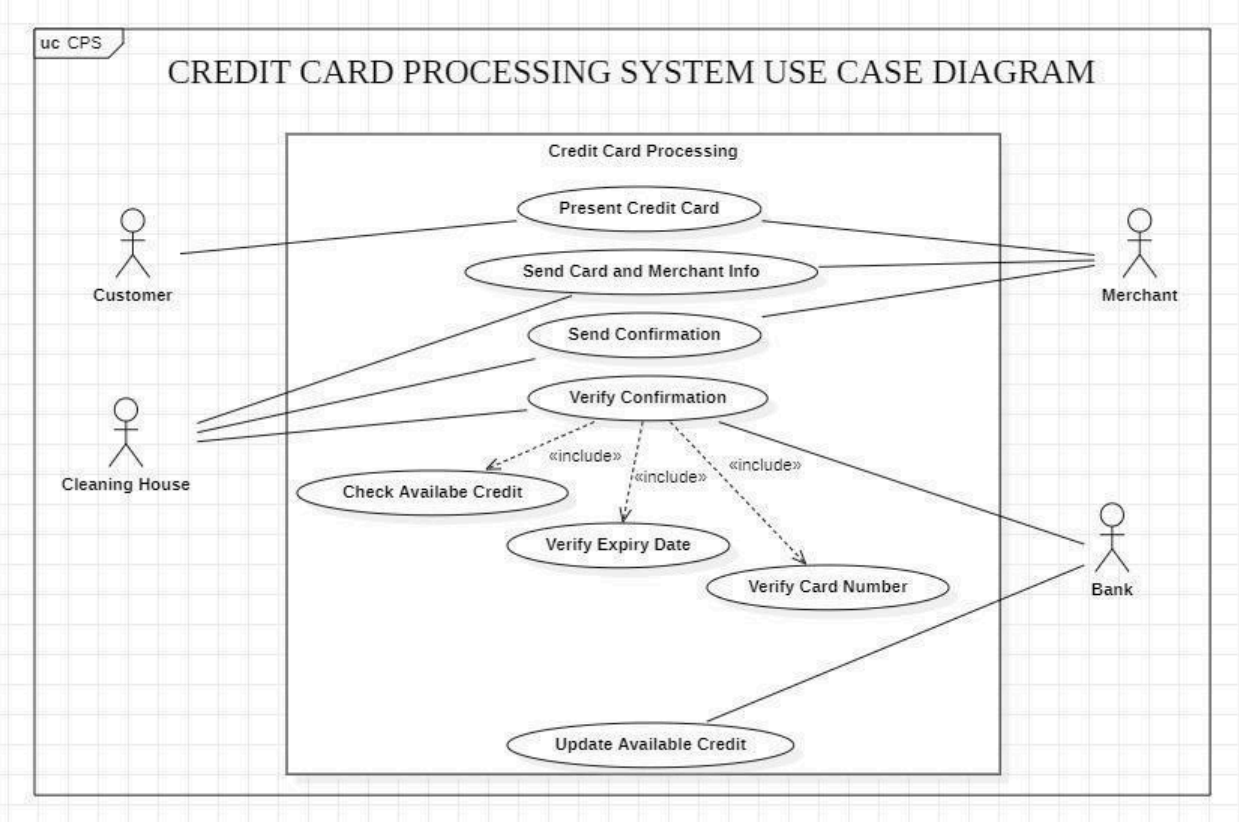
2.5 Use-Case Diagram:

Actors:

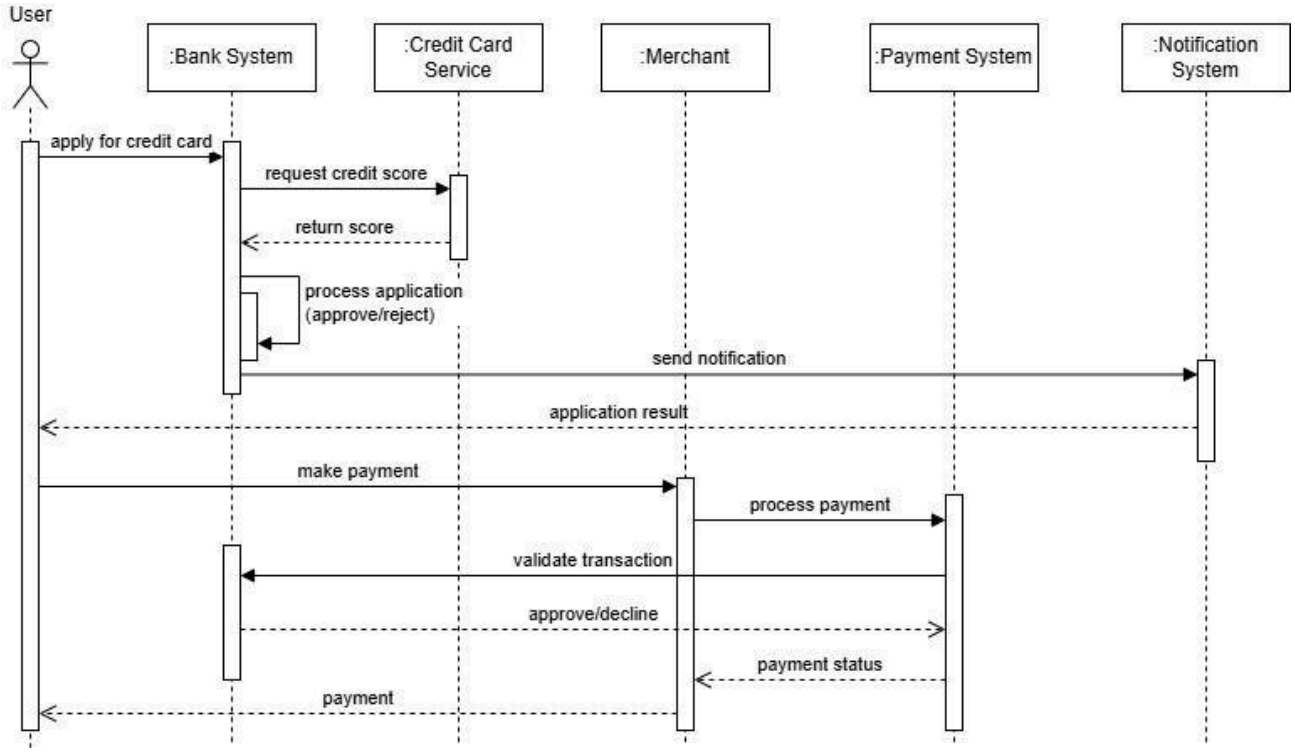
- Customer: Initiates the process by presenting their credit card for a transaction.
- Merchant: Receives the credit card information from the customer.
- Bank: The financial institution that issues the credit card and handles the transaction processing.
- Cleaning House: An entity responsible for verifying and clearing transactions.

Use Cases:

- Credit Card Processing: The main use case, encompassing the entire credit card transaction process.
- Present Credit Card: The customer presents their credit card to the merchant.
- Send Card and Merchant Info: The merchant sends the credit card information and transaction details to the bank.
- Send Confirmation: The bank sends a confirmation message back to the merchant indicating whether the transaction was approved or declined.
- Verify Confirmation: The merchant verifies the confirmation message from the bank.
- Check Available Credit: The bank checks the customer's credit limit to ensure sufficient funds are available for the transaction.
- Verify Expiry Date: The bank verifies the validity of the credit card by checking its expiry date.
- Verify Card Number: The bank verifies the validity of the credit card number.
- Update Available Credit: If the transaction is approved, the bank updates the customer's available credit limit.



2.6 Sequence Diagram:



Sequence:

- User applies for a credit card.
- Bank System requests the credit score from the Credit Card Service.
- Credit Card Service returns the credit score to the Bank System.
- Bank System processes the application and decides whether to approve or reject it.
- Bank System sends an application result notification to the User.
- User makes a payment using the credit card.
- Payment System processes the payment and validates the transaction.
- Payment System sends the payment status to the Merchant.
- Merchant receives the payment and completes the transaction.

