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

Lab Record

Object Oriented Modeling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

S Pranav Ranganath

1BM22CS355

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 S Pranav Ranganath (1BM22CS355) during the 5th Semester Oct 24 - Jan 25.

Signature of the Faculty Incharge:

Prof. Prameetha Pai
Assistant Professor
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

GITHUB LINK: https://github.com/pranavsr29-ux/oomd_1BM22CS355

1. Hotel Management System

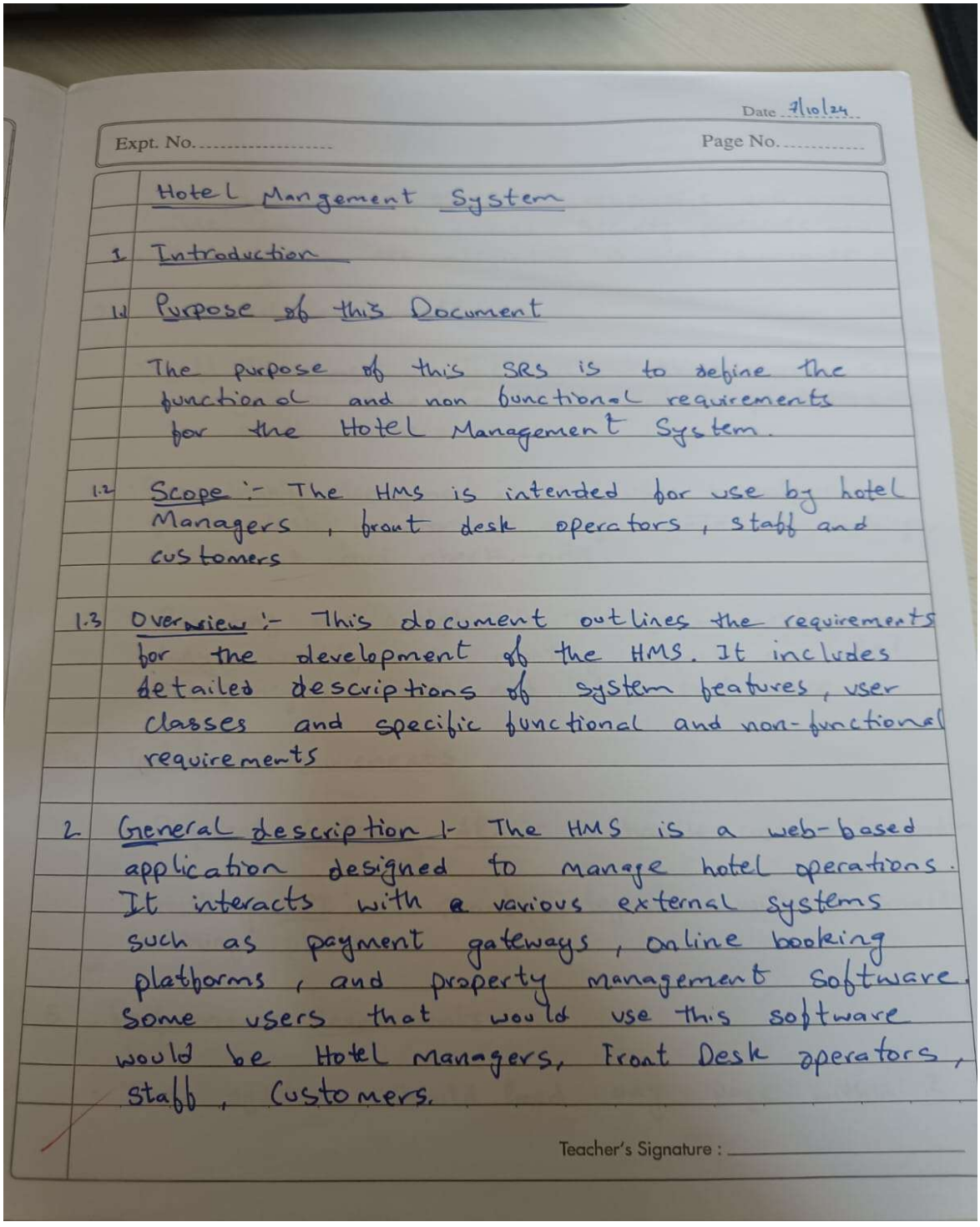
Problem Statement:

Design a software system to support a computerized hotel management network, catering to a chain of hotels operating under a single brand. Each hotel in the chain maintains its own local computer to manage bookings, room availability, and customer data. Reception desks at individual hotels communicate directly with their local computer systems to handle check-ins, check-outs, and in-house services. Front-desk staff input guest and reservation details.

An online booking portal communicates with a central server that coordinates reservations across all hotels, ensuring real-time updates for room availability. The online system allows guests to search for hotels, check room availability, make reservations, and process payments securely. Additionally, guests should be able to manage their reservations and request additional services through the portal.

The system requires robust recordkeeping, scalability to support multiple hotels, and strict security measures to protect customer data and transactions. It must handle concurrent booking requests and avoid double-booking. Each hotel will provide and maintain its software for local operations, while you are tasked with designing the software for the central reservation system, the online booking portal, and the network connecting individual hotels to the central server. The cost of the shared system will be apportioned to hotels based on the volume of online reservations they receive.

Software Requirement Specification Document



Date 7/10/24

Expt. No. ..... Page No. .....

Hotel Mangement System

1 Introduction

1.1 Purpose of this Document

The purpose of this SRS is to define the functional and non functional requirements for the Hotel Management System.

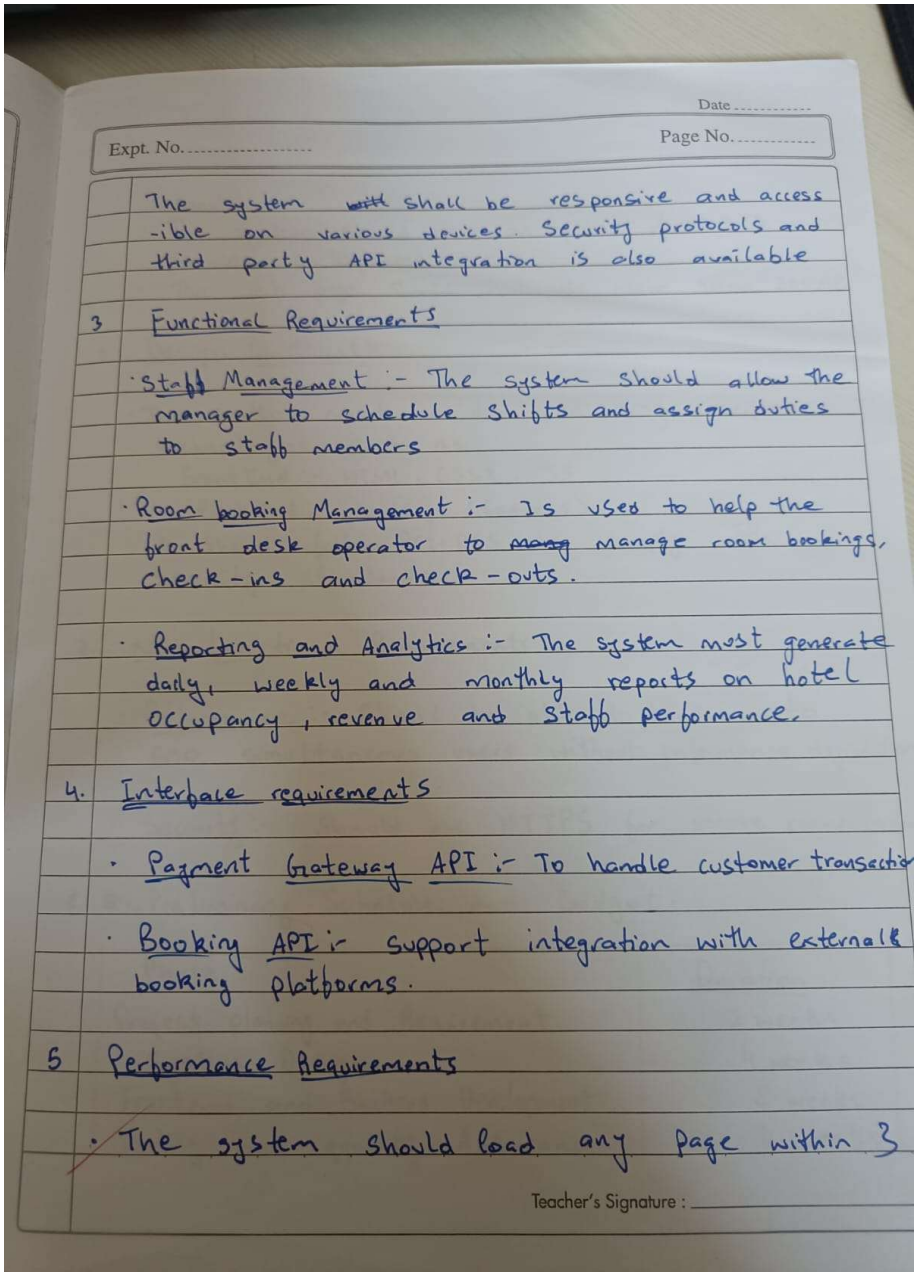
1.2 Scope :- The HMS is intended for use by hotel Managers, front desk operators, staff and customers

1.3 Overview :- This document outlines the requirements for the development of the HMS. It includes detailed descriptions of system features, user classes and specific functional and non-functional requirements

2. General description :- The HMS is a web-based application designed to manage hotel operations. It interacts with a various external systems such as payment gateways, online booking platforms, and property management software. Some users that would use this software would be Hotel Managers, Front Desk operators, staff, Customers.

Teacher's Signature : _____

Fig 1.1



Date .....

Expt. No. ..... Page No. .....

The system ~~with~~ shall be responsive and access -ible on various devices. Security protocols and third party API integration is also available

3 Functional Requirements

· Staff Management :- The system should allow the manager to schedule shifts and assign duties to staff members

· Room booking Management :- Is used to help the front desk operator to ~~many~~ manage room bookings, check-ins and check-outs.

· Reporting and Analytics :- The system must generate daily, weekly and monthly reports on hotel occupancy, revenue and staff performance.

4. Interface requirements

· Payment Gateway API :- To handle customer transactio

· Booking API :- Support integration with external booking platforms.

5 Performance Requirements

· The system should load any page within 3

Teacher's Signature : _____

Fig 1.2

Date .....

Expt. No. ..... Page No. .....

seconds

- The system should have an uptime of 99.9% and provide backup mechanism.
- The web page should not use more than 200MB

6. Design Constraints

- The system must be developed using specific technologies such as
  Front End :- HTML, CSS3, JS
  Back End :- Node.js (Express)
- Should follow PCI-DSS to comply with security standards.

7. Non-functional Requirements

- Performance :- Should be able to handle upto 500 simultaneous users without performance degradation.
- Security :- Should use HTTPS for secure communication.

8. 8. Preliminary Schedule and Budget.

Phase	Duration
Project planning and Requirement	2 weeks
System Design	4 weeks
Frontend and Backend Development	8 weeks
Testing and quality Assurance	4 weeks

Teacher's Signature : _____

Fig 1.3

Class Diagram

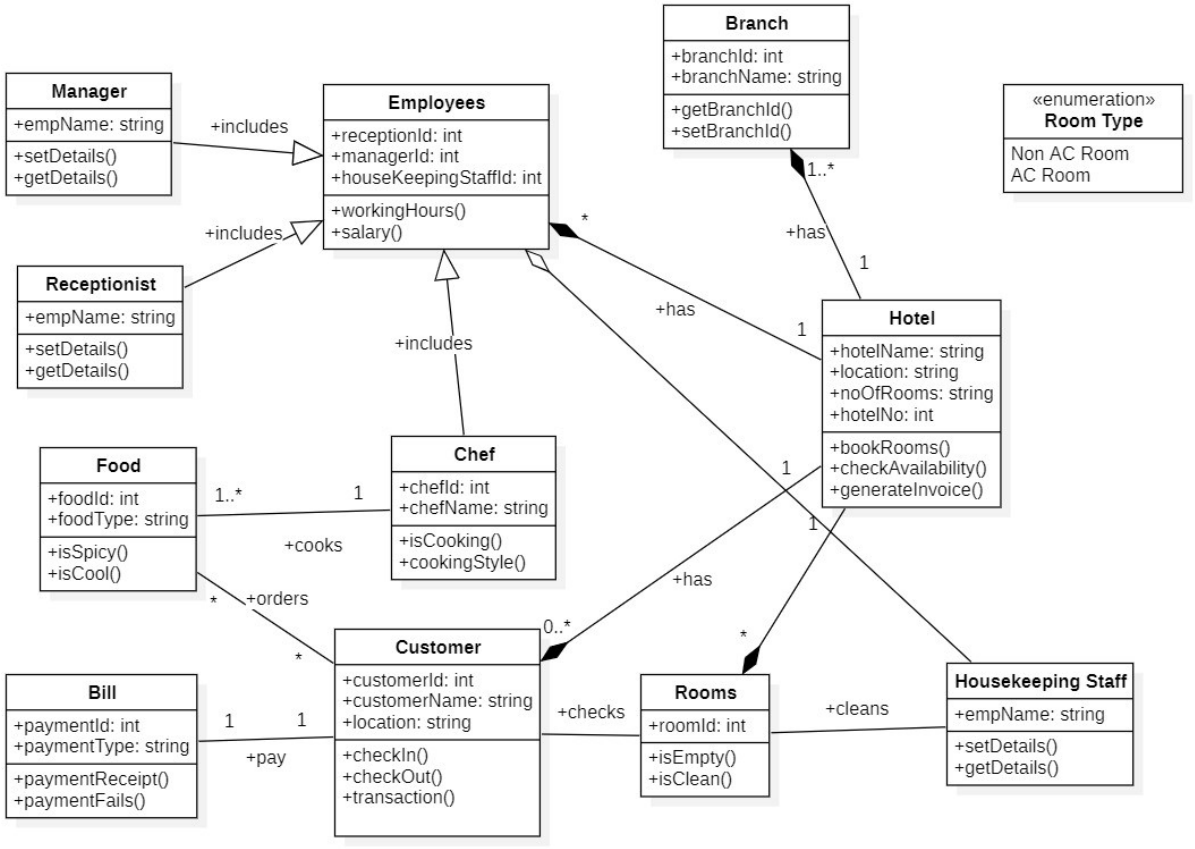


Fig 1.4

The class diagram illustrates the structure and interactions in a **Hotel Management System**, highlighting key roles, operations, and relationships:

1. **Employees**: A parent class encompassing all employee types like **Manager**, **Receptionist**, **Chef**, and **Housekeeping Staff**, each with specific attributes and methods while sharing common properties like working hours and salary.
2. **Branch and Hotel**: The **Branch** class represents the organizational hierarchy, with each branch having one or more hotels. The **Hotel** class contains details like location, number of rooms, and functionality to book rooms, check availability, and generate invoices.
3. **Rooms**: Each room is identified by a unique ID, its status (clean/empty), and belongs to a **Room Type** (AC or Non-AC). Rooms are maintained by the **Housekeeping Staff**.

4. **Customer**: Customers interact with the system to check in, check out, and handle transactions. They also book rooms and order food.
5. **Food and Chef**: Customers can place food orders, and chefs are responsible for cooking based on customer preferences.
6. **Bill**: Manages payment-related processes, including generating receipts and handling failed payments.

The system ensures smooth coordination among the components, with clearly defined relationships, such as:

- **Branch** contains **Hotels**, and **Hotels** manage **Rooms**.
- **Customers** book **Rooms**, interact with **Receptionists**, and make payments through the **Bill** system.
- **Housekeeping Staff** maintain **Rooms**, while **Chefs** prepare ordered **Food**.

State Diagram

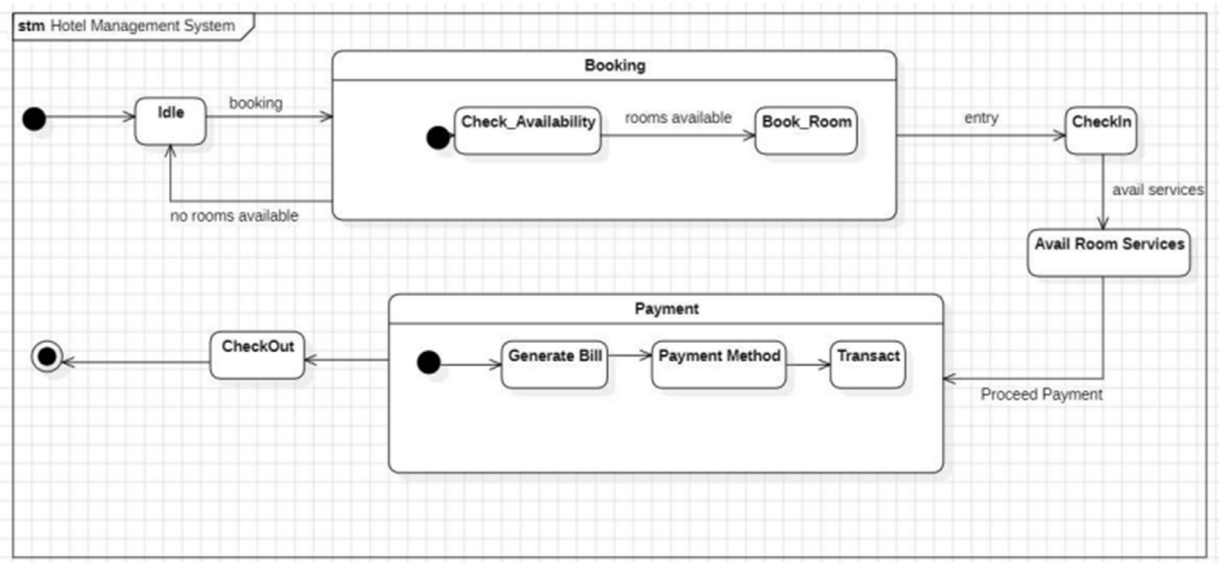


Fig 1.5

The statechart diagram illustrates the workflow for managing hotel bookings and payments within a Hotel Management System. It outlines the states and transitions involved in the booking and payment processes.

States:

Booking Process

- **Idle:** The initial state where the system is waiting for a booking request.
- **Check Availability:** The state where the system checks for room availability based on the customer's request.
- **Book_Room:** The state where a room is booked for the customer.
- **Checkin:** The state where the customer checks in to the hotel.
- **Avail Room Services:** The state where the customer avails of additional room services.

Payment Process:

- **CheckOut:** The state where the customer checks out of the hotel.
- **Generate Bill:** The state where the system generates a bill for the customer's stay and services.
- **Payment Method:** The state where the customer selects the preferred payment method.
- **Transact:** The state where the payment transaction is being processed.
- **Processed Payment:** The final state where the payment is successfully processed.

Transitions:

- From **Idle** to **Check Availability:** When a booking request is received.
- From **Check Availability** to **Book_Room:** If rooms are available.
- From **Check Availability** to **Idle:** If no rooms are available.
- From **Book_Room** to **Checkin:** When the customer arrives for check-in.
- From **Checkin** to **Avail Room Services:** If the customer requests additional services.
- From **Checkin** to **Payment:** When the customer is ready to check out.
- From **CheckOut** to **Generate Bill:** When the customer checks out.
- From **Generate Bill** to **Payment Method:** After the bill is generated.
- From **Payment Method** to **Transact:** When the customer selects a payment method.
- From **Transact** to **Processed Payment:** When the payment is successfully processed.

The statechart diagram provides a clear and concise overview of the hotel booking and payment process, highlighting the different stages involved and the possible transitions between states.

Use Case Diagram

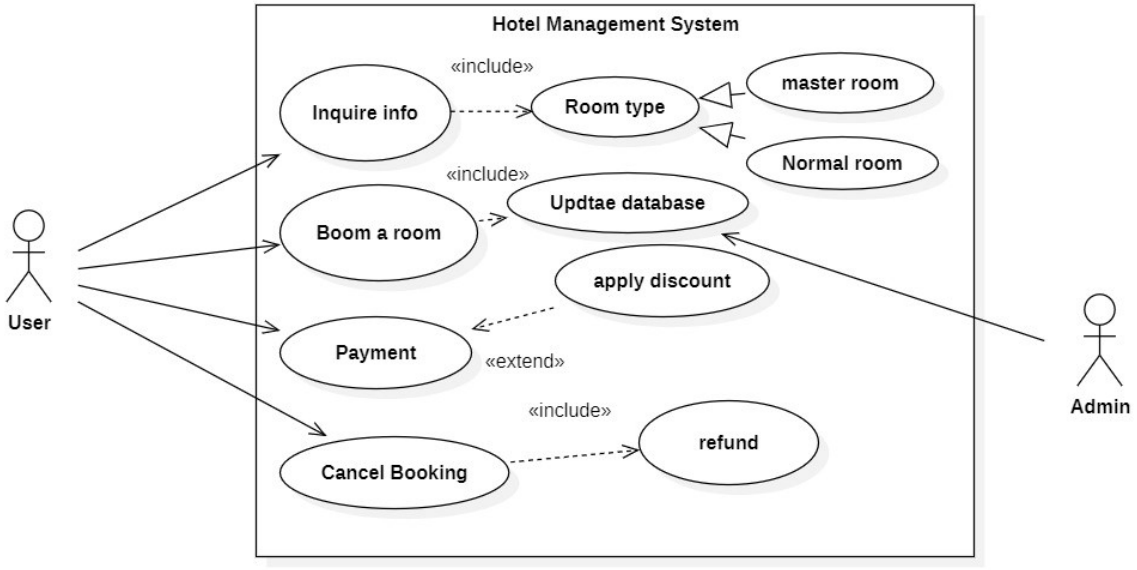


Fig 1.6

The use case diagram for the Hotel Management System represents the interactions between key actors—**User** and **Admin**—and the various services offered by the system. It highlights the system's modularity and functionality for managing hotel operations and guest interactions. Below is a detailed breakdown:

Actors:

1. **User:**
   The primary end-user interacting with the system to access services such as booking rooms, managing reservations, requesting services, and providing feedback.

2. **Admin:**
   Internal users who manage and fulfill the services requested by guests, such as room management, billing, housekeeping, and maintenance.

Use Cases:

1. **Room Booking:**

- Guests can book rooms online or at the reception.
- Extends to optional services like **Event Booking** (e.g., conferences, parties) and **Loyalty Program** (for frequent customers).

2. **Billing:**
    - Essential for processing payments, generating receipts, and managing failed transactions.
    - Directly connected to the check-in and check-out process and integrated with room services and additional guest services.
3. **Inquire info:**
    - Guests ask for info.
    - the hotel provides its services, covering aspects like housekeeping, maintenance, and overall guest satisfaction.

Relationships:

- **Extend:**
    - Payments Extends to applying discounts.
- **Include:**
    - Book a room includes billing to ensure a smooth guest departure process.
    - Inquire info includes room type.

System Features and Benefits:

- **Guest-Centric Services:** Enables seamless booking, personalized room service, and effortless payment processes for guests.
- **Staff Operations:** Simplifies tasks for hotel staff, ensuring efficient coordination between housekeeping, maintenance, and billing.

Sequence Diagram

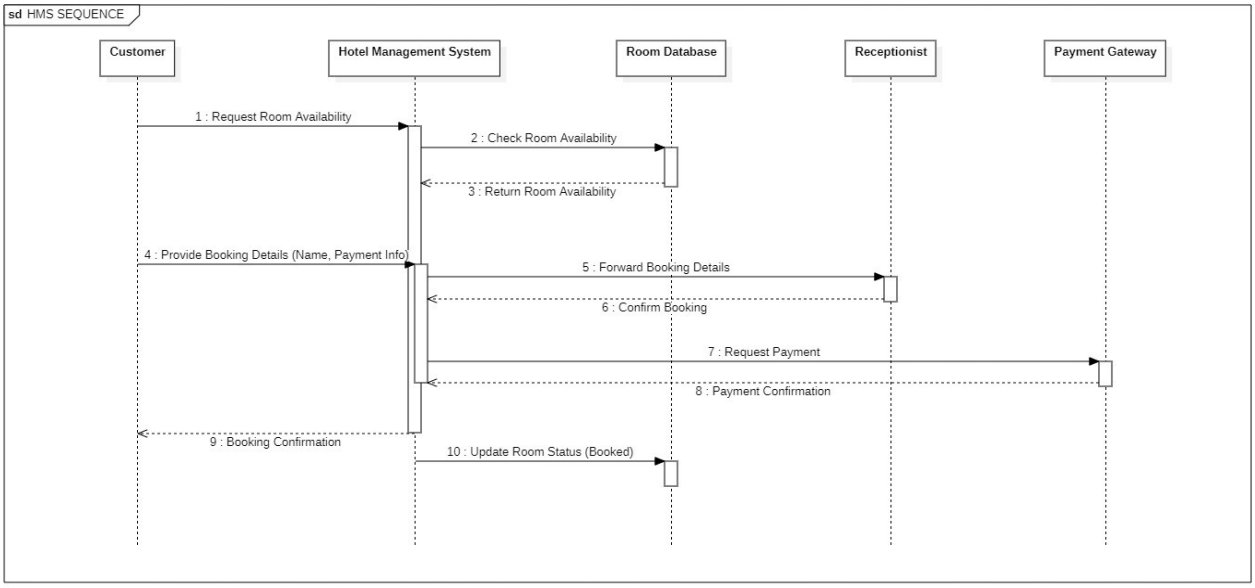


Fig 1.7

The sequence diagram for the Hotel Management System demonstrates the step-by-step process of room booking, showcasing interactions among key entities: Customer, Hotel Management System, Room Database, Receptionist, and Payment Gateway. Below is a detailed description:

Actors:

1. **Customer:**
   Initiates the room booking process by requesting room availability and providing booking details.

2. **Hotel Management System (HMS):**
   Serves as the central platform managing requests, verifying room availability, and coordinating between other entities.

3. **Room Database:**
   Validates room availability and updates room status after successful booking.

4. **Receptionist:**
   Confirms booking details and oversees the interaction between the system and the customer.

5. **Payment Gateway:**
   Processes payment transactions and confirms payment completion.

Key Interactions:

1. **Request Room Availability:**
   The customer begins by sending a request to the HMS to check available rooms.
2. **Check and Return Room Availability:**
   HMS queries the Room Database to validate the availability of rooms and sends the information back to the customer.
3. **Provide Booking Details:**
   The customer provides essential booking information such as name and payment details to proceed with the booking.
4. **Forward Booking Details:**
   HMS forwards the received booking details to the Receptionist for confirmation.
5. **Confirm Booking:**
   The Receptionist validates the booking and notifies the HMS of the confirmation.
6. **Request Payment:**
   The HMS communicates with the Payment Gateway to initiate the payment process.
7. **Payment Confirmation:**
   Upon successful payment, the Payment Gateway sends confirmation back to the HMS.
8. **Update Room Status:**
   HMS updates the Room Database, marking the room as booked.
9. **Booking Confirmation:**
   The customer receives confirmation of their booking, completing the sequence.

Features and Benefits:

- **Centralized System Coordination:**
  The HMS ensures seamless communication between the customer, room database, receptionist, and payment gateway.

- **Real-Time Room Availability Updates:**
  The Room Database efficiently tracks and updates room status to prevent double booking.
- **Secure Payment Processing:**
  Integration with the Payment Gateway ensures safe and reliable transaction handling.
- **Simplified Customer Experience:**
  Customers experience a streamlined process, from checking room availability to receiving booking confirmation.

Activity Diagram

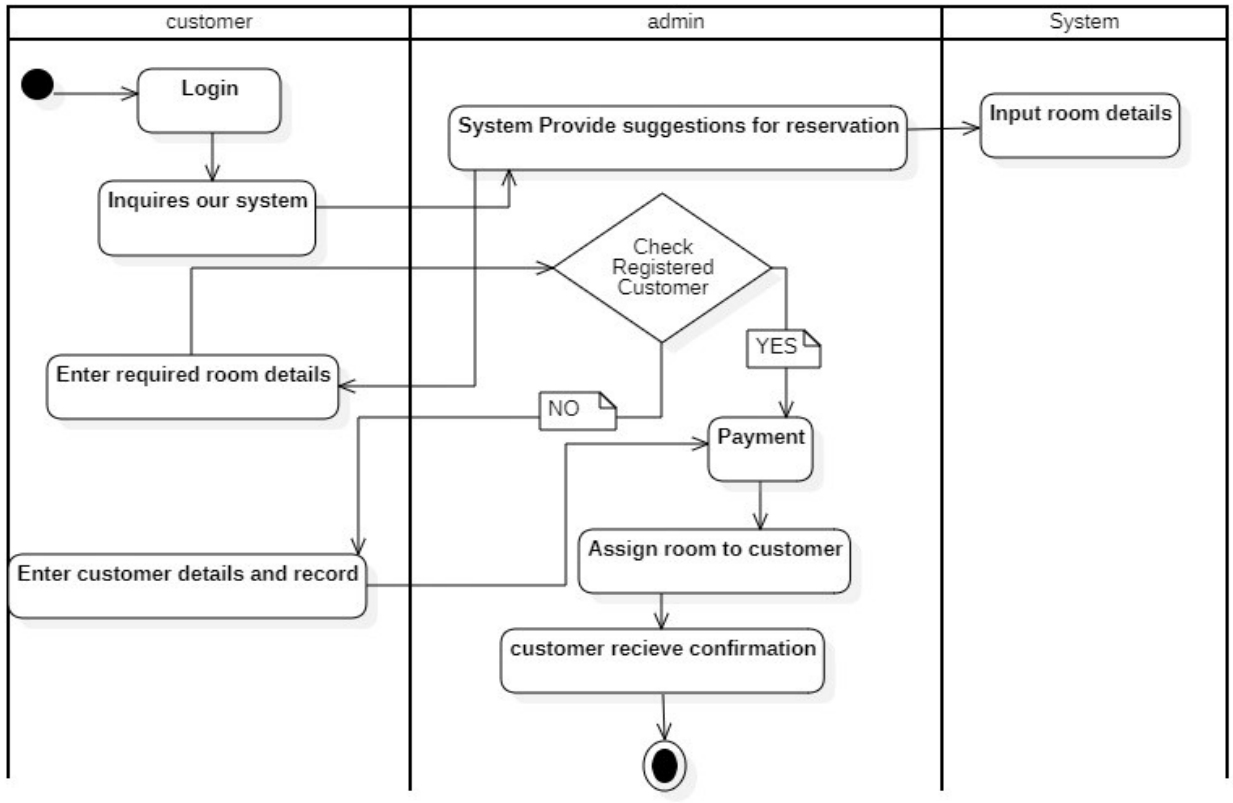


Fig 1.8

The activity diagram illustrates the process flow for room reservation in a hotel management system. It highlights the interactions between the customer, admin, and system components involved in the booking process.

Actors:

- **Customer:** The end-user who initiates the room reservation request.
- **Admin:** The system administrator responsible for managing room details and customer data.
- **System:** Represents the hotel management system itself, encompassing various functionalities.

Key Interactions:

1. **Customer Login:** The process starts with the customer logging into the system.
2. **System Provides Suggestions for Reservation:** Based on the customer's preferences or previous inquiries, the system suggests available rooms and their details.
3. **Input Room Details:** The customer selects a desired room and enters the necessary details, such as check-in/check-out dates and room preferences.
4. **Check Registered Customer:** The system verifies if the customer is already registered in the system.
    - **YES:** If the customer is registered, the system proceeds to the payment step.
    - **NO:** The customer is prompted to enter their details for registration.
5. **Payment:** The customer makes the payment for the reservation.
6. **Assign Room to Customer:** Upon successful payment, the system assigns the selected room to the customer.
7. **Customer Receives Confirmation:** The system sends a confirmation message to the customer, completing the reservation process.

Features and Benefits:

- **Centralized System Management:** The diagram showcases how the system coordinates various activities, from room availability checks to payment processing.
- **Efficient Customer Interaction:** The system provides a user-friendly interface for customers to easily inquire, select, and book rooms.
- **Secure Payment Processing:** The integration of payment functionality ensures secure and reliable transactions.
- **Enhanced Customer Experience:** The streamlined process and timely confirmations contribute to a positive customer experience.

Additional Considerations:

- **Error Handling:** The diagram could be further enhanced by incorporating error handling mechanisms to address potential issues like invalid payment or room unavailability.
- **Cancellation and Modification:** The process could be extended to include functionalities for cancelling or modifying reservations.

---

2. Credit Card Processing System

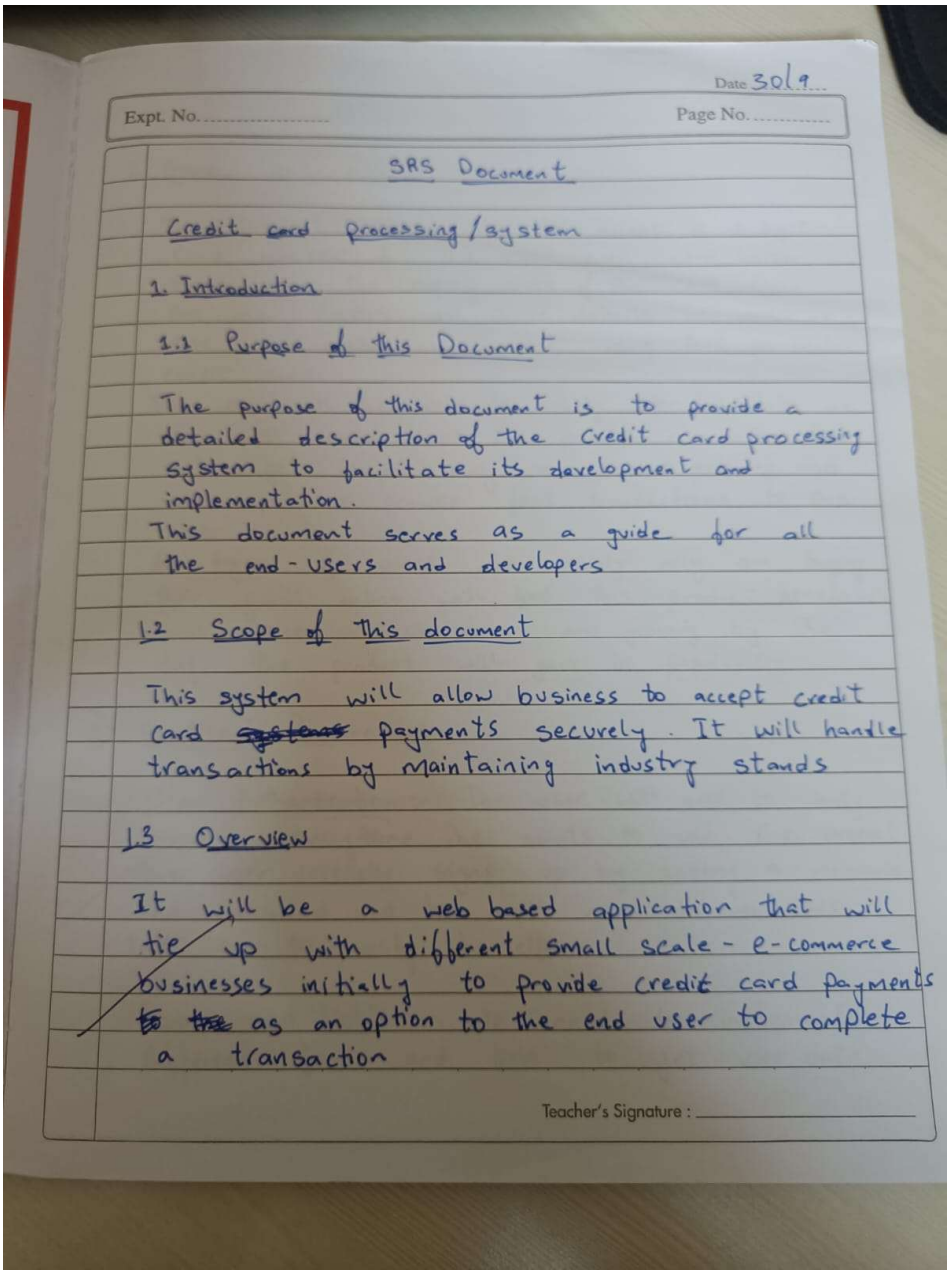
Problem Statement:

Design a software system to support a computerized credit card processing network for a large retail organization with multiple stores operating under a single brand. Each store maintains its own point-of-sale (POS) terminal to manage transactions locally. POS terminals at individual stores communicate directly with their local computer systems to handle sales, refunds, and void transactions. Sales associates input customer and transaction details.

An online payment gateway communicates with a central server that coordinates transaction processing and authorization across all stores, ensuring real-time updates for transaction status and inventory levels. The online system allows customers to make purchases online, process payments securely, and access their transaction history.

The system requires robust recordkeeping, scalability to support multiple stores, and strict security measures to protect customer data and transactions. It must handle concurrent transaction requests and avoid fraudulent activities. Each store will provide and maintain its software for local POS operations, while you are tasked with designing the software for the central authorization server, the online payment gateway, and the network connecting individual stores to the central server. The cost of the shared system will be apportioned to stores based on the volume of transactions they process.

Software Requirement Specification Document



Date 30/9...

Expt. No. ..... Page No. .....

SRS Document

Credit card processing / system

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to provide a detailed description of the credit card processing system to facilitate its development and implementation.
This document serves as a guide for all the end-users and developers

1.2 Scope of this document

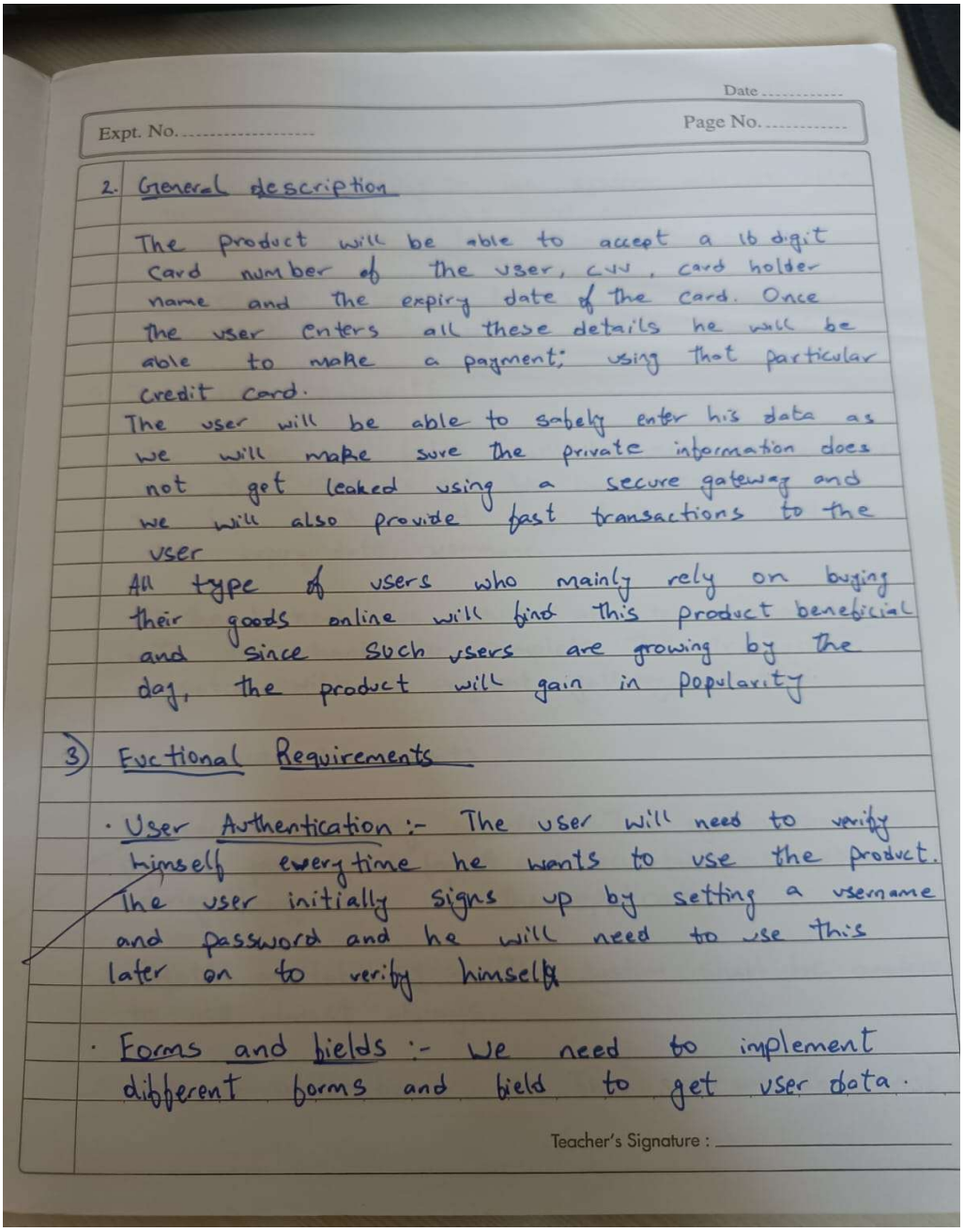
This system will allow business to accept credit card ~~systems~~ payments securely. It will handle transactions by maintaining industry stands

1.3 Overview

It will be a web based application that will tie up with different small scale - e-commerce businesses initially to provide credit card payments ~~to the~~ as an option to the end user to complete a transaction

Teacher's Signature : _____

Fig 2.1



Date .....