2.7 Activity Diagram:

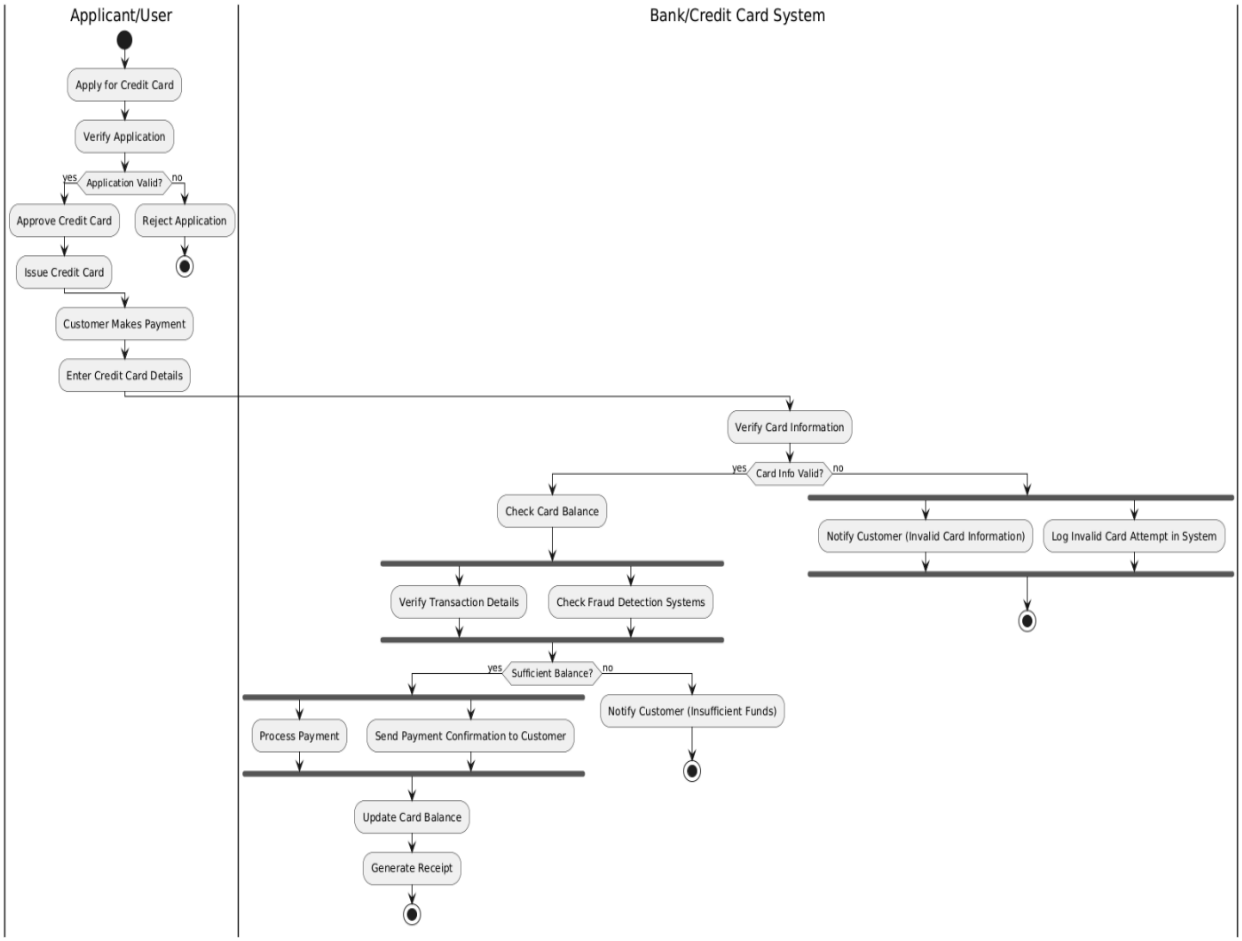
Applicant/User:

- Applies for a Credit Card.
- The system verifies the application.

- If valid, the system approves the credit card and issues it.
- If invalid, the application is rejected.

Customer Makes Payment:

- Enters Credit Card Details.
- The system verifies the card information.
- If valid, it checks the card balance.
- If insufficient funds, the customer is notified.
- If sufficient funds, the system verifies transaction details and checks fraud detection systems.
- If no issues, the system processes the payment, sends a confirmation to the customer, updates the card balance, and generates a receipt.
- If invalid card information, the customer is notified and the system logs the invalid attempt.



3. Library Management System

3.1 Problem Statement:

This project aims to develop a comprehensive and efficient library management system. The system will streamline library operations and provide a seamless experience for both users and librarians. It will encompass key functionalities such as book cataloging, user management, and transaction processing. The system should cater to the needs of both users and librarians. Users will be able to easily search for books, place orders, make online payments, manage their memberships, and access their transaction history. Librarians will have tools to manage book inventory, process user requests, and assist users with their queries. The system must be robust, scalable, and secure. It should be able to handle a growing number of users and books while maintaining optimal performance. Data security measures will be implemented to protect user information and prevent unauthorized access. The system should also incorporate accessibility features to ensure usability for all users, including those with disabilities. The system will be developed using appropriate technologies, including a suitable programming language, database management system, and web framework.

3.2 SRS-Software Requirements Specification:

4/10/24

Library Management System

1. Introduction:

- Purpose:
This document provides the complete requirements for the Library Management System, outlining its functionalities, performance and design constraints. The purpose of the system is to facilitate the effective management of library operations, including book cataloging, borrowing, returning and managing member records.

- Scope: The scope of this document is to define the objectives, requirements and functionalities of a LBSYS. It will serve as a solution for libraries to manage their inventory of books and track user's borrowing and return activities.

- Overview: It is a software application designed to handle the operation of a library, such as tracking book inventory, managing member registration, issuing books and sending reminders for overdue books.

2. General Description: It designed to manage library operations efficiently by keeping track of book inventories, user profiles, borrowed books and fines. The system will allow users to search for books, borrow and return books.

3. Functional Requirements:

→ Book Management -
- Add, update or remove books from the system
- Track the availability status of the books.
- Maintain book details including title, author, ISBN.

→ User Management -
- Register and manage library members.
- Maintain member profiles with contact information.

→ Book Borrowing and returning -
- Allow members to borrow books based on availability.
- Enable librarians to issue and receive books.

→ Reservation System –
- Allow members to reserve books that are currently unavailable.
- Notify users when a reserved book becomes available.

→ Fine Management –
- Automatically calculates fine for overdue books.
- Calculate fine if any user misplaces any book.

→ Search and catalog –
- provide search functionalities for books by title, author or genre.

4. Interface Requirements –

→ A web based system accessible through any modern browser

→ Separate interfaces for librarians and members.

→ Easy-to use navigation for searching books.

5. Performance Requirements:

- Response Time - The system must respond to user requests within 2 seconds.
- Data handling - Support for upto 50,000 book entries and 10,000 active members.
- Concurrency - Should handle 500 concurrent users without performance degradation.
- Error rate - Should be less than 0.05% of all operations.
- Uptime - Maintain an uptime of 99.9% to ensure system availability.

6. Design constraints:

- It must be compatible with popular operating systems like Windows, MacOS, Linux.
- MERN stack can be used.
- System must support cloud deployment for scalability

7. Non-functional attributes:

- Security
- Reliability
- Scalability

8. Schedule

Phase 1: (Requirement gathering and Analysis) 1 month

Phase 2: (Design and Development) 4 months

Phase 3: (Testing) 1 month

Phase 4: (Deployment) 15 days.

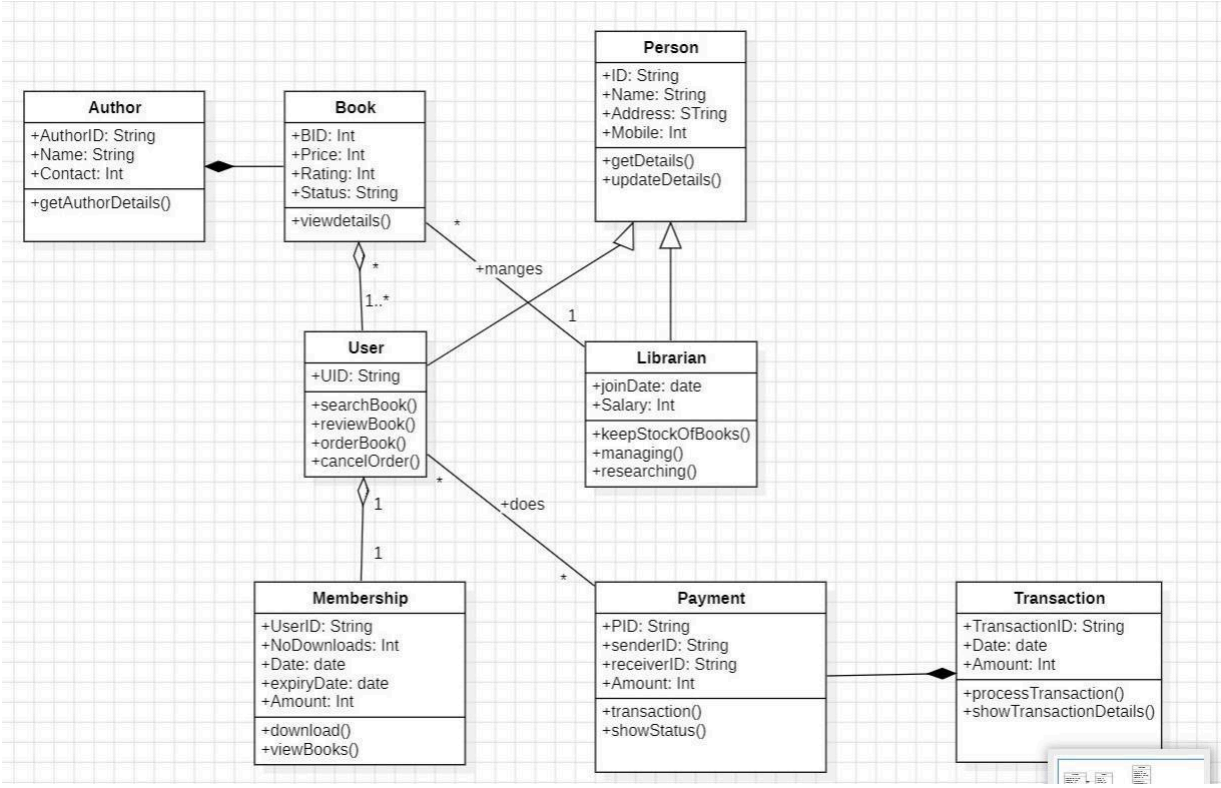
9. Budget

Estimated budget: ₹2,00,000

Planning - ₹ 20,000
Development - ₹1,00,000
Testing - ₹ 30,000
Deployment - ₹ 50,000

7/10/2024

3.3 Class Diagram:



Entities:

- Person: A base class with attributes like name, address, and mobile. It is inherited by User and Librarian.
- User: Represents users of the system, with attributes like UID, joinDate, and methods for searching, ordering, and canceling books.
- Librarian: Represents library staff, with attributes like salary, joinDate, and methods for managing books, keeping stock, and researching.
- Book: Represents books with attributes like BID, price, rating, and status.
- Author: Represents authors of books with attributes like authorID, name, and contact.
- Membership: Represents user memberships with attributes like UID, date, expiryDate, and methods for checking available books and renewing memberships.
- Payment: Represents payment transactions with attributes like PID, sender, receiverID, amount, and methods for processing and showing transaction details.
- Transaction: Represents book transactions with attributes like transactionID, date, amount, and methods for processing and showing transaction details.