Expt. No. ..... Page No. .....

2. General description

The product will be able to accept a 16 digit card number of the user, CVV, card holder name and the expiry date of the card. Once the user enters all these details he will be able to make a payment; using that particular credit card.

The user will be able to safely enter his data as we will make sure the private information does not get leaked using a secure gateway and we will also provide fast transactions to the user

All type of users who mainly rely on buying their goods online will find this product beneficial and since such users are growing by the day, the product will gain in popularity

3) Fuctional Requirements

- User Authentication :- The user will need to verify himself everytime he wants to use the product. The user initially signs up by setting a username and password and he will need to use this later on to verify himself

- Forms and fields :- We need to implement different forms and field to get user data.

Teacher's Signature : _____

Fig 2.2

Date .....

Expt. No. ..... Page No. .....

- Transaction processing :- The system will be able to validate a credit card and allow users to initiate transactions.

- Cancellation policy :- The system will allow users to get a refund on their payments.

4 Interface Requirements

- The formas and fields will be used to communicate with the user to get his bank details The form will consist of :- Bank name, Card no, Cvv, expiry date, username.

- At the backend we will be able to communicate with the bank to complete the transaction, integration with a payment gateway, and API's

5. Performance Requirements

- Transaction processing speeds :- The system shall provide fast speed for 4 and acceptable speeds during heavy load of the application.

- System availability :- The system shall be available to use almost always.

- Resource utilization :- The system shall not

Teacher's Signature : _____

Fig 2.3

Date .....

Expt. No. ..... Page No. .....

use too much of the system's resource.

6. Design Constraints

- We need to use fine grain technology to improve the speed of our system
- Decreasing in redondancy will improve security, we need to store all the personal information in the inner layer of our system so that it can't be easily breached.
- We could use RSA to encrypt our data and store it.

7. Non functional attributes

1) Performance :- The system shall be able to handle a decent amounter of transactions/min

2) Security :- The system shall be able to provide safe and secure transactions, without any breach of data

3) Usability :- The system shall provide a friendly user interface, which can be easily used.

4) Backup and Recovery.

Teacher's Signature : _____

Fig 2.4

Date .....

Expt. No. ..... Page No. .....

8. Preliminary Schedule and Budget.

Phase	Duration
• Project initiation	2 weeks
• Requirements gathering	4 weeks
• System Design	6 weeks
• Development	12 weeks
• Testing	4 weeks
• Deployment	2 weeks
• Training and documentation	2 weeks
• Deployment support	4 weeks

Budget breakdown	Estimated cost
• Personnel Costs	$ 200,000
• Software licenses	$ 10,000
• Infrastructure	$ 15,000
• Security	$ 5,000
• Training	$ 5,000
• Marketing and Launch	$ 10,000.

Teacher's Signature : _____

Fig 2.5

Class Diagram

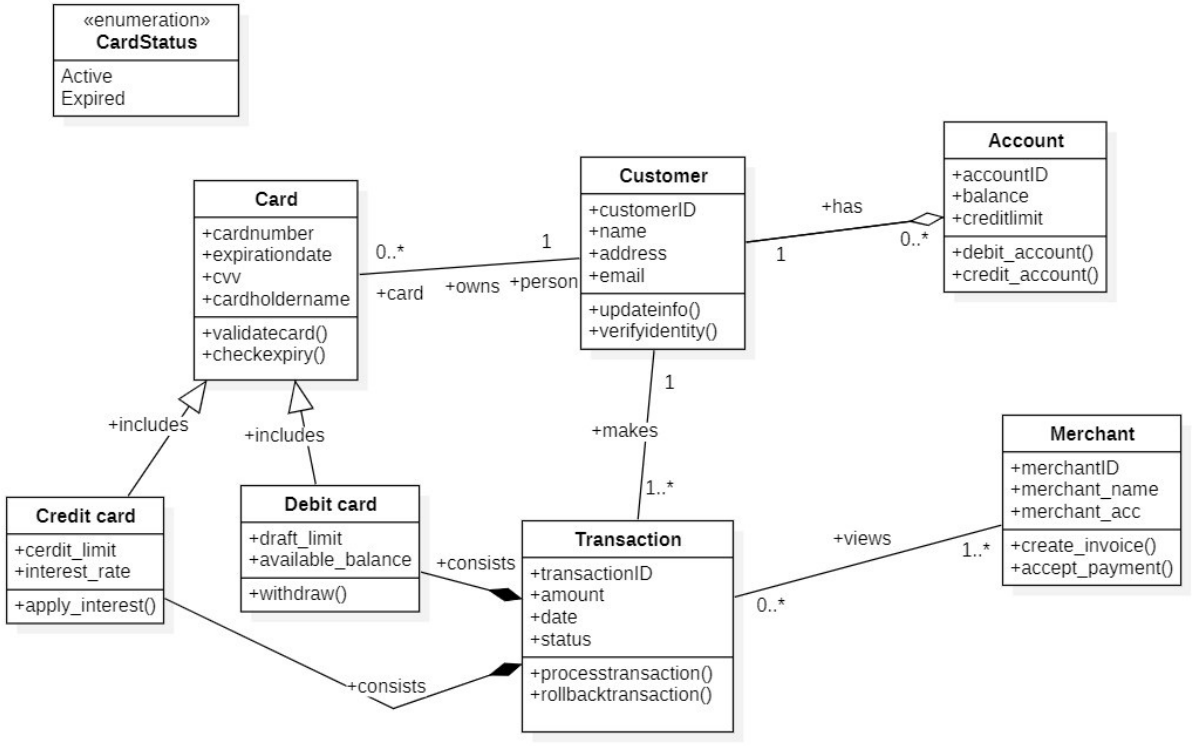


Fig 2.6

The class diagram illustrates the structure and interactions within a credit card processing system, highlighting key roles, operations, and relationships.

Key Classes:

- **UserAccount:** Represents user accounts with attributes like Name, AccountName, and Password.
- **Merchant:** Represents employees with attributes like Name, AccountName, and Password.
- **Admin:** A specialized type of Employee with administrative privileges.
- **Credit Card:** Represents credit card details with attributes like Name, PIN, and Number.
- **Transaction:** Represents a single financial transaction, with attributes like TID, Type (e.g., purchase, refund), and Amount.
- **Transaction History:** Records a collection of transactions associated with a specific account.
-

Relationships:

- **Inheritance:** Employee inherits from UserAccount, and Admin inherits from Employee, indicating that Admins are a specific type of Employee.
- **Composition:** Credit Card is composed of Credit Card Number, meaning the Credit Card Number cannot exist independently.
- **Aggregation:** Transaction History is aggregated by Transaction, meaning a Transaction History can contain multiple Transactions.
- **Associations:** Various associations exist between classes like "client," "processes," "makes," "linked," and "records." These indicate relationships between different entities, such as a UserAccount making a Transaction or a Payment Gateway processing a Transaction.

The structure ensures smooth coordination between different components of the credit card processing system, with clearly defined relationships and responsibilities for each class.

State Diagram

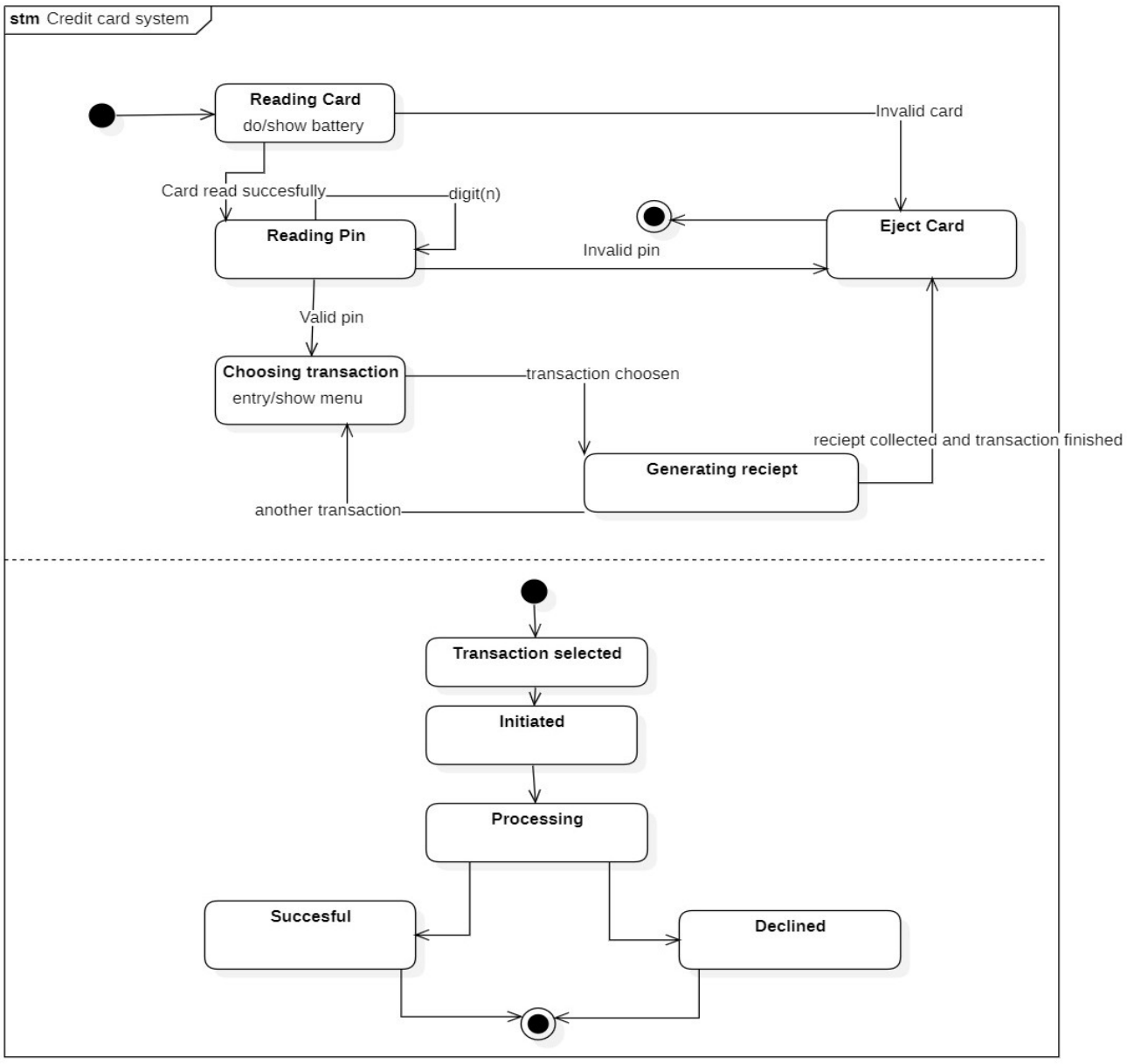


Fig 2.7

The statechart diagram illustrates the workflow of a credit card system. It outlines the various states and transitions involved in a typical credit card transaction.

States:

- **Reading Card:** The initial state where the system reads the credit card information.
- **Reading Pin:** The state where the system prompts the user to enter their PIN.

- **Choosing Transaction:** The state where the user selects the type of transaction they wish to perform (e.g., purchase, cash withdrawal).
- **Generating Receipt:** The state where the system generates a receipt for the completed transaction.
- **Transaction Selected:** The state where the user has chosen the type of transaction.
- **Initiated:** The state where the transaction has been initiated and is being processed.
- **Processing:** The state where the transaction is being processed by the system.
- **Successful:** The state where the transaction has been successfully completed.
- **Declined:** The state where the transaction has been declined.

Transitions:

- **Reading Card:**
  - If the card is read successfully, the system transitions to the "Reading Pin" state.
  - If the card is invalid, the system transitions to the "Eject Card" state.
- **Reading Pin:**
  - If the entered PIN is valid, the system transitions to the "Choosing Transaction" state.
  - If the entered PIN is invalid, the system transitions to the "Eject Card" state.
- **Choosing Transaction:**
  - Once the user selects a transaction, the system transitions to the "Transaction Selected" state.
- **Transaction Selected:**
  - The system transitions to the "Initiated" state to begin processing the transaction.
- **Initiated:**
  - The system transitions to the "Processing" state to carry out the transaction.
- **Processing:**
  - If the transaction is successful, the system transitions to the "Successful" state.
  - If the transaction is declined, the system transitions to the "Declined" state.
- **Successful:**
  - The system transitions to the "Generating Receipt" state to generate a receipt for the successful transaction.
- **Generating Receipt:**

- Once the receipt is generated, the system can either transition to the "Reading Card" state for another transaction or to the end state.

Use Case Diagram

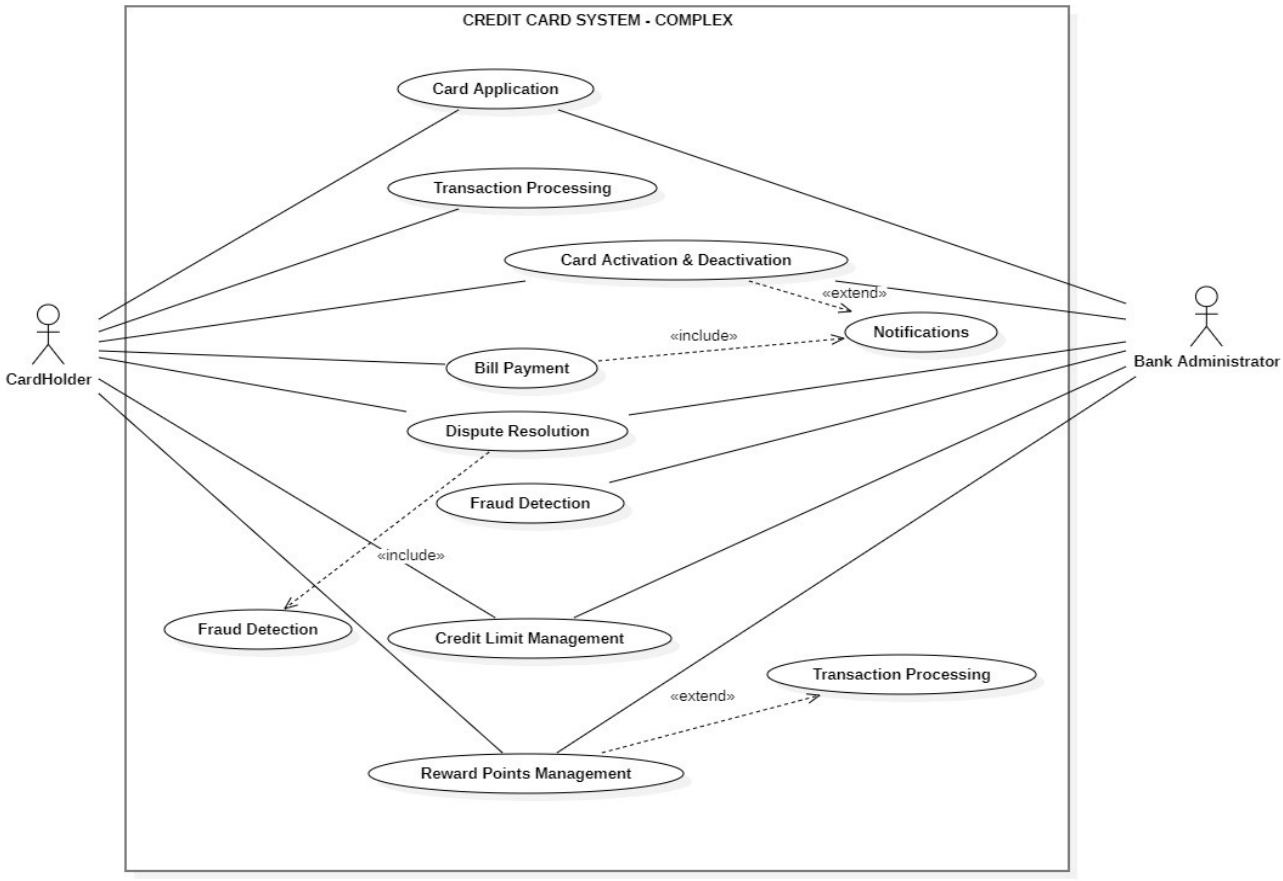


Fig 2.8

The use case diagram illustrates the various functionalities and interactions within a complex credit card system. It highlights the different use cases available to both Cardholders and Bank Administrators, as well as the relationships between these use cases.

Actors:

- **CardHolder:** Represents the end-user of the credit card, who interacts with the system for various purposes.
- **Bank Administrator:** Represents the system administrator responsible for managing the credit card system and its functionalities.

Use Cases:

- **Card Application:** The process of applying for a new credit card.
- **Transaction Processing:** Handles the processing of credit card transactions, including purchases, payments, and refunds.
- **Card Activation & Deactivation:** Allows for the activation and deactivation of credit cards.
- **Bill Payment:** Enables cardholders to make bill payments.
- **Dispute Resolution:** Handles disputes related to transactions or charges.
- **Fraud Detection:** Implements mechanisms to detect and prevent fraudulent activities.
- **Credit Limit Management:** Manages credit limits for individual cardholders.
- **Reward Points Management:** Manages the accumulation and redemption of reward points associated with credit card usage.
- **Notifications:** Sends notifications to cardholders and administrators regarding transactions, alerts, and other important information.

Relationships:

- **Include:** The "Fraud Detection" use case is included in both "Transaction Processing" and "Dispute Resolution," indicating that fraud detection is an integral part of both processes.
- **Extend:** The "Notifications" use case extends "Card Activation & Deactivation," indicating that notifications can be sent as part of the card activation and deactivation process.

The use case diagram provides a comprehensive overview of the credit card system's functionalities, highlighting the interactions between different actors and use cases, as well as the relationships between various components of the system.