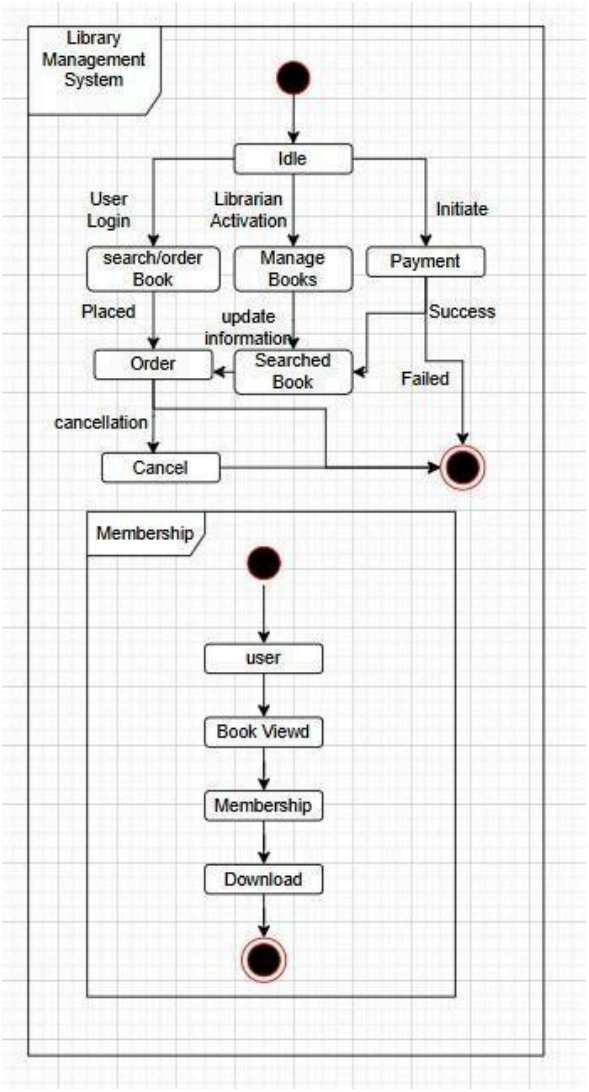
Relationships:

- Inheritance: User and Librarian inherit from Person.

Association:

- A Book can have multiple Authors.
- A User can order multiple Books.
- A Librarian can manage multiple Books.
- A User can have a Membership.
- A User can make Payments.
- A Transaction can be associated with a User.

3.4 State Diagram:



States:

- Idle: The initial state where the user is not actively interacting with the system.
- User Login: The state where the user logs into the system.
- Librarian Activation: The state where a librarian activates their account.
- Manage Books: The state where a librarian manages book information (adding, editing, deleting).
- Search/Order Book: The state where a user searches for a book and places an order.
- Placed: The state where the user's order is placed.
- Order Cancel: The state where the user cancels an order.
- Payment: The state where the user makes a payment for a book.
- Success: The state where the payment is successful.
- Failed: The state where the payment fails.
- Membership: The state where the user manages their membership.
- User: The state where the user is actively using the system.
- Book Viewed: The state where the user views a book.
- Membership: The state where the user manages their membership.
- Download: The state where the user downloads a book.

Transitions:

- The diagram shows transitions between these states based on user actions and system responses. For example, a user can transition from the "Idle" state to the "User Login" state by logging in.

3.5 Use-Case Diagram:

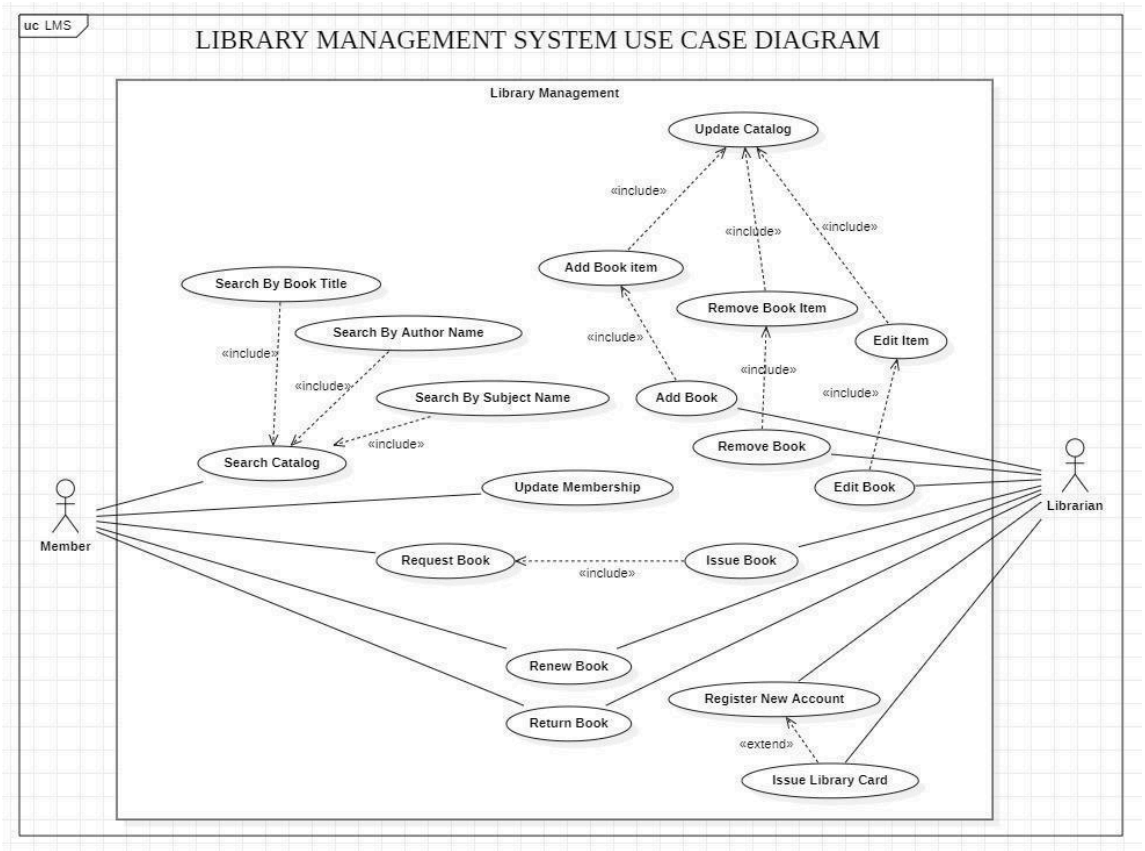
Use Cases:

- Handle Cataloging: This is a core use case that includes several sub-use cases related to book management.
- Add Book: Allows adding new books to the library catalog.
- Remove Book: Enables removing books from the catalog.
- Edit Book: Allows updating existing book records.
- Search Catalog: Allows members to search for books based on various criteria:

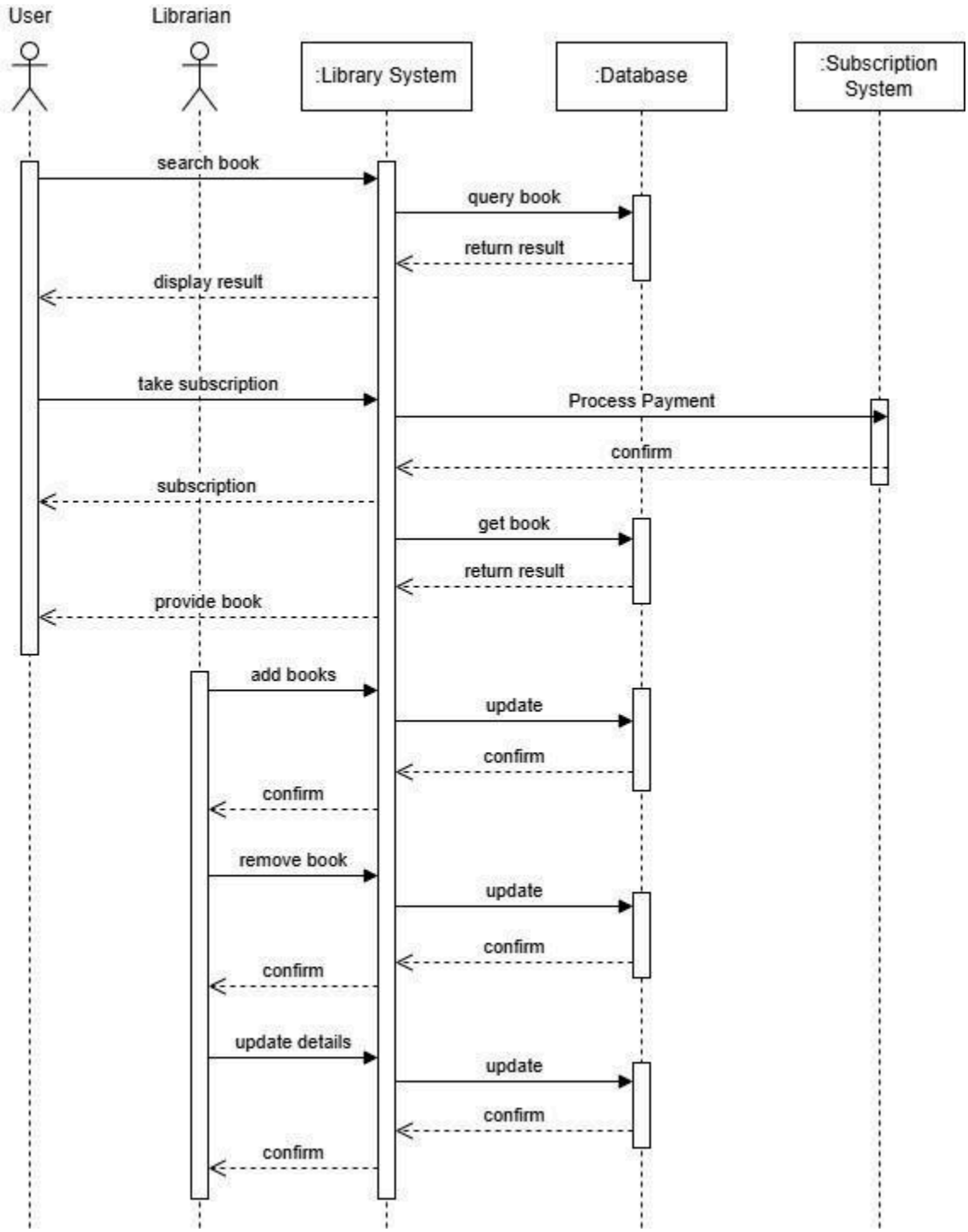
- Search By Book Title
- Search By Author Name
- Search By Subject Name
- Request Book: Allows members to request books that are currently unavailable.
- Issue Book: Allows librarians to issue books to members.
- Renew Book: Allows members to renew borrowed books.
- Return Book: Allows members to return borrowed books.
- Update Membership: Allows members to update their membership information.
- Register New Account: Allows new members to register for a library account.
- Issue Library Card: Allows librarians to issue library cards to new members.