Sequence Diagram

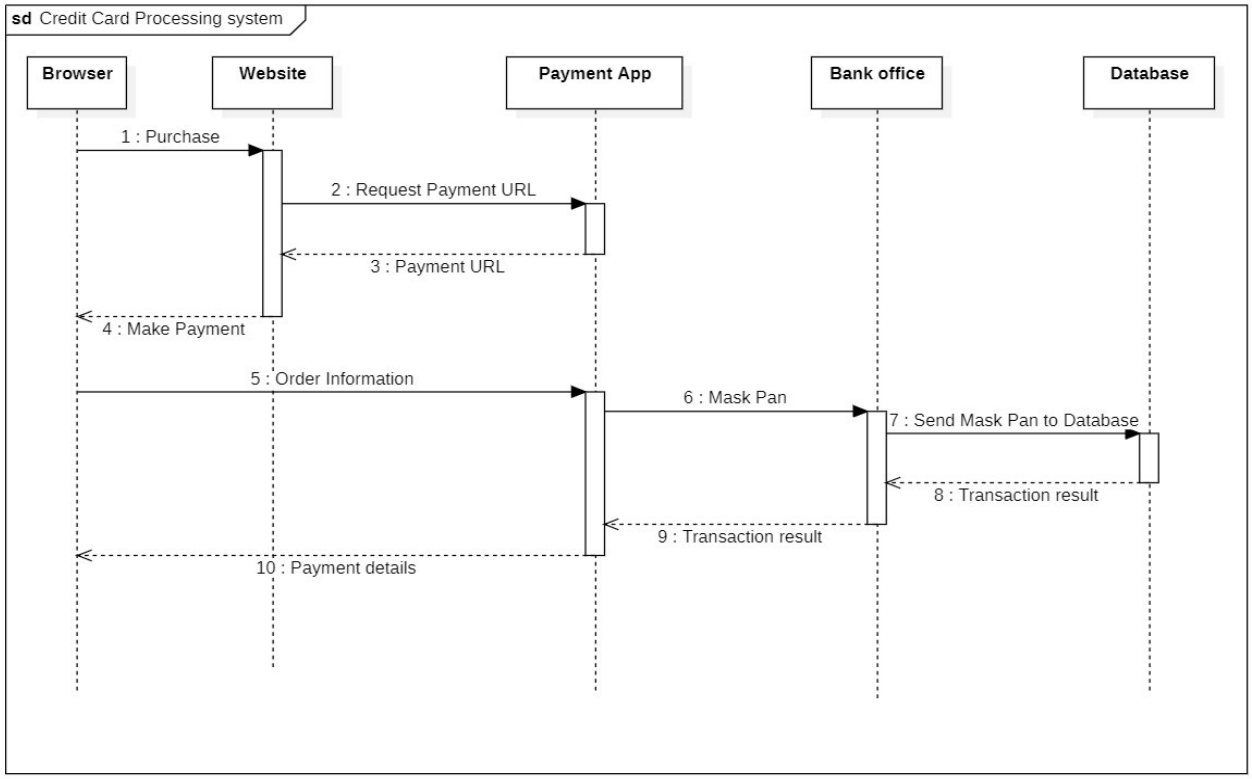


Fig 2.9

This sequence diagram illustrates the flow of interactions involved in a credit card processing system during an online purchase. It outlines the sequence of messages exchanged between different components of the system, including the Browser, Website, Payment App, Bank Office, and Database.

Actors:

- **Browser:** Represents the user's web browser.
- **Website:** The online store where the user is making a purchase.
- **Payment App:** A third-party application used to process the payment.
- **Bank Office:** The bank's system that handles the actual transaction processing.
- **Database:** The central repository for storing transaction data.

Sequence of Events:

1. **Purchase:** The user initiates a purchase on the website by selecting items and proceeding to checkout.
2. **Request Payment URL:** The website requests a payment URL from the Payment App.

3. **Payment URL:** The Payment App sends the payment URL back to the website.
4. **Make Payment:** The user is redirected to the Payment App using the provided URL.
5. **Order Information:** The Payment App receives order information from the website.
6. **Mask Pan:** The Payment App masks the credit card number (Pan) before sending it to the Bank Office for security purposes.
7. **Send Mask Pan to Database:** The Bank Office sends the masked Pan to the Database for processing.
8. **Transaction Result:** The Database processes the transaction and sends the result (success or failure) back to the Bank Office.
9. **Transaction Result:** The Bank Office sends the transaction result to the Payment App.
10. **Payment Details:** The Payment App sends the payment details (including the result) back to the website.

The sequence diagram provides a clear and concise overview of the steps involved in a typical online credit card transaction, highlighting the interactions between different components of the system and the flow of information during the process.

Activity Diagram

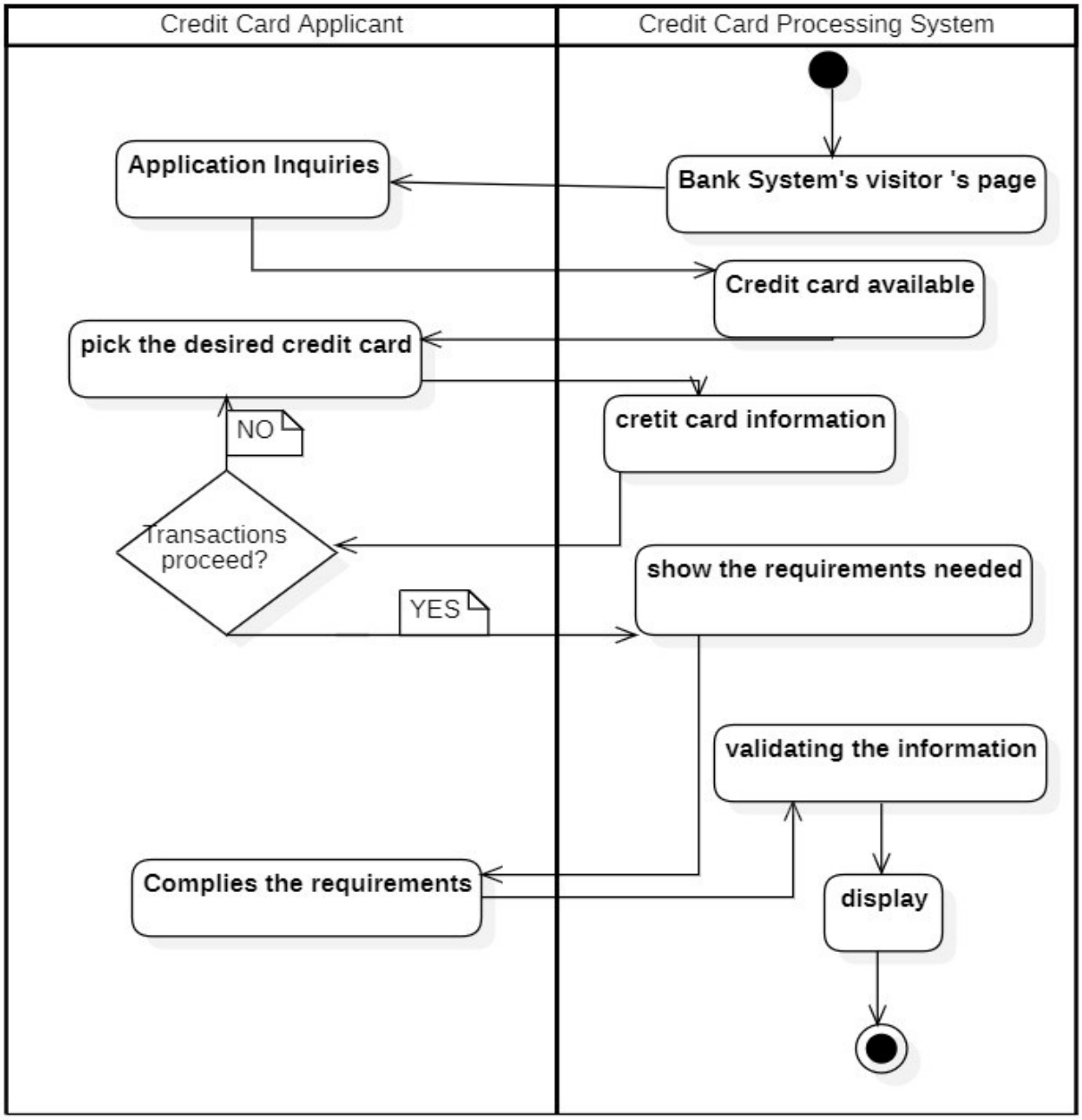


Fig 2.10

The activity diagram illustrates the process flow for a credit card application within a credit card processing system. It outlines the steps involved from the applicant's perspective and the system's response.

Actors:

- **Credit Card Applicant:** The individual who is applying for a credit card.
- **Credit Card Processing System:** The system that handles the application process.

Activities:

- **Application Inquiries:** The applicant initiates the process by making inquiries about available credit cards.
- **Bank System's Visitor's Page:** The applicant is directed to the bank's system's visitor's page to view available credit card options.
- **Credit Card Available:** The applicant selects a desired credit card.
- **Credit Card Information:** The system provides the applicant with information about the selected credit card, including requirements and terms.
- **Transactions Proceed?** The system checks if the applicant meets the requirements for the selected credit card.
    - **NO:** The system informs the applicant that they do not meet the requirements and may suggest alternative options.
    - **YES:** The applicant proceeds with the application process.
- **Show the Requirements Needed:** The system displays the specific requirements that the applicant needs to fulfill.
- **Complies the Requirements:** The applicant provides the necessary information and documents to comply with the requirements.
- **Validating the Information:** The system validates the information provided by the applicant.
- **Display:** The system displays the outcome of the application process, indicating whether the application is approved or denied.

The activity diagram provides a clear and concise overview of the credit card application process, highlighting the key steps involved and the interactions between the applicant and the credit card processing system.

---

3. Library Management System

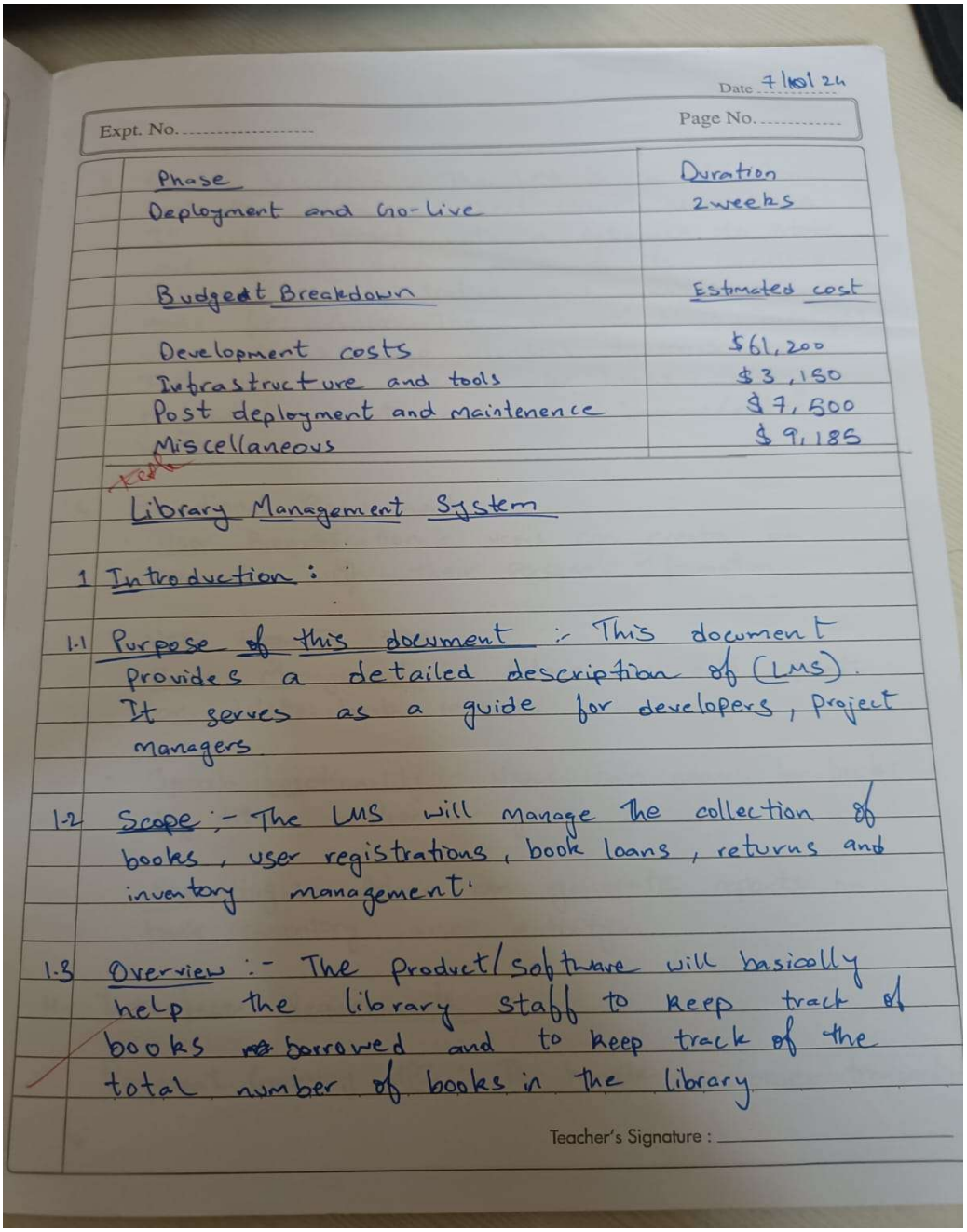
Problem Statement:

Design a software system to support a computerized library management network for a large library system with multiple branches operating under a single administration. Each branch maintains its own local computer system to manage book lending, returns, and member records. Library staff at individual branches interact directly with their local computer systems to handle checkouts, check-ins, and member inquiries. Library staff input book and member details.

A central library management system communicates with a central server that coordinates book availability across all branches, ensures real-time updates for book locations, and manages member accounts centrally. The central system allows library members to search for books, place holds, renew loans, and access their borrowing history online.

The system requires robust recordkeeping, scalability to support multiple branches, and strict security measures to protect member data and library assets. It must handle concurrent requests for books and prevent overdue fines. Each branch will provide and maintain its software for local library operations, while you are tasked with designing the software for the central library management system, the online member portal, and the network connecting individual branches to the central server. The cost of the shared system will be apportioned to branches based on the volume of book transactions they process.

Software Requirement Specification Document



Date 7/10/24

Expt. No. ..... Page No. .....

Phase	Duration
Deployment and Go-Live	2 weeks

Budget Breakdown	Estimated cost
Development costs	$61,200
Infrastructure and tools	$3,150
Post deployment and maintenence	$7,500
Miscellaneous	$9,185

Library Management System

1 Introduction :

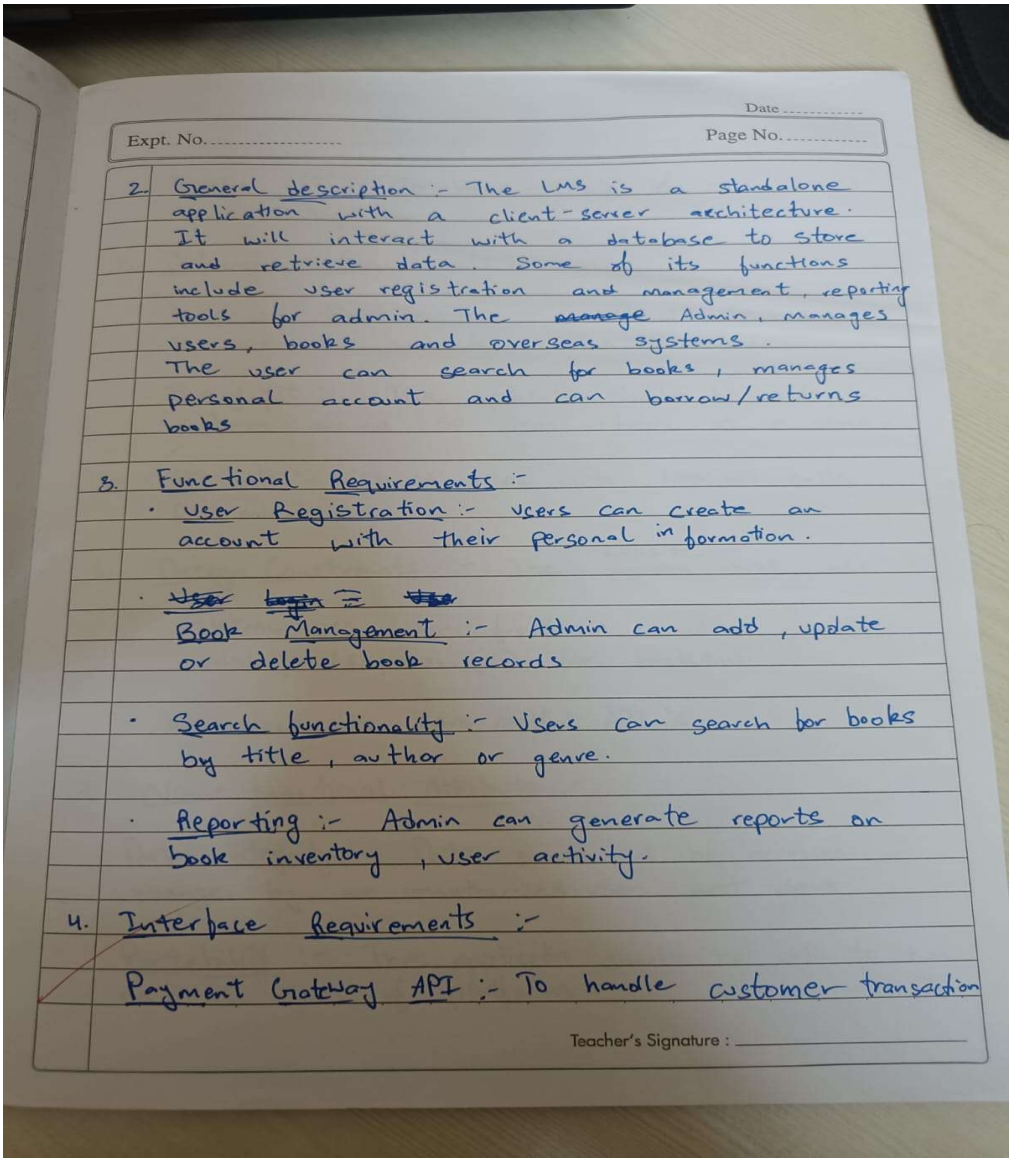
1.1 Purpose of this document :- This document provides a detailed description of (LMS). It serves as a guide for developers, project managers.

1.2 Scope :- The LMS will manage the collection of books, user registrations, book loans, returns and inventory management.

1.3 Overview :- The product/software will basically help the library staff to keep track of books borrowed and to keep track of the total number of books in the library.

Teacher's Signature : _____

Fig 3.1



Date .....

Expt. No. ..... Page No. .....

2. General description :- The LMS is a standalone application with a client-server architecture. It will interact with a database to store and retrieve data. Some of its functions include user registration and management, reporting tools for admin. The ~~manage~~ Admin, manages users, books and overseas systems. The user can search for books, manages personal account and can borrow/returns books

3. Functional Requirements :-

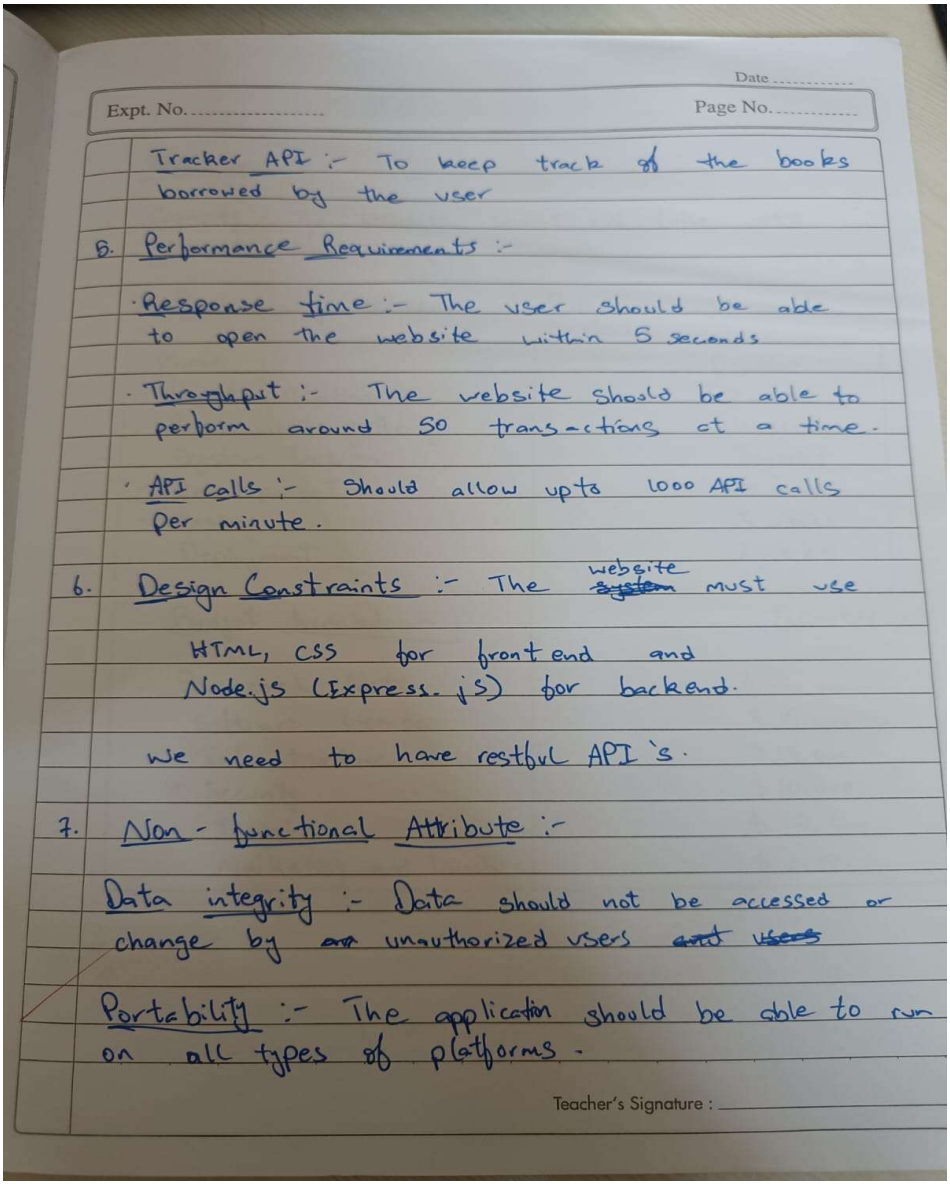
- User Registration :- Users can create an account with their personal information.
- ~~User login~~ Book Management :- Admin can add, update or delete book records
- Search functionality :- Users can search for books by title, author or genre.
- Reporting :- Admin can generate reports on book inventory, user activity.

4. Interface Requirements :-

Payment Gateway API :- To handle customer transaction

Teacher's Signature : _____

Fig 3.2



Date .....

Expt. No. ..... Page No. .....

Tracker API :- To keep track of the books borrowed by the user

5. Performance Requirements :-

- Response time :- The user should be able to open the website within 5 seconds
- Throughput :- The website should be able to perform around 50 transactions at a time.
- API calls :- Should allow upto 1000 API calls per minute.

6. Design Constraints :- The ~~system~~ website must use HTML, CSS for front end and Node.js (Express.js) for backend.

We need to have restful API's.

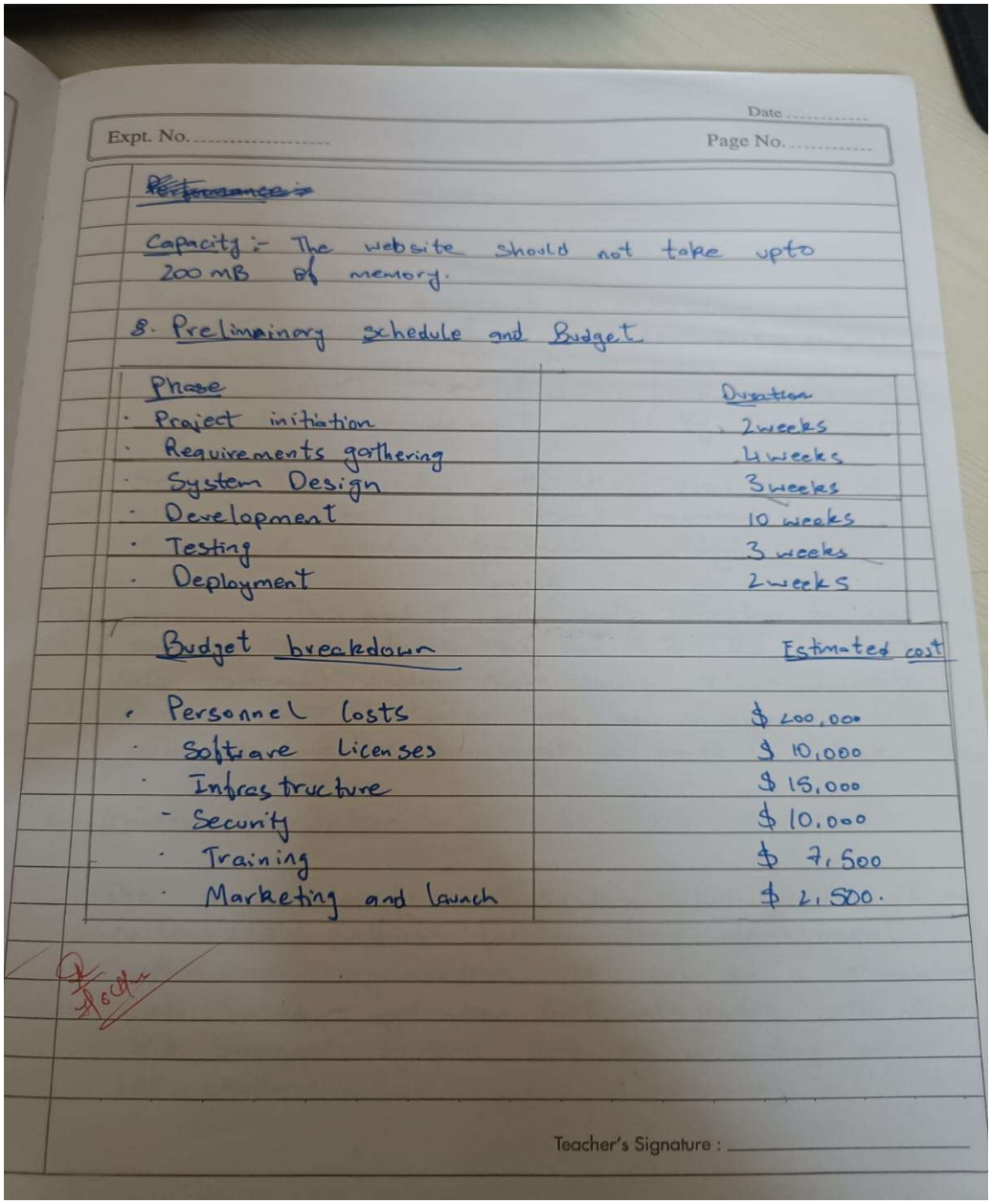
7. Non-functional Attribute :-

Data integrity :- Data should not be accessed or change by ~~an~~ unauthorized users ~~and users~~

Portability :- The application should be able to run on all types of platforms.

Teacher's Signature : _____

Fig 3.3



Date .....

Expt. No. ..... Page No. .....

~~Performance :-~~

Capacity :- The website should not take upto 200 MB of memory.

8. Preliminary schedule and Budget

Phase	Duration
· Project initiation	2 weeks
· Requirements gathering	4 weeks
· System Design	3 weeks
· Development	10 weeks
· Testing	3 weeks
· Deployment	2 weeks

Budget breakdown	Estimated cost
· Personnel Costs	$ 200,000
· Software Licenses	$ 10,000
· Infrastructure	$ 15,000
- Security	$ 10,000
· Training	$ 7,500
· Marketing and launch	$ 2,500.

Teacher's Signature : _____

Fig 3.4

Class Diagram

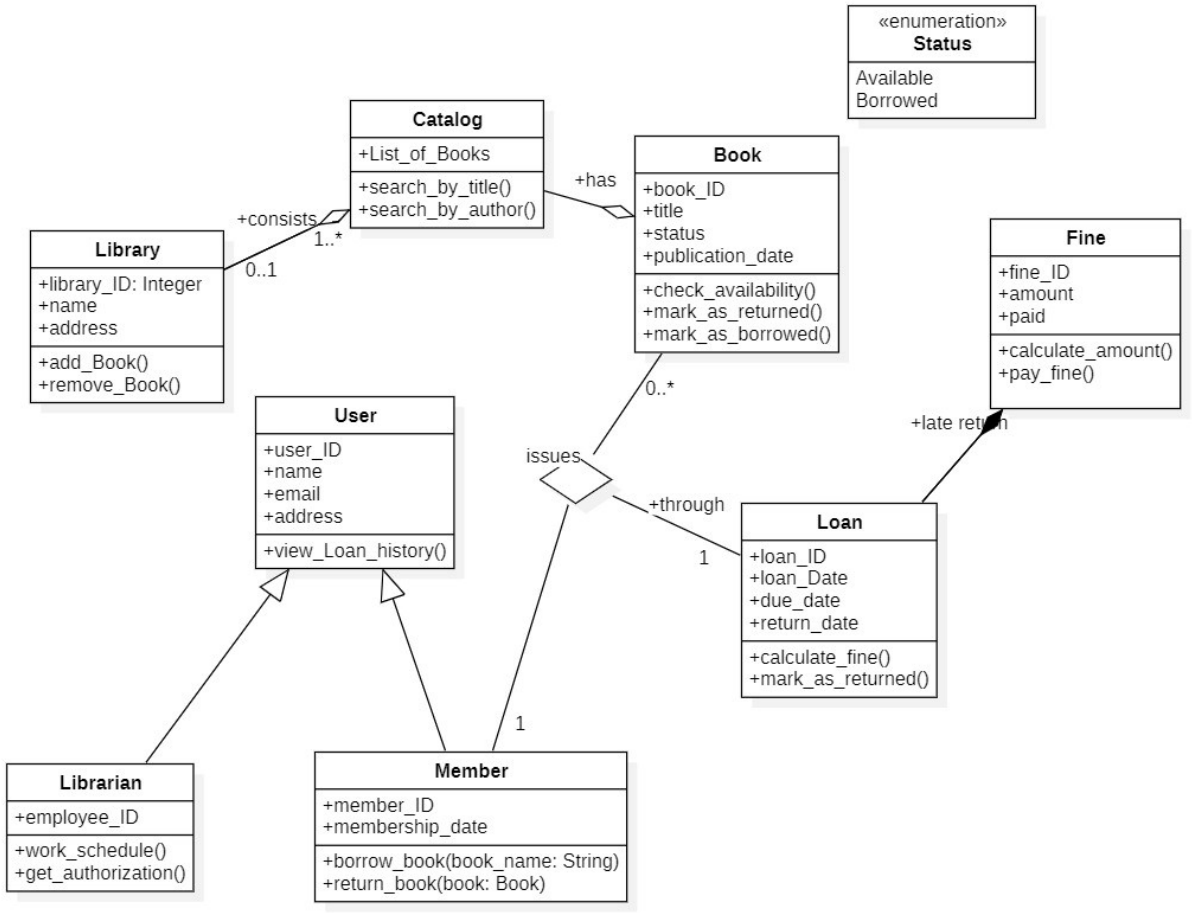


Fig 3.5

The class diagram illustrates the structure and relationships within a Library Management System. It highlights the key entities, their attributes, and the interactions between them.

Key Classes

- **Library:** Represents a library with attributes like library_name, library_id, and has a relationship with Divisions.
- **Book:** Represents a book with attributes like book_id, name, author, publisher, price, and date_of_purchase. It has a relationship with BookType.
- **BookType:** An enumeration class representing different types of books (e.g available,borrowed).
- **Member:** Represents library members with attributes like mem_id, name, address, and branchName.

- **Librarian:** Represents library staff with attributes like name, password, and address.
- **Loan:** Represents a transaction related to book borrowing or returning, with attributes like transaction_id, book_id, member_id, issue_date, and due_date.
- **Bill:** Represents a bill for library services, with attributes like bill_id, mem_id, and amount.
- **Student:** Represents student members with attributes like SName, SUSN, SDept, and SSem.

Relationships

- **Library has Divisions:** A library can have multiple divisions.
- **Library has Books:** A library maintains a collection of books.
- **Book has BookType:** Each book belongs to a specific type.
- **Member requests Book:** Members can request to borrow books.
- **Member pays Bill:** Members pay for library services.
- **Librarian creates Transaction:** Librarians manage the book borrowing and returning process.
- **Transaction refers Bill:** Transactions can be associated with a bill.
- **Staff includes Teaching_staff and Non_teaching_staff:** Teaching_staff and Non_teaching_staff are types of Staff.
- **Student includes SName, SUSN, SDept, and SSem:** Students have additional attributes like student name, student ID, department, and semester.

The class diagram provides a clear and concise representation of the entities and their relationships within the library management system, laying the foundation for the design and implementation of the software.

State Diagram

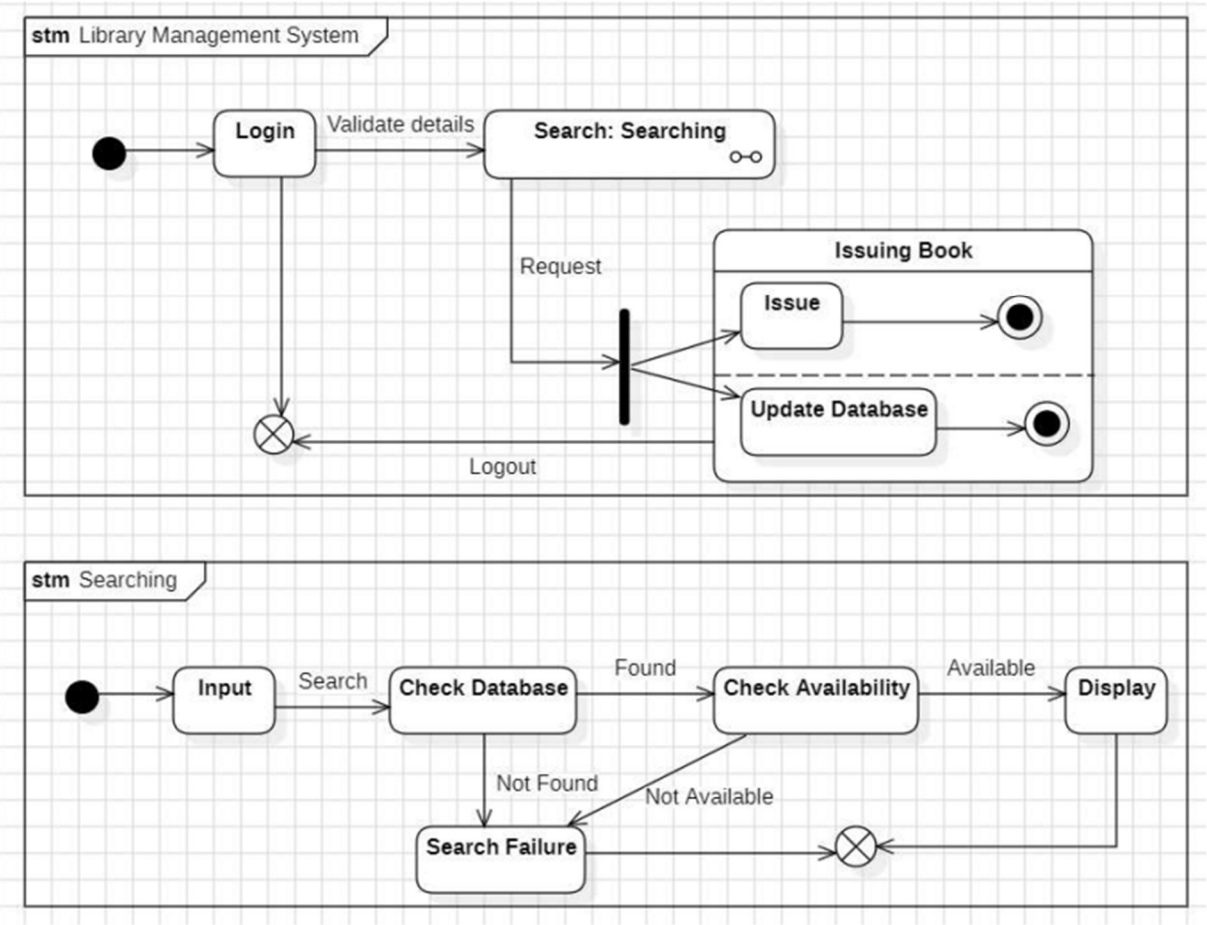


Fig 3.6

The statechart diagram illustrates the workflow within a Library Management System, focusing on the book searching and issuing processes. It outlines the states and transitions involved in these operations.

Top Level (Library Management System)

- **States:**
  - **Login:** The initial state where the user logs into the system.
  - **Search: Searching:** The state where the system is actively searching for books based on the user's query.
  - **Issuing Book:** The state where the book is being issued to the user.
- **Transitions:**
  - From **Login** to **Search: Searching:** After successful login, the system transitions to the searching state.

- From **Search: Searching** to **Issuing Book:** If a book is found and available, the system transitions to the issuing state.
- From **Issuing Book** to **Issue:** The book is issued to the user.
- From **Issuing Book** to **Update Database:** The system updates the database to reflect the book issue.
- From any state to **Logout:** The user logs out of the system.

Bottom Level (Searching)

- **States:**
  - **Input:** The initial state where the user enters the search criteria.
  - **Search:** The state where the system is searching the database based on the input criteria.
  - **Check Database:** The state where the system checks the database for matching books.
  - **Found:** The state where a book matching the criteria is found.
  - **Check Availability:** The state where the system checks if the found book is available for issue.
  - **Available:** The state where the found book is available for issue.
  - **Display:** The state where the system displays the book details to the user.
  - **Not Found:** The state where no books matching the criteria are found.
  - **Not Available:** The state where the found book is not available for issue.
  - **Search Failure:** The final state when the search fails to find any available books.
- **Transitions:**
  - From **Input** to **Search:** When the user enters the search criteria.
  - From **Search** to **Check Database:** The system starts checking the database.
  - From **Check Database** to **Found:** If a book is found.
  - From **Check Database** to **Not Found:** If no book is found.
  - From **Found** to **Check Availability:** The system checks the availability of the found book.
  - From **Check Availability** to **Available:** If the book is available.
  - From **Check Availability** to **Not Available:** If the book is not available.
  - From **Available** to **Display:** The system displays the book details to the user.

- From **Not Found** or **Not Available** to **Search Failure:** The search process ends with a failure.

The statechart diagram provides a clear and concise overview of the book searching and issuing processes within the Library Management System, highlighting the different states and transitions involved in these operations.

Use Case Diagram

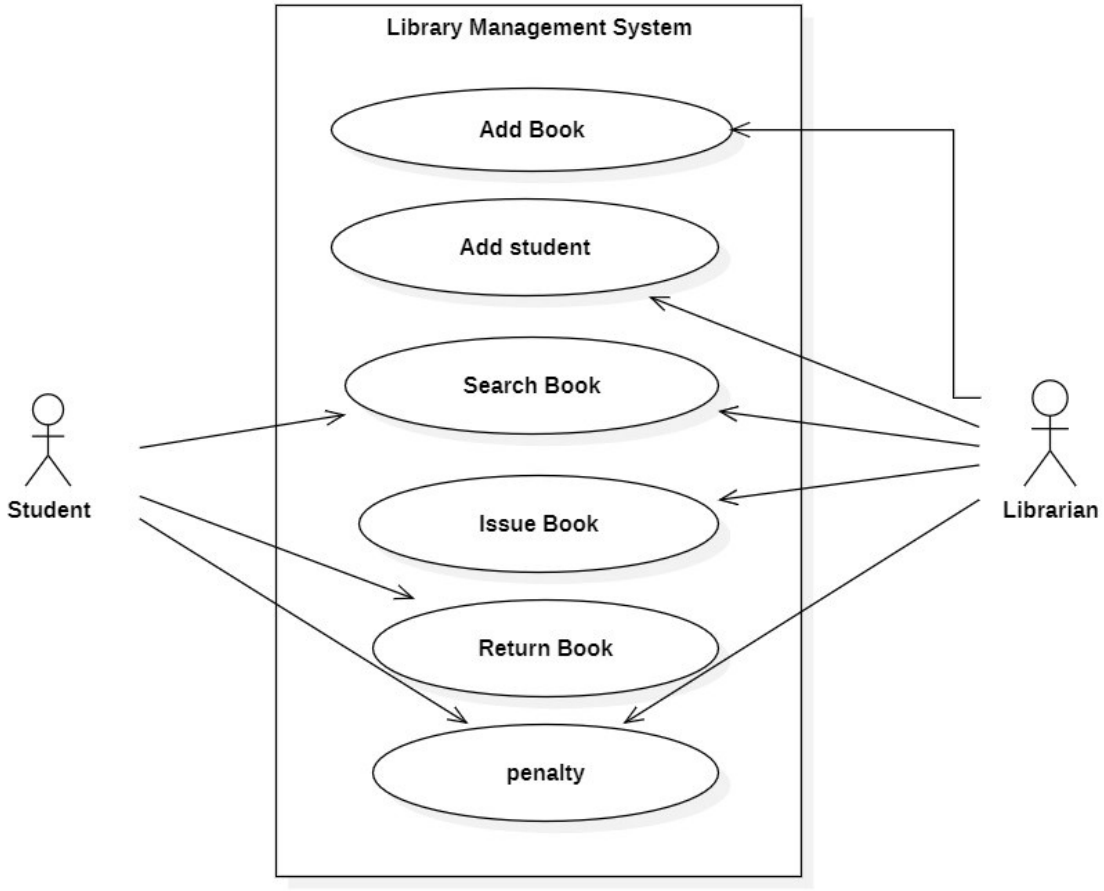


Fig 3.7

The use case diagram illustrates the various functionalities and interactions within a complex Library Management System. It highlights the different use cases available to both Library Users and Librarians, as well as the relationships between these use cases.