Relationships:

- Include: Several sub-use cases are included within the "Handle Cataloging" use case.
- Extend: The "Issue Library Card" use case extends the "Register New Account" use case, indicating that issuing a library card is an optional step after registering a new account.



3.6 Sequence Diagram:



sequence:

- User searches for a book: The user initiates the process by searching for a book.

- Library System queries the database: The Library System sends a query to the Database to retrieve information about the searched book.
- Database returns results: The Database responds to the Library System with the search results.
- Library System displays results: The Library System displays the search results to the User.
- User takes subscription: The User decides to subscribe or borrow the book.
- Subscription System processes payment: If required, the Subscription System handles the payment process.
- Library System gets book: The Library System retrieves the book from the database.
- Library System provides book: The Library System provides the book to the User.
- Librarian adds books: The Librarian adds new books to the system.
- Library System updates database: The Library System updates the database with the new book information.
- Librarian removes books: The Librarian removes books from the system.
- Library System updates database: The Library System updates the database to reflect the removal of books.
- Librarian updates database: The Librarian updates other information in the database.

3.7 Activity Diagram:

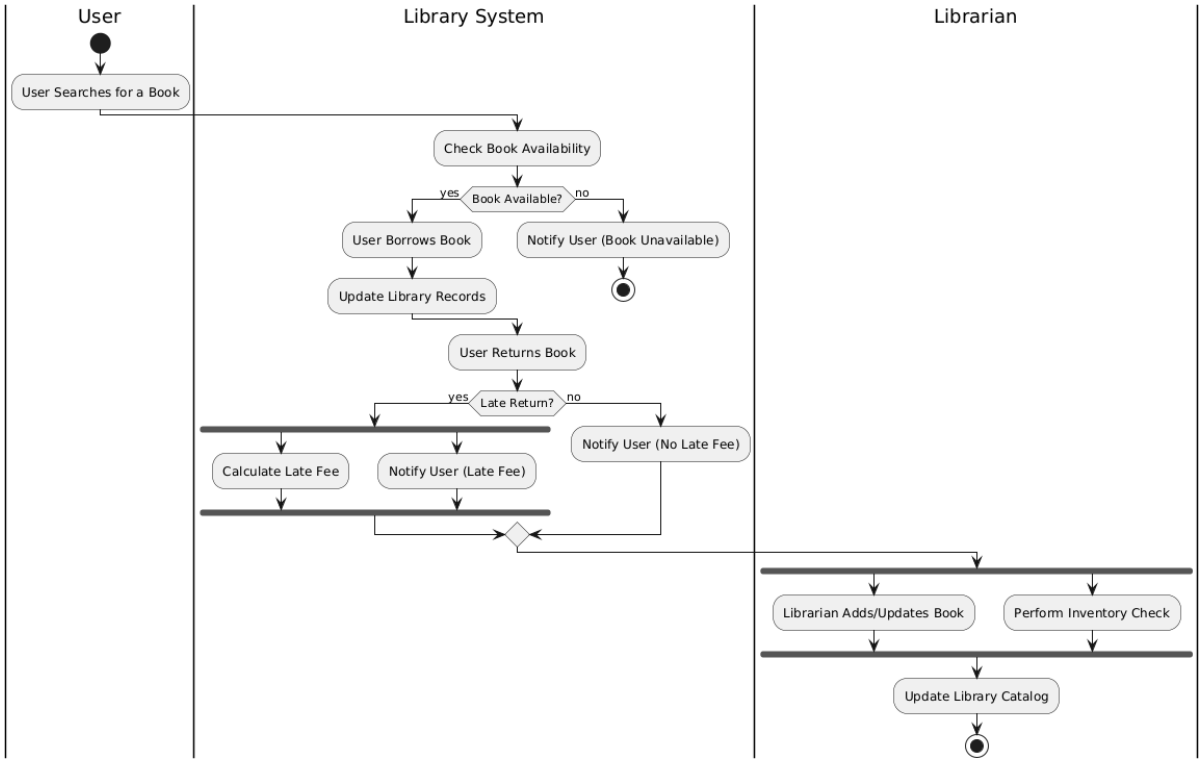
Activities:

- User Actions:
- User logs in to the system.
- User searches for a book.
- User selects a book.
- User checks out the book.
- User returns the book.
- User views their account information.
- User updates their account information.
- Librarian Actions:
- Librarian logs in to the system.
- Librarian manages books (adds, removes, updates).
- Librarian manages user accounts.

- Librarian processes subscriptions.
- Librarian updates library inventory.

Workflow:

- The diagram shows the sequence of these activities, indicating how user and librarian actions interact.
- For example, a user first logs in, then searches for a book, selects it, and finally checks it out.
- Librarians manage the book inventory, process user requests, and update user information.



4. Stock Maintenance System

4.1 Problem Statement:

Stock maintenance systems are critical for businesses that deal with inventory, encompassing a wide range of processes and technologies aimed at optimizing the flow of goods. These systems enable businesses to accurately track stock levels across all locations, monitor stock movements, and generate insightful reports on inventory trends and usage. By effectively managing inventory, businesses can minimize the risks associated with stockouts and overstocking, leading to significant cost savings. Furthermore, these systems facilitate efficient order fulfillment, ensuring timely delivery to customers and enhancing overall customer satisfaction. By leveraging data- driven insights and employing forecasting techniques, businesses can optimize inventory levels, predict future demand, and proactively address potential challenges. Key components of stock maintenance systems often include Warehouse Management Systems (WMS), Enterprise Resource Planning (ERP) systems, and technologies like RFID and barcoding for real-time tracking. By implementing robust stock maintenance practices, businesses can gain a competitive edge by streamlining operations, reducing costs, and improving overall efficiency.

4.2 SRS-Software Requirements Specification:

14/10/24

METRO
Page
Date / /

Stock Maintenance System

1. Introduction

→ Purpose: The document outlines the software requirements for the development of a stock Management System. The purpose of this system is to help businesses efficiently manage their inventory, including stock levels, order management, supplier coordination and reporting.

→ Scope - It will facilitate the following functions:
→ Stock entry and update
→ Track inventory levels
→ Generate low stock alerts
→ Monitor stock movement (Inward/outward)
→ Provide detailed reporting for stock analysis

→ Overview - It will be designed to manage inventory for businesses, providing features such as stock entry, update and notifications for low stock levels. It will operate in real-time, accessible through web and mobile interfaces and is intended to streamline stock management processes for small to medium-sized enterprises.

2. General Description: The system is aimed at improving inventory accuracy and efficiency by automating stock-related processes. The main functions include real-time tracking of stock levels, automated notifications for low stock, and generating comprehensive reports.

3. Functional requirements:

- Stock entry and update - The system shall allow users to enter and update stock details, including product name, quantity.
- The system shall track stock in and stock out.
- Inventory tracking - The system shall display real-time stock levels of all products
- The system shall allow users to filter products by name, category and stock status.
- Low stock Alerts - The system shall send email/SMS notifications when stock falls below a predefined threshold.

- Stock Movement Reports - The system shall generate reports on stock movement within a specified time range.

4. Interface Requirements - User Interface must include admin dashboard to provide a complete overview of stock status and employee dashboard to update stock and view current inventory levels. Software Interface includes database integration and API Integration.

5. Performance Requirements -
   - The system shall support up to 10,000 unique stock items with response times under 2 seconds for all queries.
   - The system shall process stock updates within 1 second of input.
   - System shall have an uptime of 99.9% to ensure continuous availability for users.
   - It should operate efficiently on standard hardware with atleast 8GB of RAM and 4 CPUs.

6. Design Constraints

- The system shall be developed using the MERN stack.
- The system must be compatible with mobile devices and desktop environments.
- Due to security requirements, the system must use encrypted comm" (SSL) and comply with data privacy regulations.