Actors:

- **Library User:** Represents the end-user of the library system, who interacts with the system for various purposes.
- **Librarian:** Represents the library staff responsible for managing the library system and its functionalities.

Use Cases:

- **Book Search:** Enables users to search for books in the library's catalog.
- **Book Reservation:** Allows users to reserve books that are currently checked out.
- **Book Checkout:** Handles the process of checking out books to library users.
- **Book Return:** Handles the process of returning borrowed books.

- **Fine Management:** Manages the imposition and collection of fines for overdue books.
- **User Registration:** Enables new users to register for library membership.

Relationships:

- **Include:** The "Book Reservation" use case includes the "Book Search" use case, indicating that book reservation involves searching for available books.
- **Include:** The "Book Checkout" use case includes the "Fine Management" use case, indicating that fines may be assessed during the checkout process.
- **Include:** The "Book Return" use case includes the "Fine Management" use case, indicating that fines may be assessed during the return process.
- **Include:** The "User Registration" use case includes the "Membership Renewal" use case, indicating that membership renewal is part of the user registration process.
- **Extend:** The "Book Recommendation" use case extends the "Book Search" use case, indicating that book recommendations can be provided as an additional feature of the book search functionality.

The use case diagram provides a comprehensive overview of the library management system's functionalities, highlighting the interactions between different actors and use cases, as well as the relationships between various components of the system.

Sequence Diagram

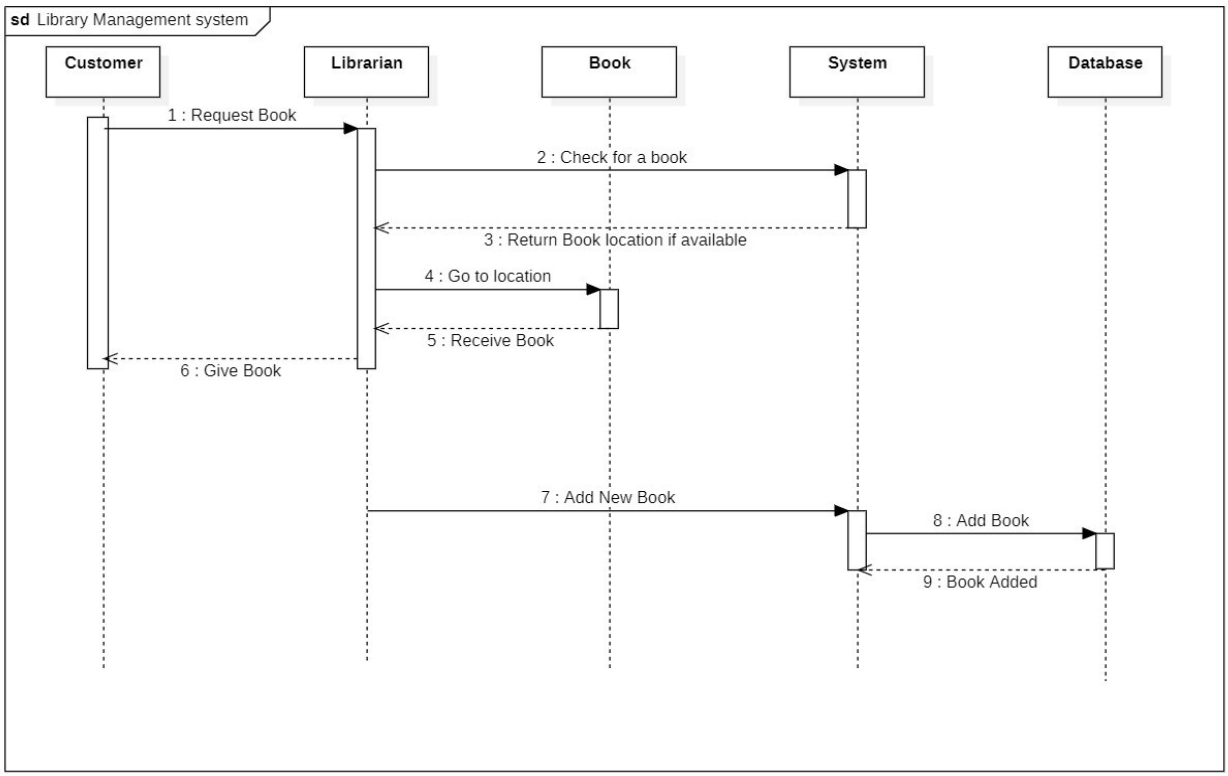


Fig 3.8

The sequence diagram illustrates the flow of interactions involved in a library management system, specifically focusing on the process of requesting and receiving a book. It outlines the sequence of messages exchanged between the Customer, Librarian, Book system, System, and Database.

Actors:

- **Customer:** Represents the library user who requests a book.
- **Librarian:** Represents the library staff who assists the customer.
- **Book System:** Represents the system used to manage book information and availability.
- **System:** Represents the overall library management system.
- **Database:** Represents the central repository for storing book and user information.

Sequence of Events:

1. **Request Book:** The Customer requests a specific book from the Librarian.
2. **Check for a book:** The Librarian checks the Book System for the availability of the requested book.
3. **Return Book location if available:** The Book System checks the Database for the location of the book and returns the information to the Librarian.

4. **Go to location:** The Librarian retrieves the book from the specified location.
5. **Receive Book:** The Librarian gives the requested book to the Customer.
6. **Give Book:** The Customer receives the book.
7. **Add New Book:** A new book is added to the library's collection.
8. **Add Book:** The System records the new book in the Database.
9. **Book Added:** The Database confirms the addition of the new book.

The sequence diagram provides a clear and concise overview of the steps involved in requesting and receiving a book from the library, highlighting the interactions between the different actors and the flow of information throughout the process.

Activity Diagram

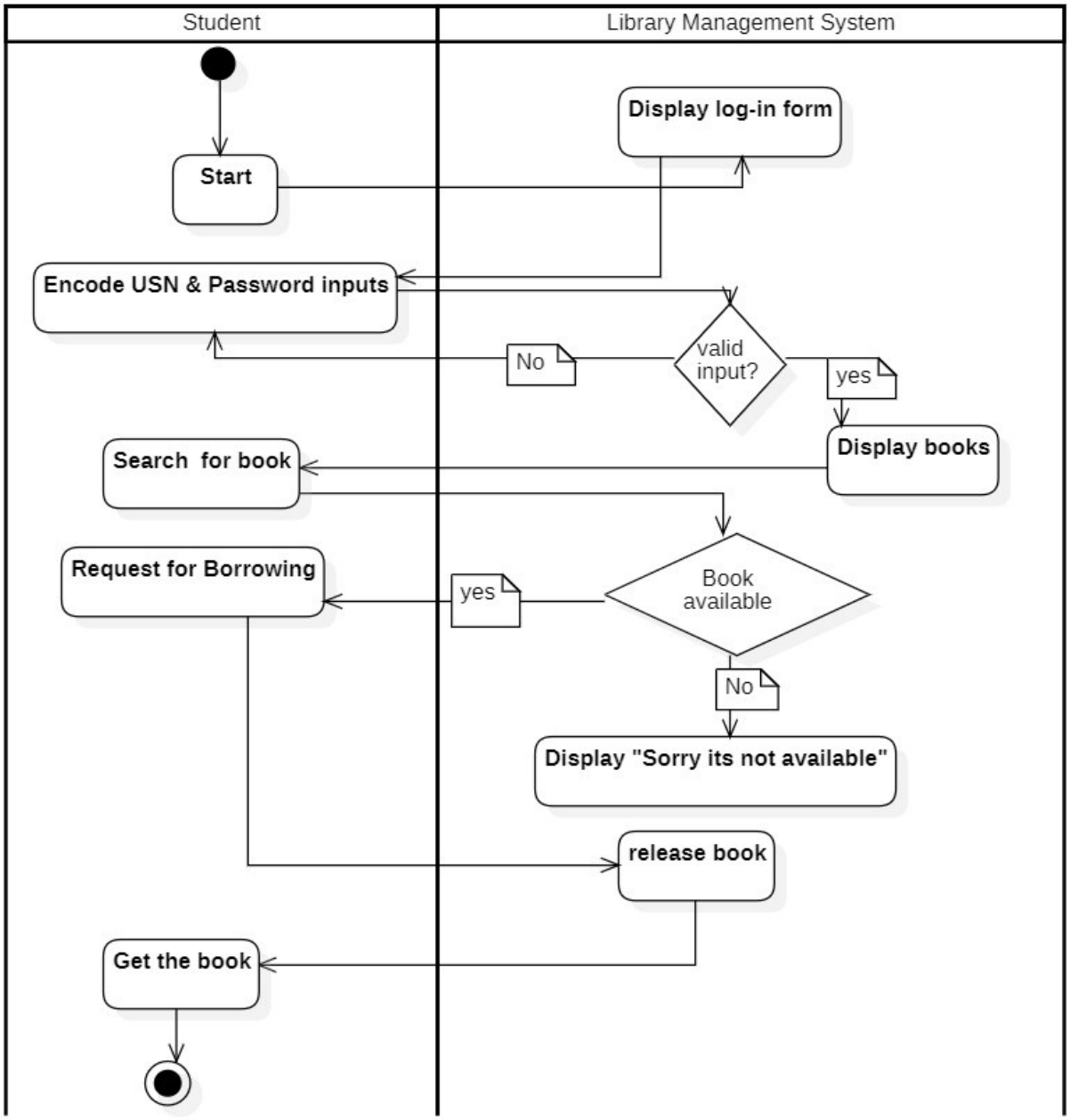


Fig 3.9

The activity diagram illustrates the process flow for a student borrowing a book from a Library Management System. It outlines the steps involved from the student's perspective and the system's response.

Actors:

- **Student:** The individual who is borrowing the book.
- **Library Management System:** The system that handles the book borrowing process.

Activities:

- **Start:** The process begins when the student initiates the book borrowing process.
- **Display log-in form:** The Library Management System displays a log-in form for the student to enter their credentials.
- **Encode USN & Password inputs:** The student enters their University Seat Number (USN) and password.
- **Valid input?:** The system validates the entered USN and password.
    - **No:** If the input is invalid, the system may display an error message or prompt the student to re-enter their credentials.
    - **Yes:** If the input is valid, the system proceeds to the next step.
- **Display books:** The system displays a list of available books to the student.
- **Search for book:** The student searches for the desired book from the list.
- **Book available?:** The system checks if the selected book is available for borrowing.
    - **No:** If the book is not available, the system displays a message like "Sorry, it's not available."
    - **Yes:** If the book is available, the system proceeds to the next step.
- **Request for Borrowing:** The student requests to borrow the selected book.
- **Get the book:** The system processes the request and allows the student to borrow the book.

The activity diagram provides a clear and concise overview of the book borrowing process, highlighting the key steps involved and the interactions between the student and the Library Management System.

---

4. Stock Maintenance System

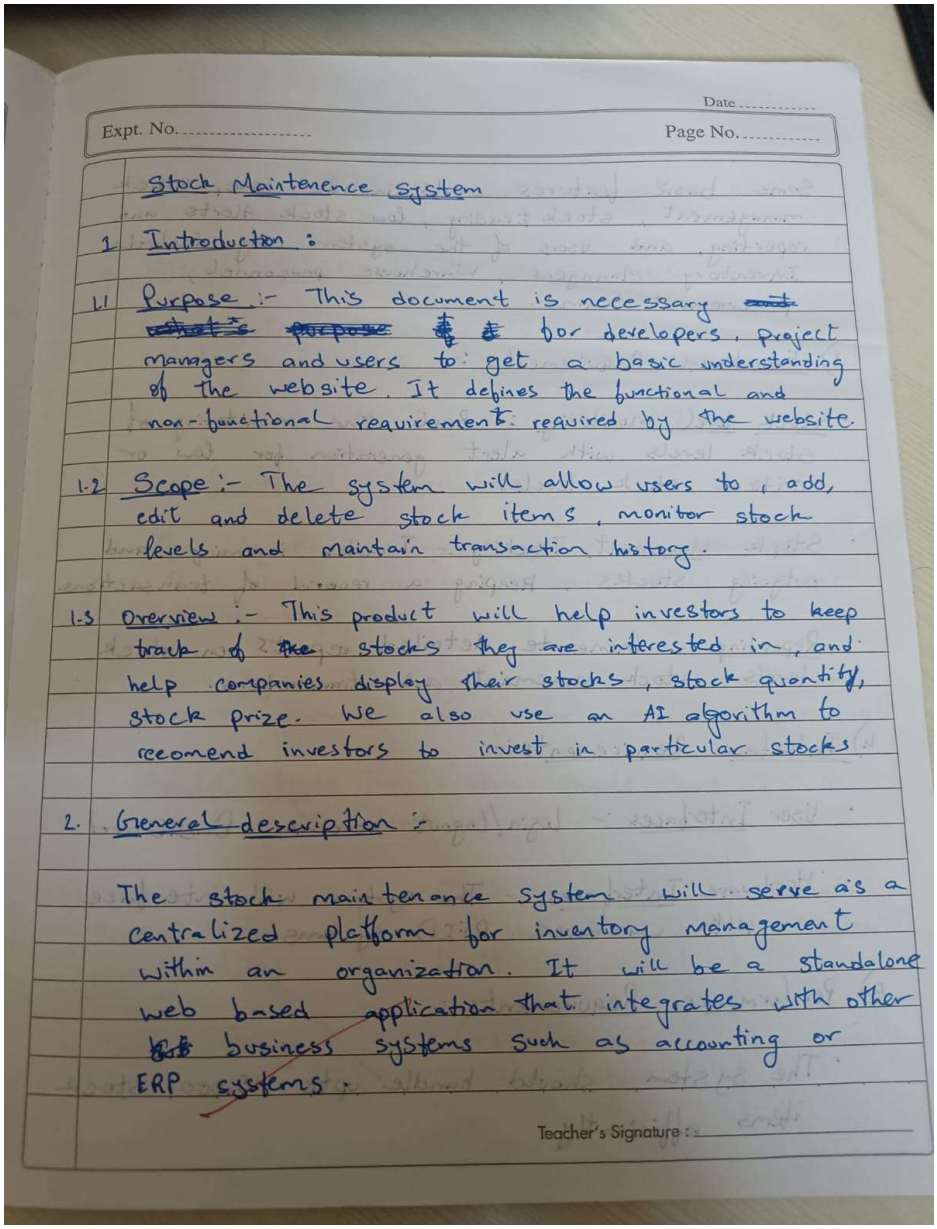
Problem Statement:

Design a software system to support a computerized stock maintenance network for a large manufacturing organization with multiple warehouses operating under a single brand. Each warehouse maintains its own local inventory management system to track stock levels, manage incoming and outgoing shipments, and generate reports. Warehouse staff at individual locations interact directly with their local systems to record stock movements, update inventory levels, and generate reports.

A central stock management system communicates with a central server that coordinates stock information across all warehouses, ensures real-time updates for stock availability, and manages stock replenishment orders. The central system allows for online order placement, order tracking, and inventory analysis.

The system requires robust recordkeeping, scalability to support multiple warehouses, and strict security measures to protect inventory data and prevent stock discrepancies. It must handle concurrent inventory updates and avoid stock-outs. Each warehouse will provide and maintain its software for local inventory management operations, while you are tasked with designing the software for the central stock management system, the online ordering platform, and the network connecting individual warehouses to the central server. The cost of the shared system will be apportioned to warehouses based on the volume of stock movements they process.

Software Requirement Specification Document



Date .....

Expt. No. ..... Page No. .....

Stock Maintenence System

1 Introduction :

1.1 Purpose :- This document is necessary ~~and what's purpose~~ for developers, project managers and users to get a basic understanding of the website. It defines the functional and non-functional requirement required by the website.

1.2 Scope :- The system will allow users to, add, edit and delete stock items, monitor stock levels and maintain transaction history.

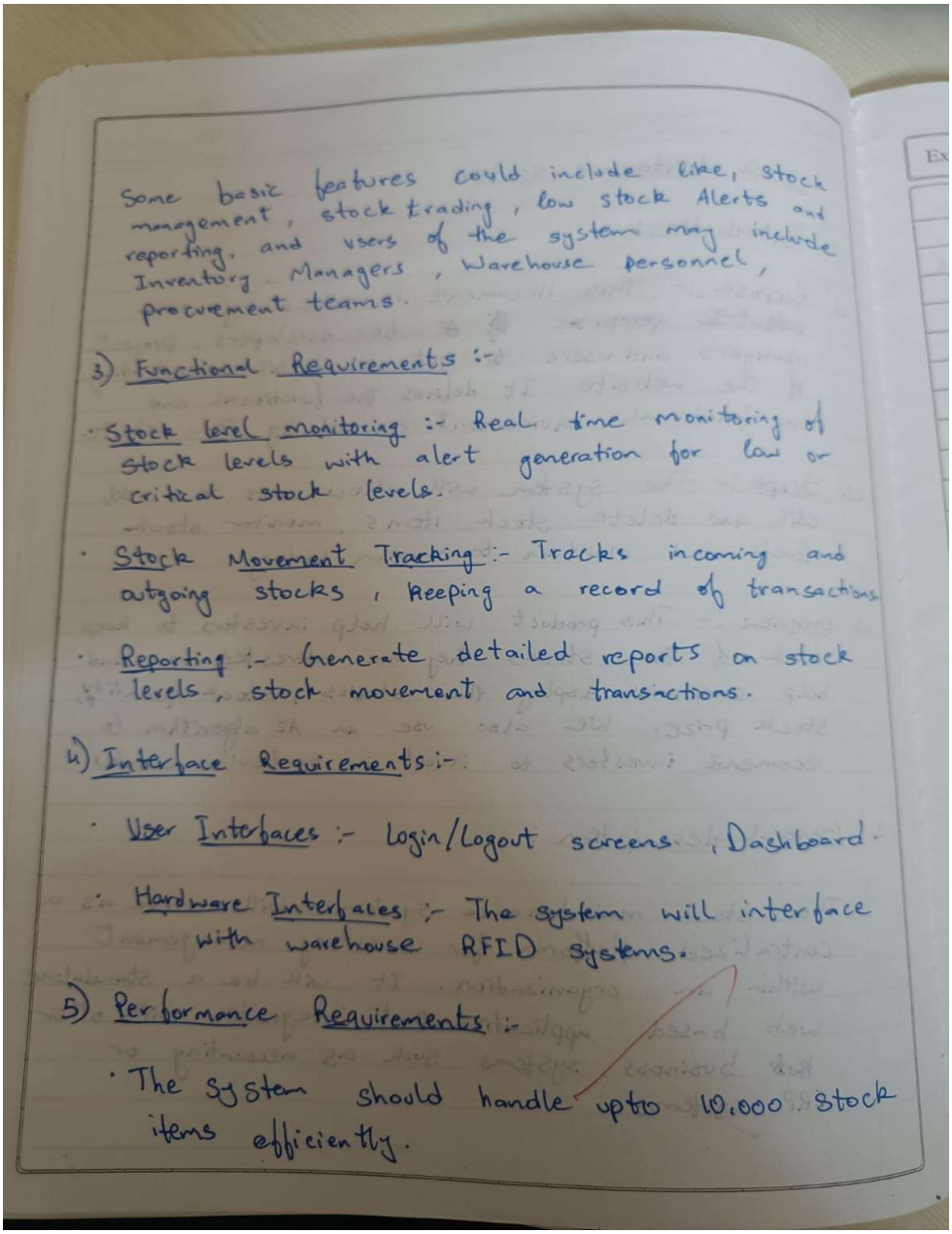
1.3 Overview :- This product will help investors to keep track of ~~the~~ stocks they ~~are~~ interested in, and help companies display their stocks, stock quantity, stock prize. We also use an AI algorithm to recomend investors to invest in particular stocks

2. General description :

The stock maintenance system will serve as a centralized platform for inventory management within an organization. It will be a standalone web based application that integrates with other ~~bus~~ business systems such as accounting or ERP systems.