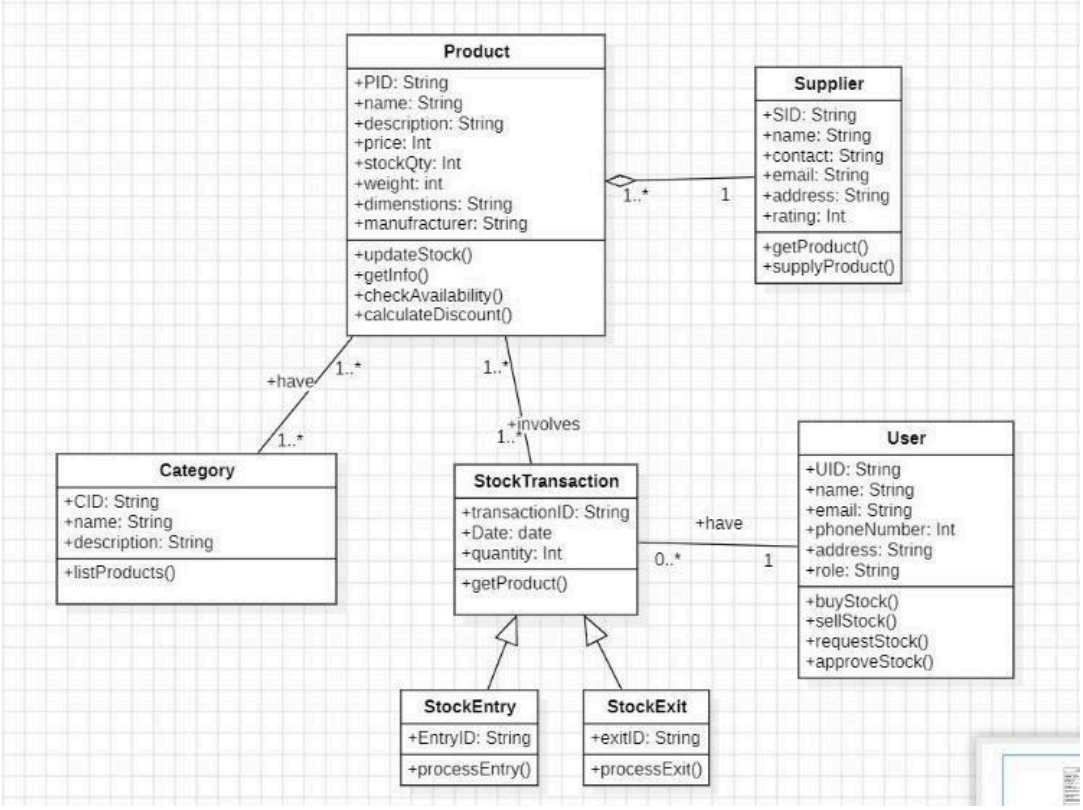
7. Non-functional Attributes

- Security
- Portability
- Reliability
- Scalability

8. Schedule: Phase 1: Planning: 2 weeks
Phase 2: Design and prototyping:
Phase 3: Development: 6 weeks
Phase 4: Testing and Bug Fixing: 8
Phase 5: Deployment: 4 weeks
Phase 6: Total Duration: 6 months

9. Budget:
- Development: $30,000
- Testing: $10,000
- Cloud Hosting: $5,000
- Total Budget: $45,000

4.3 Class Diagram:



Entities:

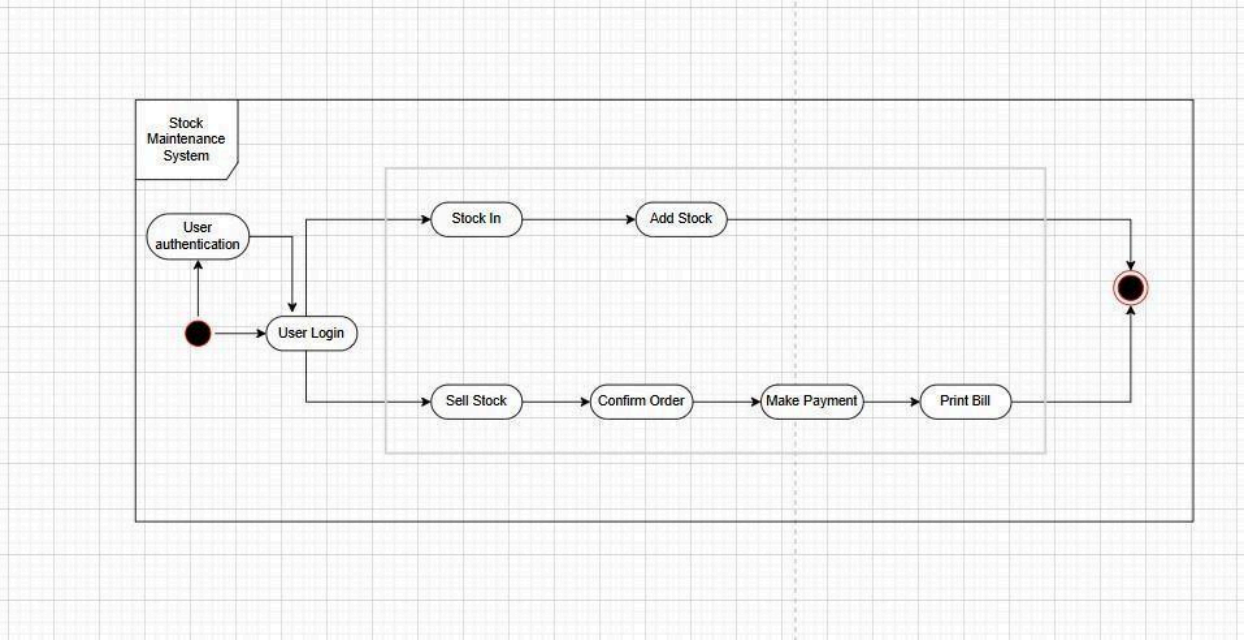
- Product: Represents individual products with attributes like ID, name, price, stock quantity, and dimensions. It also includes methods for checking availability and calculating discounts.
- Supplier: Represents suppliers of products with attributes like ID, name, contact information, and rating.
- Category: Represents product categories with attributes like ID, name, and description.
- User: Represents users of the system, potentially customers or employees, with attributes like ID, name, contact information, and roles.
- Stock Transaction: Represents changes in product stock levels, including stock entries (additions) and stock exits (removals).
- StockEntry: Represents stock additions with attributes like entry ID and processing methods.
- StockExit: Represents stock removals with attributes like exit ID and processing methods.

Relationships:

- Product-Category: A product belongs to a category.

- Product-Supplier: A product is supplied by a supplier.
- Product-StockTransaction: A product is associated with multiple stock transactions.
- StockTransaction-StockEntry/StockExit: A stock transaction can be either a stock entry or a stock exit.
- User-StockTransaction: Users can initiate stock transactions (e.g., placing orders).

4.4 State Diagram:



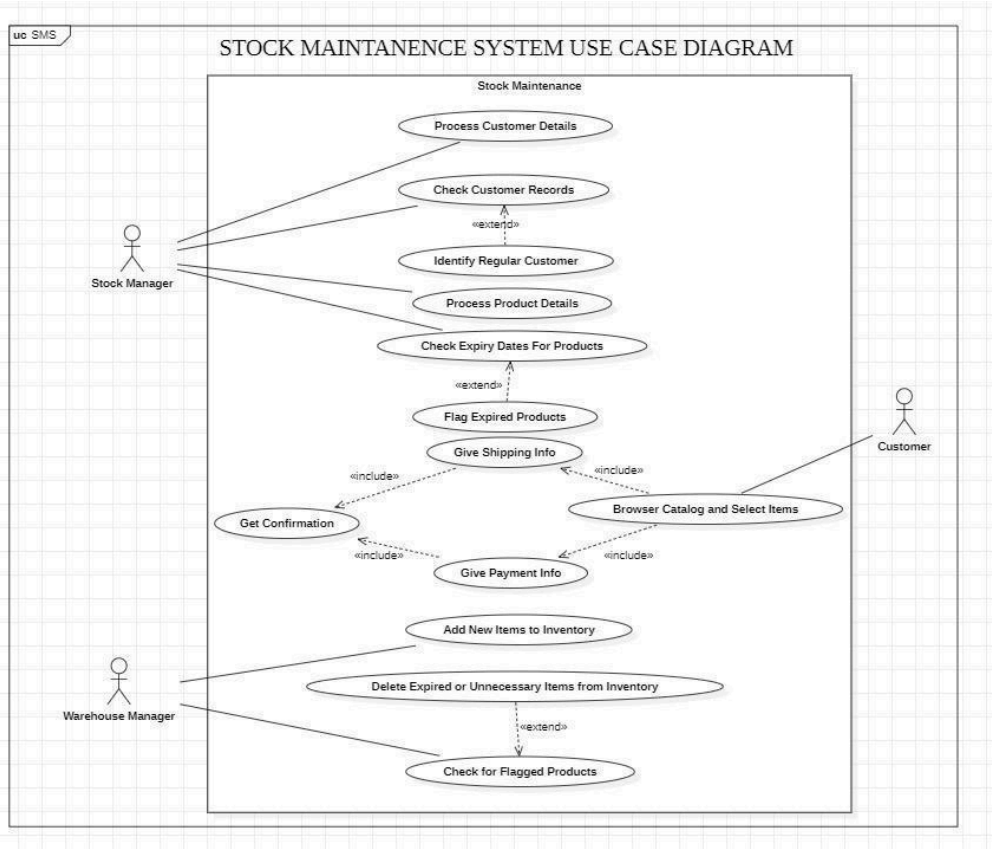
States:

- User Authentication: The initial state where the user needs to authenticate themselves to access the system.
- Stock In: The state where the user is managing incoming stock (e.g., receiving new inventory).
- Add Stock: The state where the user is adding new stock items to the system.
- Sell Stock: The state where the user is processing a sale or order.
- Confirm Order: The state where the user confirms the details of a sale or order.
- Make Payment: The state where the user handles the payment for a sale.
- Print Bill: The final state where the system generates and prints an invoice or receipt.

Transitions:

- The diagram shows the transitions between these states based on user actions and system responses. For example, a user can transition from "User Authentication" to "Stock In" if they have permission to manage incoming stock.

4.5 Use-Case Diagram:



Use Cases:

- Stock Maintenance: This is a core use case that includes several sub-use cases related to stock management.
- Process Customer Details: This likely involves gathering and verifying customer information for orders or inquiries.
- Check Customer Records: This could involve checking customer history, creditworthiness, or other relevant information.
- Identify Regular Customer: This use case might be used to identify and provide special offers or discounts to regular customers.
- Process Product Details: This likely involves handling product information, such as checking stock availability, verifying pricing, and managing product descriptions.

- Check Expiry Dates For Products: This is crucial for perishables or products with limited shelf life.
- Flag Expired Products: This involves identifying and marking products that have exceeded their expiry dates.
- Give Shipping Info: This use case involves providing shipping and delivery information to customers.
- Get Confirmation: This likely involves obtaining confirmation from the customer about their order or any changes.
- Give Payment Info: This involves processing payment information from the customer.
- Add New Items to Inventory: This allows for adding new products to the stock inventory.
- Delete Expired or Unnecessary Items from Inventory: This enables the removal of expired or obsolete products from the inventory.
- Check for Flagged Products: This allows for identifying and handling products that have been flagged for any reason (e.g., quality issues).

Relationships:

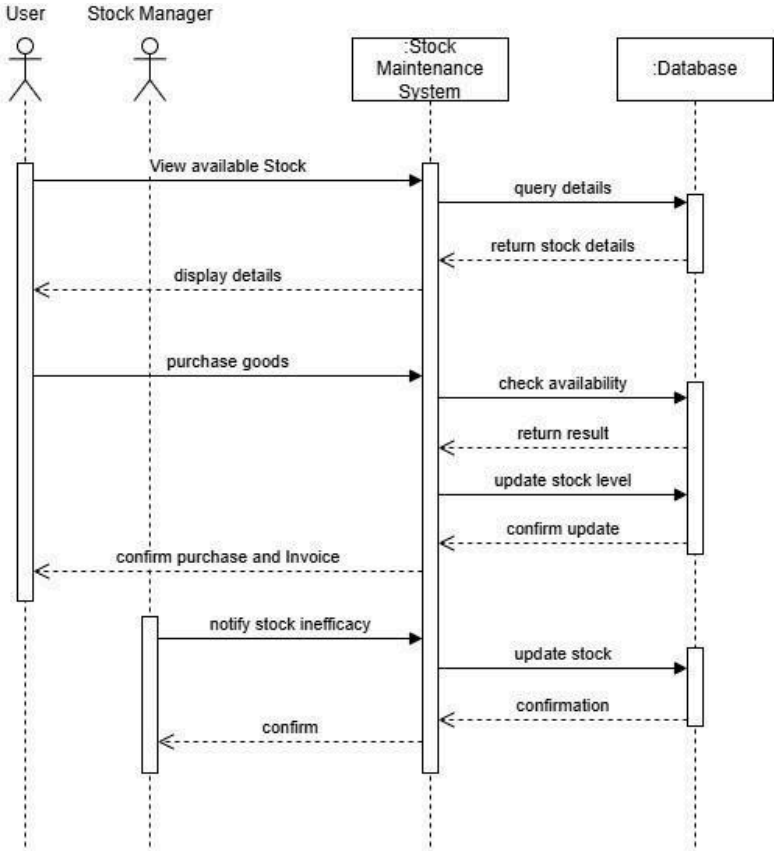
- Include: Several sub-use cases are included within the "Stock Maintenance" use case.
- Extend: The "Identify Regular Customer" use case extends the "Process Customer Details" use case, indicating that identifying regular customers is an optional step within customer processing.

4.6 Sequence Diagram:

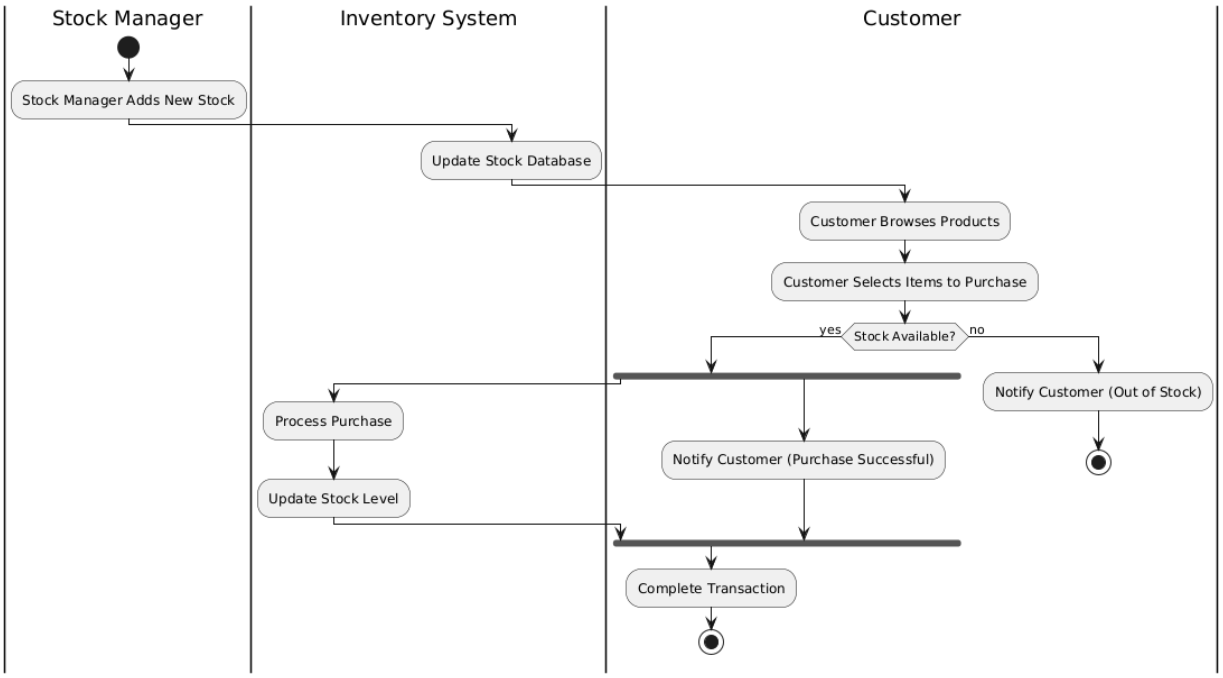
sequence:

- User views available stock: The user initiates the process by viewing the available stock.
- Stock Maintenance System queries details: The Stock Maintenance System sends a query to the Database to retrieve details about the available stock.
- Database returns stock details: The Database responds to the Stock Maintenance System with the requested stock details.
- Stock Maintenance System displays details: The Stock Maintenance System displays the retrieved stock details to the User.
- User purchases goods: The User decides to purchase certain goods.

- Stock Maintenance System checks availability: The Stock Maintenance System checks the availability of the requested goods in the database.
- Database returns result: The Database responds with the availability status of the requested goods.
- Stock Maintenance System updates stock level: If the goods are available, the Stock Maintenance System updates the stock levels in the database.
- Database confirms update: The Database confirms the successful update of the stock levels.
- Stock Maintenance System confirms purchase and invoice: The Stock Maintenance System confirms the purchase and generates an invoice for the User.
- Stock Maintenance System notifies stock inefficiency: If the stock levels fall below a certain threshold, the Stock Maintenance System notifies the relevant personnel about the stock inefficiency.
- Stock Maintenance System updates stock: The Stock Maintenance System updates the stock levels accordingly.
- Database confirms confirmation: The Database confirms the successful update of the stock levels.



4.7 Activity Diagram:



Activities:

- Stock Manager Adds New Stock: The Stock Manager initiates the process by adding new stock items to the system.
- Update Stock Database: The Inventory System updates its database to reflect the addition of new stock.
- Customer Browses Products: The Customer explores the available products within the system.
- Customer Selects Items to Purchase: The Customer chooses the products they wish to purchase.
- Check Stock Availability: The Inventory System verifies if the requested items are available in stock.
- Process Purchase: If stock is available, the Inventory System proceeds to process the purchase.
- Update Stock Level: The Inventory System updates the stock levels to reflect the items sold.
- Notify Customer (Purchase Successful): The Inventory System notifies the Customer that their purchase was successful.
- Notify Customer (Out of Stock): If any items are out of stock, the Inventory System notifies the Customer accordingly.
- Complete Transaction: The entire purchase process is completed.

5. Passport Automations System

5.1 Problem Statement:

The current manual passport application and issuance process presents significant challenges. The reliance on paper-based forms, manual data entry, and physical document handling leads to a cumbersome and time-consuming process for both applicants and government agencies. This manual approach is prone to errors, delays, and inconsistencies in data entry, increasing the risk of inaccuracies and potential fraud. Furthermore, the manual system often involves multiple touchpoints and physical visits to government offices, causing inconvenience and frustration for applicants. The long processing times associated with manual applications can have a negative impact on travel plans and business activities. To address these issues, an automated passport system is urgently needed. This system should leverage technology to streamline the entire application process, from online submission and data capture to secure document verification and passport production.