Teacher's Signature : _____

Fig 4.1



Some basic features could include like, stock management, stock trading, low stock Alerts and reporting, and users of the system may include Inventory Managers, Warehouse personnel, procurement teams.

3) Functional Requirements :-

- Stock level monitoring :- Real time monitoring of stock levels with alert generation for low or critical stock levels.
- Stock Movement Tracking :- Tracks incoming and outgoing stocks, keeping a record of transactions.
- Reporting :- Generate detailed reports on stock levels, stock movement and transactions.

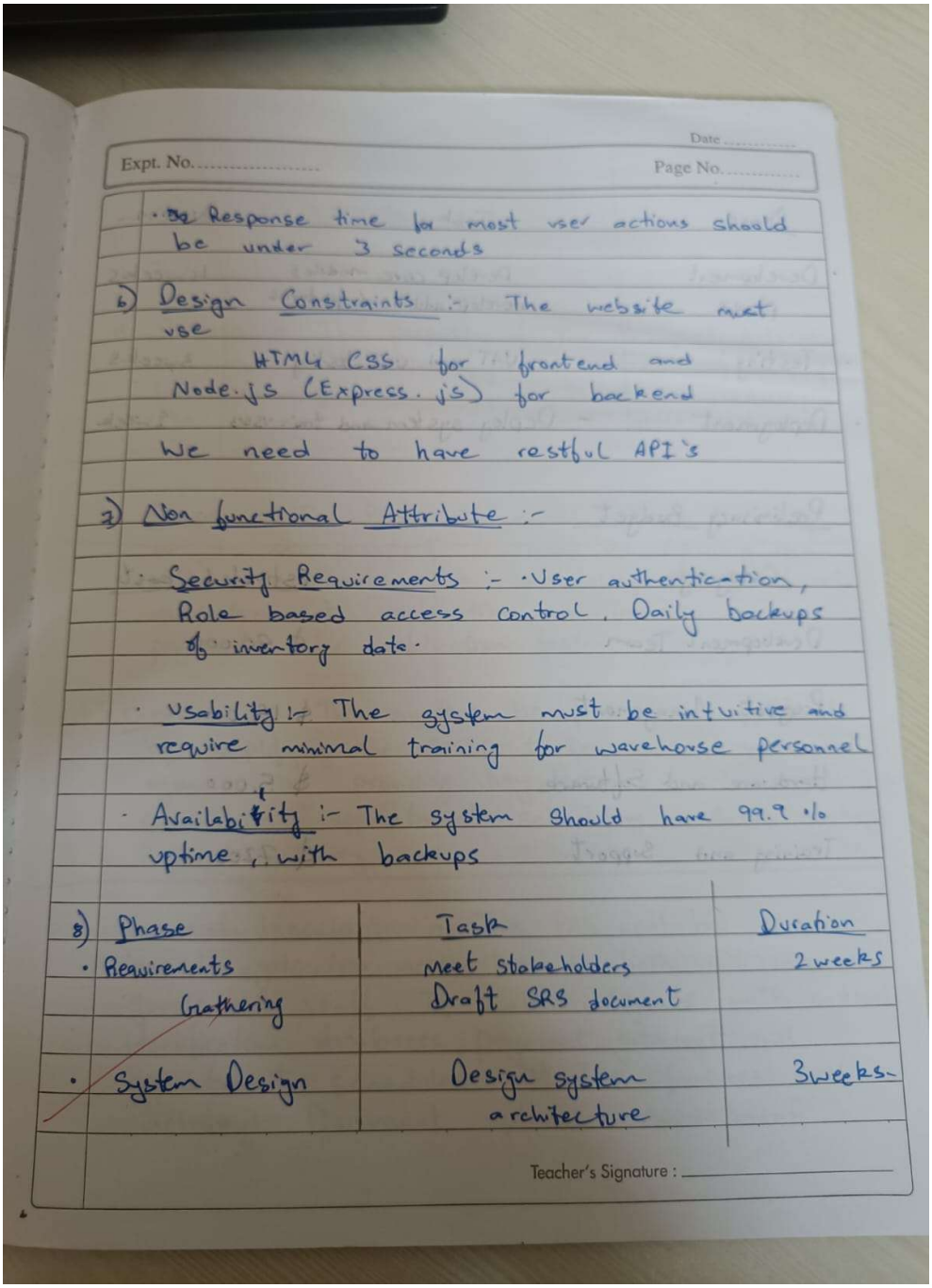
4) Interface Requirements :-

- User Interfaces :- Login/Logout screens, Dashboard.
- Hardware Interfaces :- The system will interface with warehouse RFID systems.

5) Performance Requirements :-

- The system should handle upto 10,000 stock items efficiently.

Fig 4.2



Expt. No. Date
Page No.

- Response time for most user actions should be under 3 seconds

6) Design Constraints :- The website must use
HTML, CSS for frontend and Node.js (Express.js) for backend
We need to have restful API's

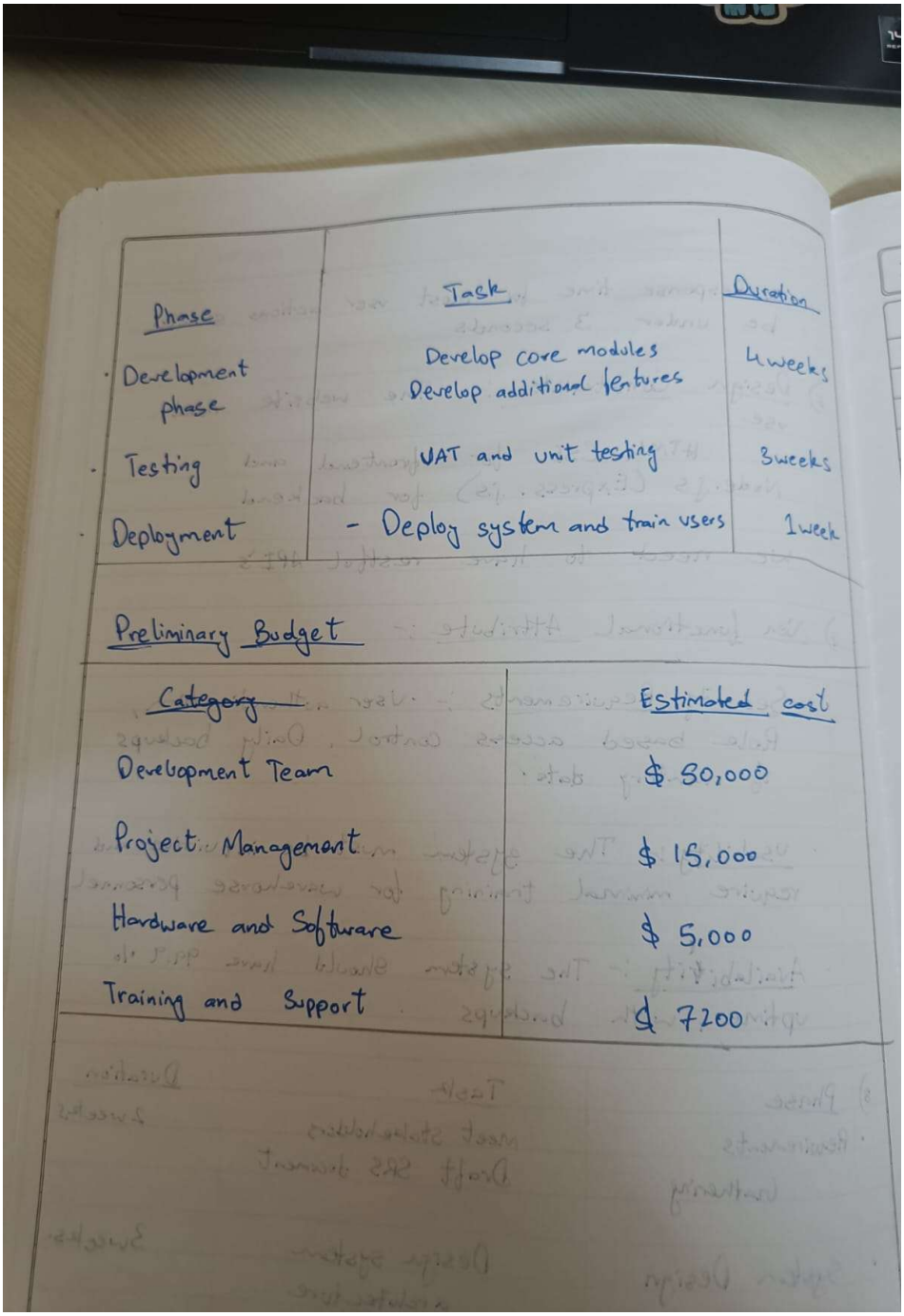
7) Non functional Attribute :-

- Security Requirements :- User authentication, Role based access control, Daily backups of inventory data.
- Usability :- The system must be intuitive and require minimal training for warehouse personnel
- Availability :- The system should have 99.9% uptime, with backups

8) Phase	Task	Duration
• Requirements Gathering	Meet stakeholders Draft SRS document	2 weeks
• System Design	Design system architecture	3 weeks

Teacher's Signature :

Fig 4.3



Phase	Task	Duration
Development phase	Develop core modules Develop additional features	4 weeks
Testing	UAT and unit testing	3 weeks
Deployment	- Deploy system and train users	1 week

Preliminary Budget

Category	Estimated cost
Development Team	$ 30,000
Project Management	$ 15,000
Hardware and Software	$ 5,000
Training and Support	$ 7200

Fig 4.4

Class Diagram

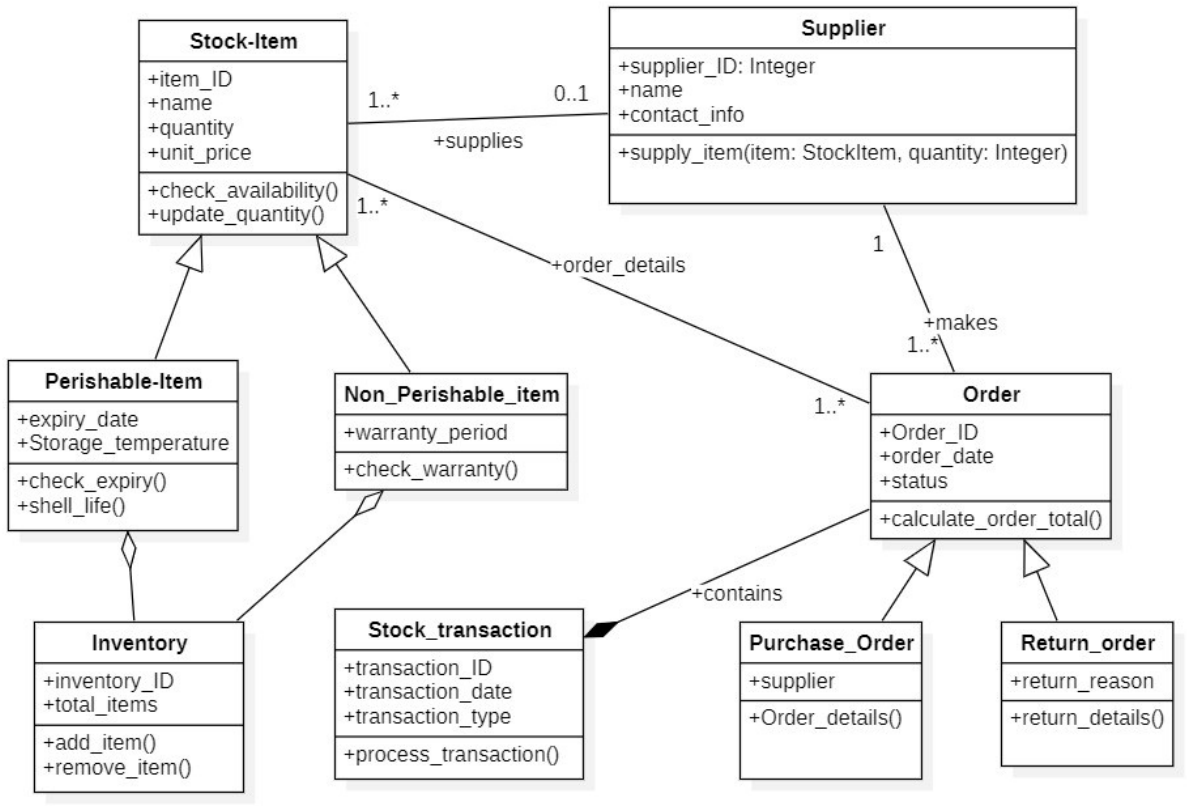


Fig 4.5

The class diagram illustrates the structure and relationships within a retail management system, highlighting key entities, their attributes, and the interactions between them.

Key Classes

- **Stock-Item:** Represents items stored in the inventory, including their ID, name, quantity, and price.
- **Perishable-Item:** Models perishable stock items with attributes like expiry date and storage temperature.
- **Non-Perishable-Item:** Represents non-perishable goods with a warranty period.
- **Supplier:** Captures details of suppliers who provide stock items.
- **Order:** Records details of stock transactions, including purchase and return orders.
- **Purchase_Order**: Represents orders made to suppliers for acquiring stock items.
- **Return_Order**: Tracks returned items and their associated reasons.
- **Inventory**: Manages the total stock, allowing items to be added or removed.
- **Stock_Transaction:** Logs and processes transactions involving stock items.

Relationships

- **Stock-Item - Supplier**: A supplier can supply multiple stock items, but each stock item can have at most one supplier.
- **Supplier - Order**: A supplier can make multiple orders, but each order is associated with one supplier.
- **Order - Purchase_Order/Return_Order**: Orders are further classified into purchase orders and return orders based on their purpose.
- **Order - Stock-Item:** An order can contain multiple stock items, and each stock item can be part of multiple orders.
- **Inventory - Stock-Item**: The inventory manages stock items by adding or removing them.
- **Stock_Transaction - Stock-Item:** Stock transactions involve processing changes in the quantity of stock items.Methods

Each class has methods for creating, modifying, deleting, and searching for objects within the class. For example:

The **Stock-Item** class includes the methods check_availability(), which verifies if an item is in stock, and update_quantity(), which adjusts the quantity of the item. The **Perishable-Item** class adds the methods check_expiry(), to determine if the item has expired, and shell_life(), to calculate the remaining usable duration. Similarly, the **Non-Perishable-Item** class introduces the method check_warranty() to verify the validity of an item's warranty.

The class diagram provides a clear and concise representation of the entities and their relationships within the retail management system, laying the foundation for the design and implementation of the software.

State Diagram

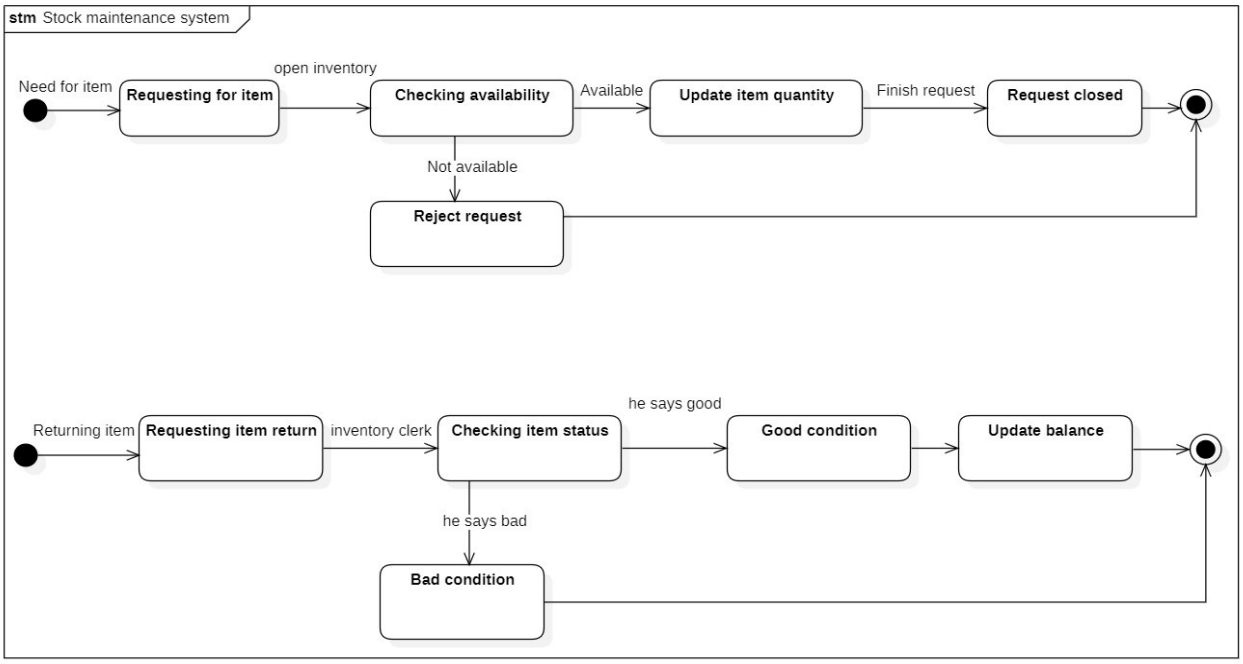


Fig 4.6

The statechart diagram illustrates the workflow for managing stock items within a Stock Maintenance System. It outlines the states and transitions involved in both requesting and returning items.

States:

- **Need for Item:** The initial state where a request for an item is made.
- **Requesting for Item:** The state where the request for an item is being processed.
- **Checking Availability:** The state where the system checks the availability of the requested item.
- **Available:** The state where the requested item is available in stock.
- **Update Item Quantity:** The state where the system updates the item quantity after the item is issued or returned.
- **Finish Request:** The state where the request is successfully completed.
- **Request Closed:** The final state after the request is processed.
- **Reject Request:** The state where the request for an item is rejected due to unavailability.
- **Returning Item:** The state where a request for returning an item is made.
- **Requesting Item Return:** The state where the request for returning an item is being processed.

- **Checking Item Status:** The state where the condition of the returned item is being checked.
- **Good Condition:** The state where the returned item is in good condition.
- **Bad Condition:** The state where the returned item is in bad condition.
- **Update Balance:** The state where the stock balance is updated after the item is returned.

Transitions:

- From "Need for Item" to "Requesting for Item": When a request for an item is made.
- From "Requesting for Item" to "Checking Availability": After the request is received.
- From "Checking Availability" to "Available": If the item is available in stock.
- From "Checking Availability" to "Not Available": If the item is not available in stock.
- From "Available" to "Update Item Quantity": If the item is available, the quantity is updated.
- From "Update Item Quantity" to "Finish Request": After the quantity is updated.
- From "Finish Request" to "Request Closed": Upon successful completion of the request.
- From "Checking Availability" to "Reject Request": If the item is not available.
- From "Returning Item" to "Requesting Item Return": When a request for returning an item is made.
- From "Requesting Item Return" to "Checking Item Status": After the return request is received.
- From "Checking Item Status" to "Good Condition": If the returned item is in good condition.
- From "Checking Item Status" to "Bad Condition": If the returned item is in bad condition.
- From "Good Condition" to "Update Balance": If the item is in good condition, the stock balance is updated.
- From "Update Balance" to the final state: After the stock balance is updated.