5.2 SRS-Software Requirements Specification:

14/10/24

METRO
Page:
Date: / /

Passport Automation System

(1) Introduction

→ Purpose: The purpose of this document is to describe the software requirements for the Passport Automation System (PAS). The system aims to automate and streamline the passport application process, reducing manual effort, minimizing errors, and improving user experience.

→ Scope – The system will enable applicants to apply for a passport online, track the status of their application and receive updates.

→ Overview – This document provides a detailed description of the system's functional and non-functional requirements, including use cases, design constraints and system interaction diagrams.

(2) General Description

The system will streamline the passport application and issuance process by allowing applicants to fill out forms, upload documents, make payments and track the

status of their application online.

(3) Functional requirements

(1) User registration and authentication

→ The system shall allow applicants to register using their email or phone no.

→ The system shall send an OTP for authentication during registration or login.

(2) Passport Application Submission

→ The system shall allow users to fill out a passport application form.

→ The system shall allow users to upload necessary documents in the specified formats.

(3) Fee payment

→ The system shall integrate with a payment gateway to allow users to pay the application fee.

→ It must generate a receipt after successful payment.

(4) Appointment Scheduling → The system shall provide available appointment slots for document

verification

(5) Application Status Tracking - It shall display real-time application status updates to the user.

(6) Administrative functions - It shall allow administrators to verify documents and approve or reject applications.

(4) Interface Requirements - It includes user interfaces and software interfaces.

(5) Performance Requirements -

- Shall handle upto 100,000 users simultaneously.
- Shall process passport applications within an average of 2 seconds per submission.
- Shall store upto 1 million applications with a backup plan for additional storage as needed.
- Shall have an error rate of less than 0.1% in terms of transaction failures.

(6) Non-functional attributes

- Security
- Portability

- Reliability
- Scalability

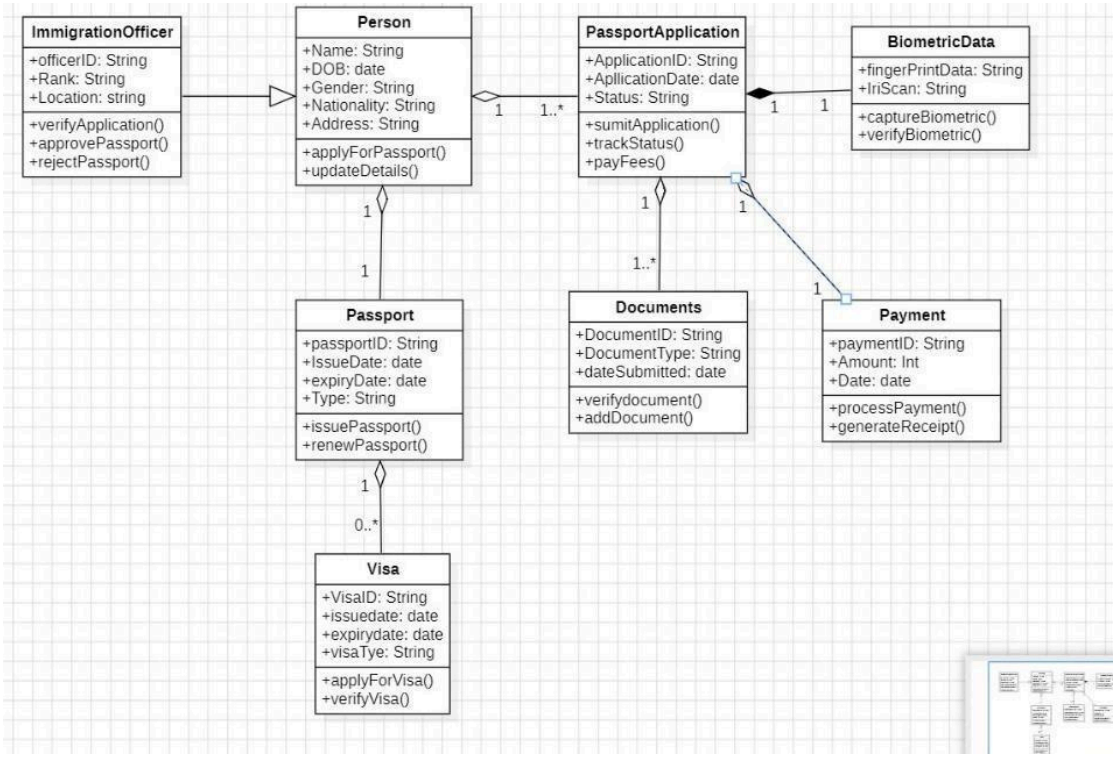
(7) Schedule

- Phase 1: Requirements gathering (2 months)
- Phase 2: Design (2 months)
- Phase 3: Development (4 months)
- Phase 4: Testing and Deployment (1 month)

(8) Budget

- Development Cost: $2,50,000
- Testing and maintenance cost: $50,000
- Total cost: $300,000

5.3 Class Diagram:



Entities:

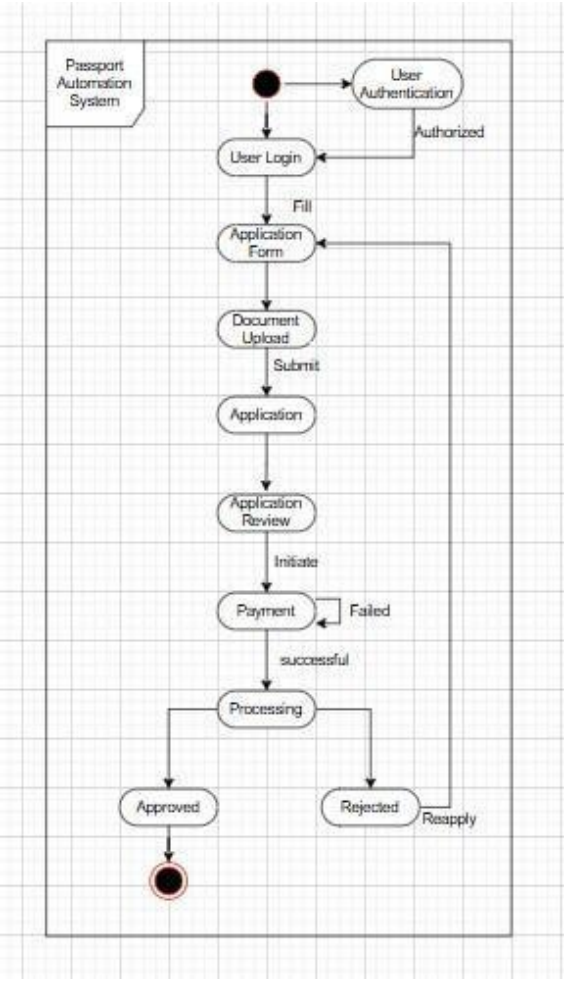
- Person: A base class representing individuals with attributes like name, date of birth, gender, nationality, and address.
- ImmigrationOfficer: Represents the officers responsible for processing passport applications, with attributes like officerID, rank, and location.
- PassportApplication: Represents an individual's application for a passport, with attributes like applicationID, application date, and status.
- Passport: Represents an issued passport with attributes like passportID, issue date, expiry date, and type.
- Documents: Represents the documents submitted along with the passport application, with attributes like documentID, document type, and date submitted.
- BiometricData: Represents biometric information collected as part of the application process, such as fingerprints and iris scans.
- Payment: Represents the payment made for the passport application, with attributes like paymentID, date, and amount.

- Visa: Represents a visa issued to an individual, with attributes like visaID, expiry date, and visa type.

Relationships:

- Inheritance: ImmigrationOfficer inherits from Person.
- Association:
  1. A Person can apply for multiple PassportApplications.
  2. A PassportApplication is associated with one or more Documents.
  3. A PassportApplication is associated with BiometricData.
  4. A PassportApplication is associated with a Payment.
  5. A Passport is issued to a Person.
  6. A Visa is issued to a Person.

5.4 State Diagram:



States:

- User Authentication: The initial state where the user needs to authenticate themselves to access the system.
- User Login: The state where the user successfully logs into the system.
- Application: The state where the user fills out and submits the passport application.
- Application Review: The state where the application is under review by the authorities.
- Payment Initiated: The state where the user initiates the payment for the passport application.
- Payment Processing: The state where the payment is being processed.
- Payment Successful: The state where the payment is successfully processed.
- Payment Failed: The state where the payment process fails.
- Approved: The state where the passport application is approved.
- Rejected: The state where the passport application is rejected.
- Resubmit: The state where the user is required to resubmit the application after rejection.

Transitions:

- The diagram shows the transitions between these states based on user actions and system responses. For example, a user can transition from "User Login" to "Application" after successfully logging in.

5.5 Use-Case Diagram:

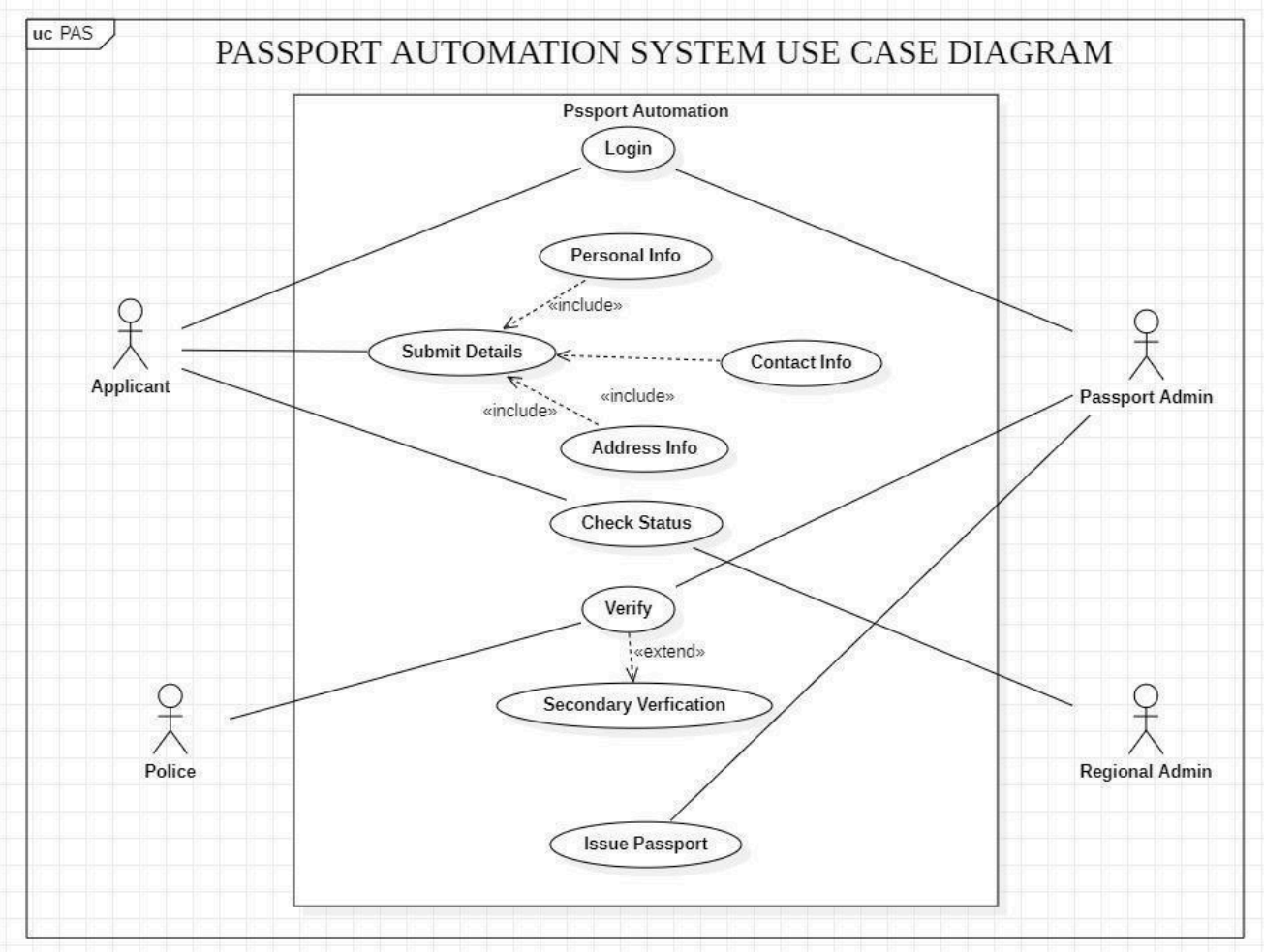
Use Cases:

- Passport Automation: This is the main use case, encompassing the entire passport application process.
- Login: The initial step where users (both applicants and administrators) log into the system.
- Personal Info: Applicants provide their personal information, such as name, date of birth, and contact details.
- Submit Details: Applicants submit their application forms and supporting documents.
- Contact Info: Applicants provide contact information for further communication.
- Address Info: Applicants provide their current address and other relevant address details.
- Check Status: Applicants can check the status of their passport applications.

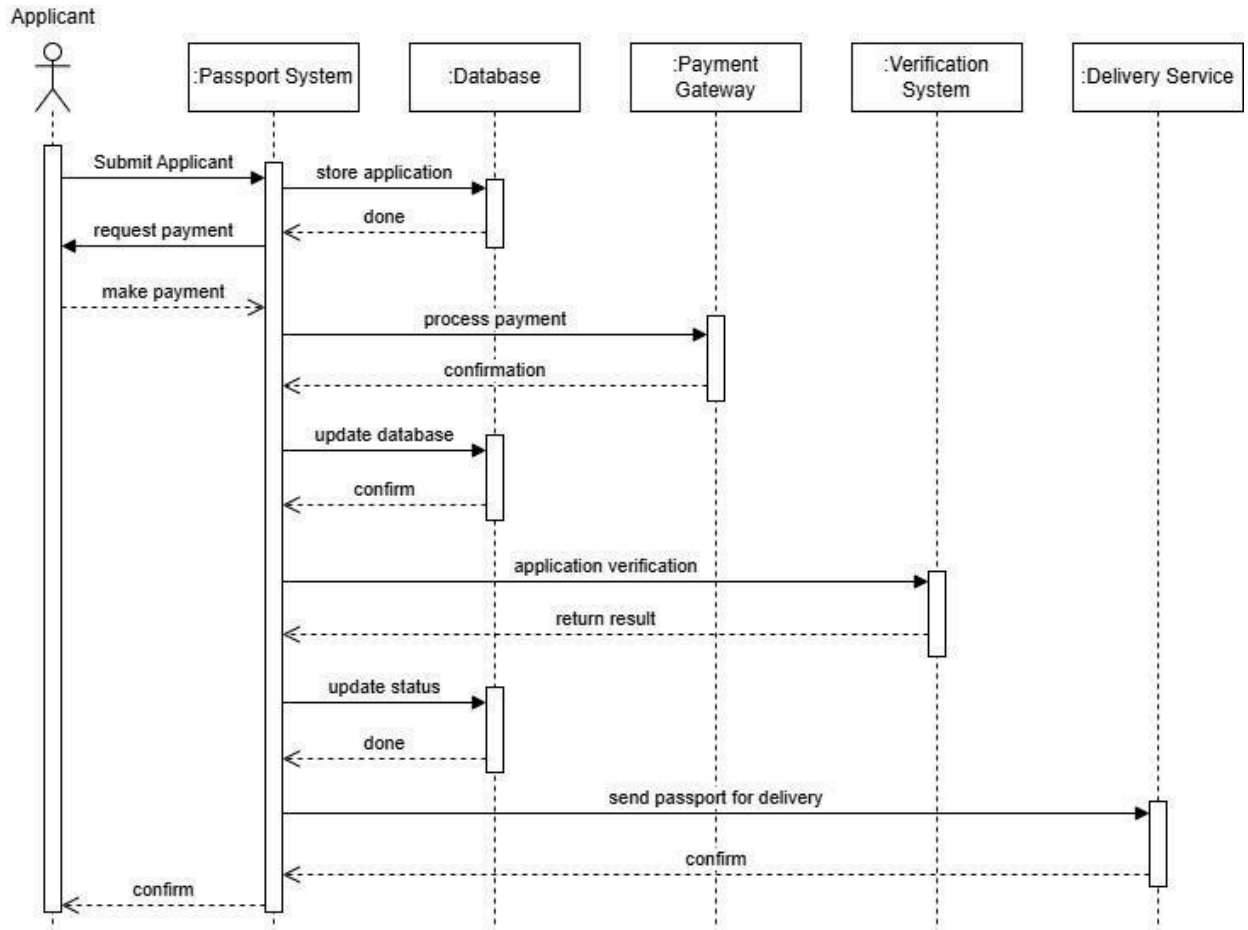
- Verify: This use case involves verification of applicant information, possibly by Police or Regional Admin.
- Secondary Verification: This use case represents additional verification steps, which might involve background checks or other procedures.
- Issue Passport: The final step where the Passport Admin issues the passport to the applicant.

Relationships:

- Include: Several sub-use cases are included within the "Passport Automation" use case, indicating that these are necessary steps within the overall process.
- Extend: The "Secondary Verification" use case extends the "Verify" use case, suggesting that secondary verification is an optional step that might be required in certain situations.



5.6 Sequence Diagram:



Sequence:

- Applicant Requests Payment: The applicant initiates the process by requesting payment for the passport application.
- Passport System Stores Application: The Passport System receives the payment request and stores the application details.
- Applicant Makes Payment: The applicant proceeds to make the payment through the Payment Gateway.
- Payment Gateway Processes Payment: The Payment Gateway processes the payment and confirms the transaction.
- Payment Gateway Confirms Payment: The Payment Gateway sends a confirmation of the successful payment to the Passport System.
- Passport System Updates Database: The Passport System updates its database to reflect the successful payment.

- Passport System Requests Verification: The Passport System sends the application to the Verification System for further processing.
- Verification System Verifies Application: The Verification System performs necessary checks and verifications on the application.
- Verification System Returns Result: The Verification System returns the result of the verification process to the Passport System.
- Passport System Updates Status: Based on the verification result, the Passport System updates the application status.
- Passport System Sends for Delivery: If the application is approved, the Passport System sends the passport for delivery to the Delivery Service.
- Delivery Service Confirms Delivery: The Delivery Service confirms the delivery of the passport to the applicant.

5.7 Activity Diagram:

Activities:

- Applicant Submits Application: The process begins with the applicant submitting their passport application.
- Request Payment: The system prompts the applicant to make the required payment for the application.
- Make Payment: The applicant makes the payment through the Payment System.
- Notify Payment Success to Applicant: The Payment System notifies the applicant about the successful payment.
- Send Details for Verification: The Passport System sends the application details to the Verification System for further processing.
- Wait for Verification Result: The Passport System waits for the verification results from the Verification System.
- Approve Application/Reject Application: Based on the verification results, the Passport Office either approves or rejects the application.
- Generate Passport: If approved, the Passport Office generates the passport.
- Update Records: The Passport System updates its records to reflect the status of the application.

- Send Passport for Delivery: The Passport Office sends the passport to the Delivery Service for delivery to the applicant.
- Delivery Service Confirms Delivery: The Delivery Service confirms the successful delivery of the passport to the applicant.