The statechart diagram provides a clear and concise overview of the stock management process, highlighting the different stages involved in requesting and returning items, as well as the possible outcomes and transitions between states.

Use Case Diagram

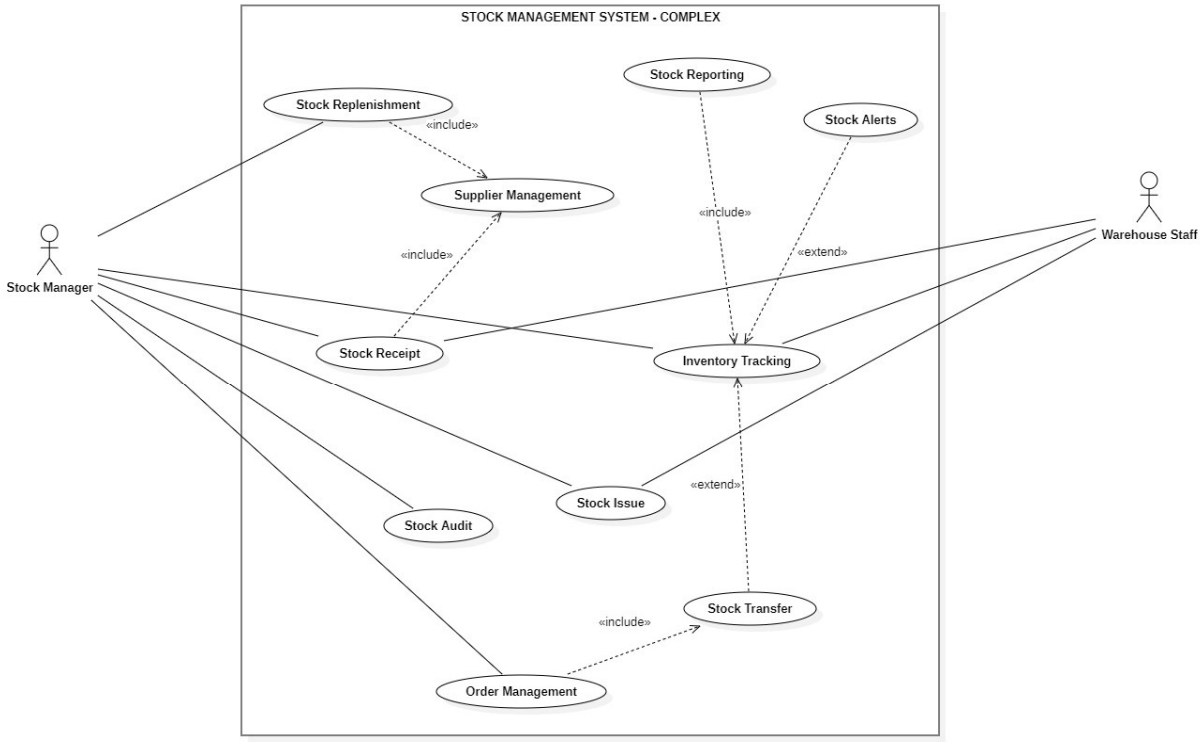


Fig 4.7

The use case diagram illustrates the various functionalities and interactions within a complex Stock Management System. It highlights the different use cases available to both Stock Managers and Warehouse Staff, as well as the relationships between these use cases.

Actors:

- **Stock Manager:** Represents the individual responsible for managing the stock in the system.
- **Warehouse Staff:** Represents the staff members working in the warehouse who are involved in various stock-related activities.

Use Cases:

- **Stock Replenishment:** Handles the process of ordering and receiving new stock items.
- **Stock Reporting:** Generates reports on stock levels, inventory trends, and other relevant metrics.
- **Stock Alerts:** Issues alerts for low stock levels, potential stockouts, and other critical inventory situations.
- **Supplier Management:** Manages relationships with suppliers, including adding, modifying, and deleting supplier information.

- **Stock Receipt:** Handles the process of receiving incoming stock shipments.
- **Inventory Tracking:** Tracks the movement of stock items within the warehouse, including receiving, issuing, and transferring stock.
- **Stock Issue:** Handles the process of issuing stock items from the warehouse, such as fulfilling orders or transferring stock to other locations.
- **Stock Audit:** Performs regular audits of stock levels to ensure accuracy and identify discrepancies.
- **Stock Transfer:** Handles the transfer of stock items between different locations within the warehouse or across different warehouses.
- **Order Management:** Manages orders for stock items, including order placement, processing, and fulfillment.

Relationships:

- **Include:** "Stock Replenishment" includes "Supplier Management," indicating that supplier management is a necessary part of the stock replenishment process.
- **Include:** "Stock Reporting" includes "Inventory Tracking," indicating that inventory tracking data is essential for generating stock reports.
- **Include:** "Stock Receipt" includes "Supplier Management," indicating that supplier information is required for receiving stock shipments.
- **Include:** "Stock Issue" includes "Inventory Tracking," indicating that stock issue transactions affect inventory levels.
- **Include:** "Stock Audit" includes "Inventory Tracking," as audits are performed on the tracked inventory data.
- **Include:** "Order Management" includes "Inventory Tracking," as order fulfillment relies on accurate inventory information.
- **Extend:** "Stock Alerts" extends "Inventory Tracking," indicating that stock alerts are generated based on changes in inventory levels.
- **Extend:** "Stock Issue" extends "Inventory Tracking," indicating that stock issuance affects inventory levels.
- **Extend:** "Stock Transfer" extends "Inventory Tracking," indicating that stock transfers affect inventory levels.

The use case diagram provides a comprehensive overview of the stock management system's functionalities, highlighting the interactions between different actors and use cases, as well as the relationships between various components of the system.

Sequence Diagram

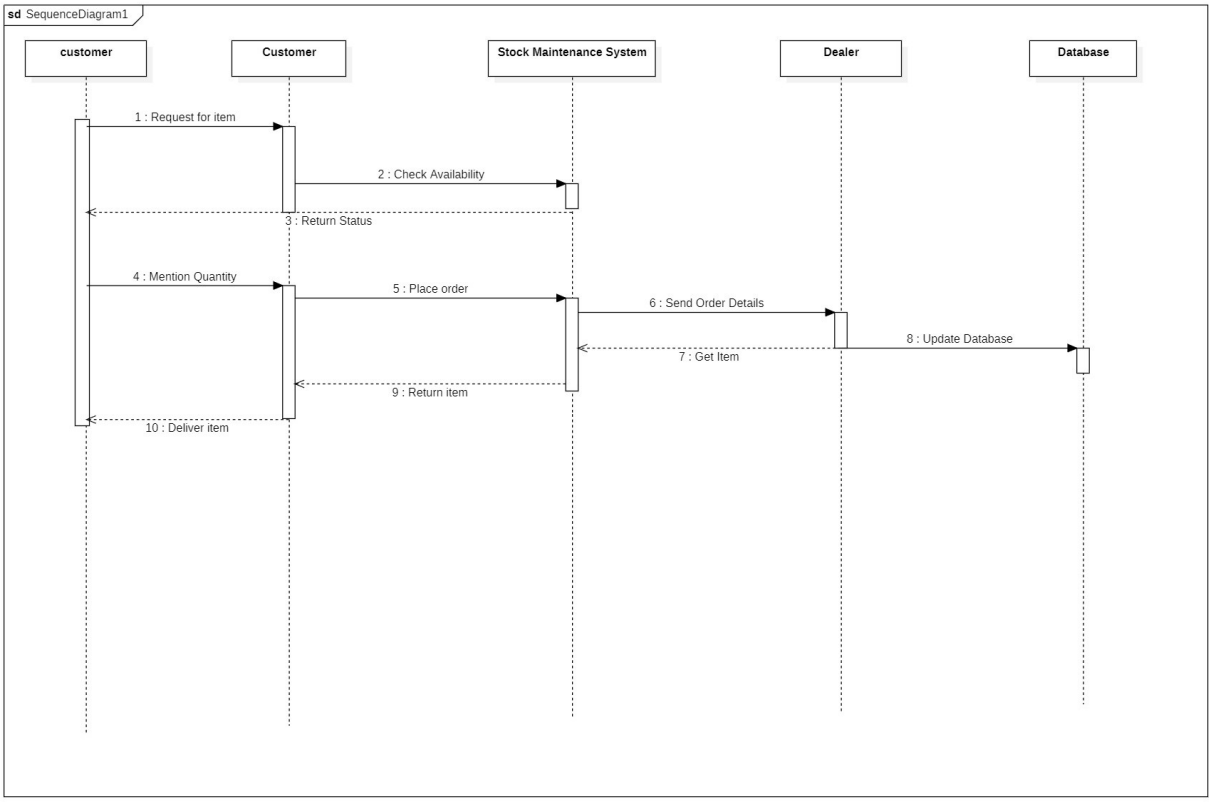


Fig 4.8

The sequence diagram illustrates the flow of interactions involved in a stock maintenance system, specifically focusing on the process of ordering an item. It outlines the sequence of messages exchanged between the Customer, Stock Maintenance System, Dealer, and Database.

Actors:

- **Customer:** Represents the individual who is placing an order.
- **Stock Maintenance System:** Represents the system used to manage stock information and orders.
- **Dealer:** Represents the entity responsible for fulfilling the order.
- **Database:** Represents the central repository for storing stock and order information.

Sequence of Events:

1. **Request for Item:** The Customer requests a specific item from the Stock Maintenance System.
2. **Check Availability:** The Stock Maintenance System checks the database for the availability of the requested item.

3. **Return Status:** The Stock Maintenance System sends the availability status (available or unavailable) back to the Customer.
4. **Mention Quantity:** The Customer specifies the desired quantity of the item.
5. **Place order:** The Customer places an order for the specified quantity of the item.
6. **Send Order Details:** The Stock Maintenance System sends the order details to the Dealer.
7. **Get Item:** The Dealer retrieves the ordered item from the stock.
8. **Update Database:** The Dealer updates the database to reflect the reduction in stock quantity.
9. **Return Item:** The Dealer delivers the ordered item to the Customer.
10. **Deliver Item:** The Customer receives the ordered item.

The sequence diagram provides a clear and concise overview of the steps involved in placing and fulfilling an order within the stock maintenance system, highlighting the interactions between the different actors and the flow of information throughout the process.

Activity Diagram

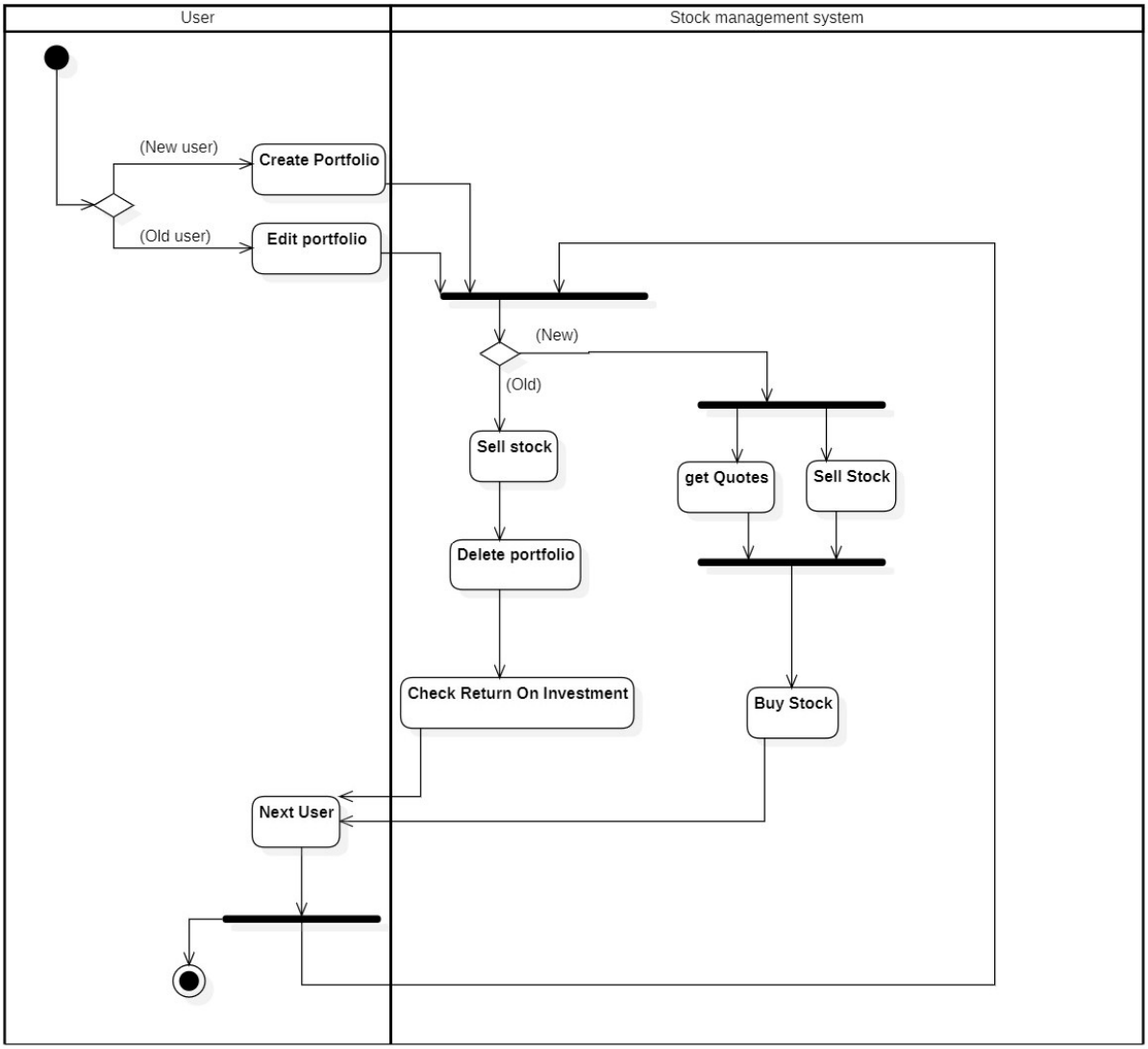


Fig 4.9

The activity diagram illustrates the workflow for a user interacting with a Stock Management System. It outlines the various actions a user can perform within the system.

Actors:

- **User:** Represents the individual interacting with the Stock Management System.

Activities:

- **Create Portfolio:** This activity allows new users to create a portfolio to track their investments.
- **Edit Portfolio:** Existing users can edit their existing portfolios by adding, removing, or modifying investments.

- **Sell Stock:** This activity enables users to sell stocks from their portfolio.
- **Get Quotes:** The system retrieves real-time stock quotes to assist users in making informed decisions.
- **Sell Stock:** The system processes the user's sell order.
- **Buy Stock:** This activity allows users to buy new stocks to add to their portfolio.
- **Delete Portfolio:** Users can delete their portfolio if they no longer wish to track it.
- **Check Return On Investment:** This activity calculates and displays the return on investment for the user's portfolio.
- **Next User:** The system transitions to the next user to begin their interaction with the system.

Conditions:

- The diagram includes conditional branches to differentiate between "New User" and "Old User" actions.

The activity diagram provides a clear and concise overview of the user interactions within the Stock Management System, highlighting the different actions available to users and the flow of the system.

---

5. Passport Automation System

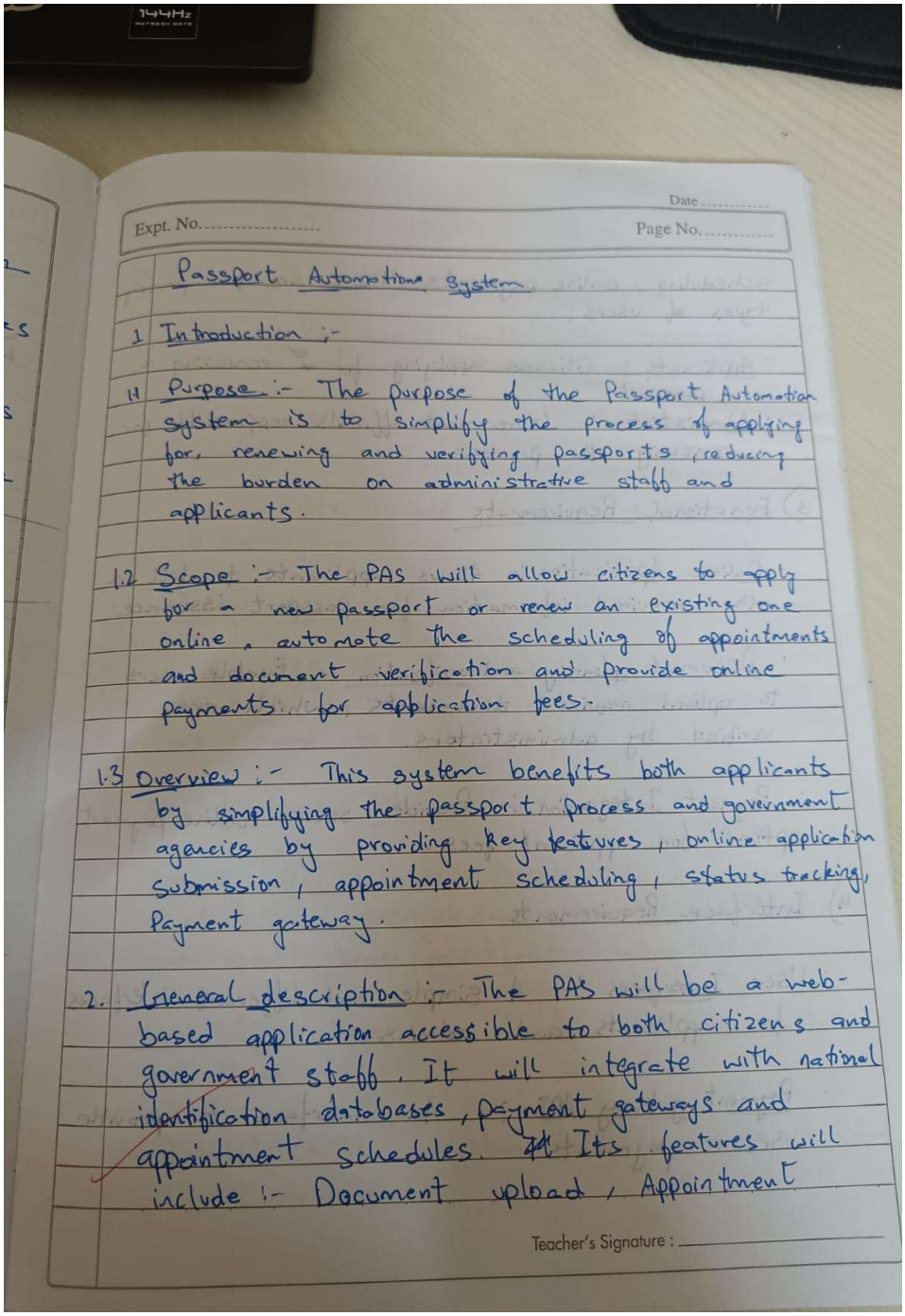
Problem Statement:

Design a software system to support a computerized Passport Automation System for a government agency. The system should streamline the passport application process, reducing manual intervention and ensuring efficiency. Passport offices maintain their own local computer systems to handle application intake, document verification, and initial data entry. Passport officers interact directly with their local systems to capture applicant information, verify documents, and update application status.

A central passport processing system communicates with a central server that coordinates application processing, manages document verification, and tracks passport issuance across all offices. The central system allows applicants to track their application status online, submit supporting documents electronically, and make online payments for passport fees.

The system requires robust data security, scalability to handle a large volume of applications, and strict adherence to government regulations and data privacy standards. It must handle concurrent applications, ensure data integrity, and prevent fraudulent activities. Each passport office will provide and maintain its software for local operations, while you are tasked with designing the software for the central passport processing system, the online applicant portal, and the network connecting individual passport offices to the central server. The cost of the shared system will be managed by the government agency.

Software Requirement Specification Document



Expt. No. ..... Date .....
Page No. .....

Passport Automation System

1 Introduction :-

1.1 Purpose :- The purpose of the Passport Automation system is to simplify the process of applying for, renewing and verifying passports, reducing the burden on administrative staff and applicants.

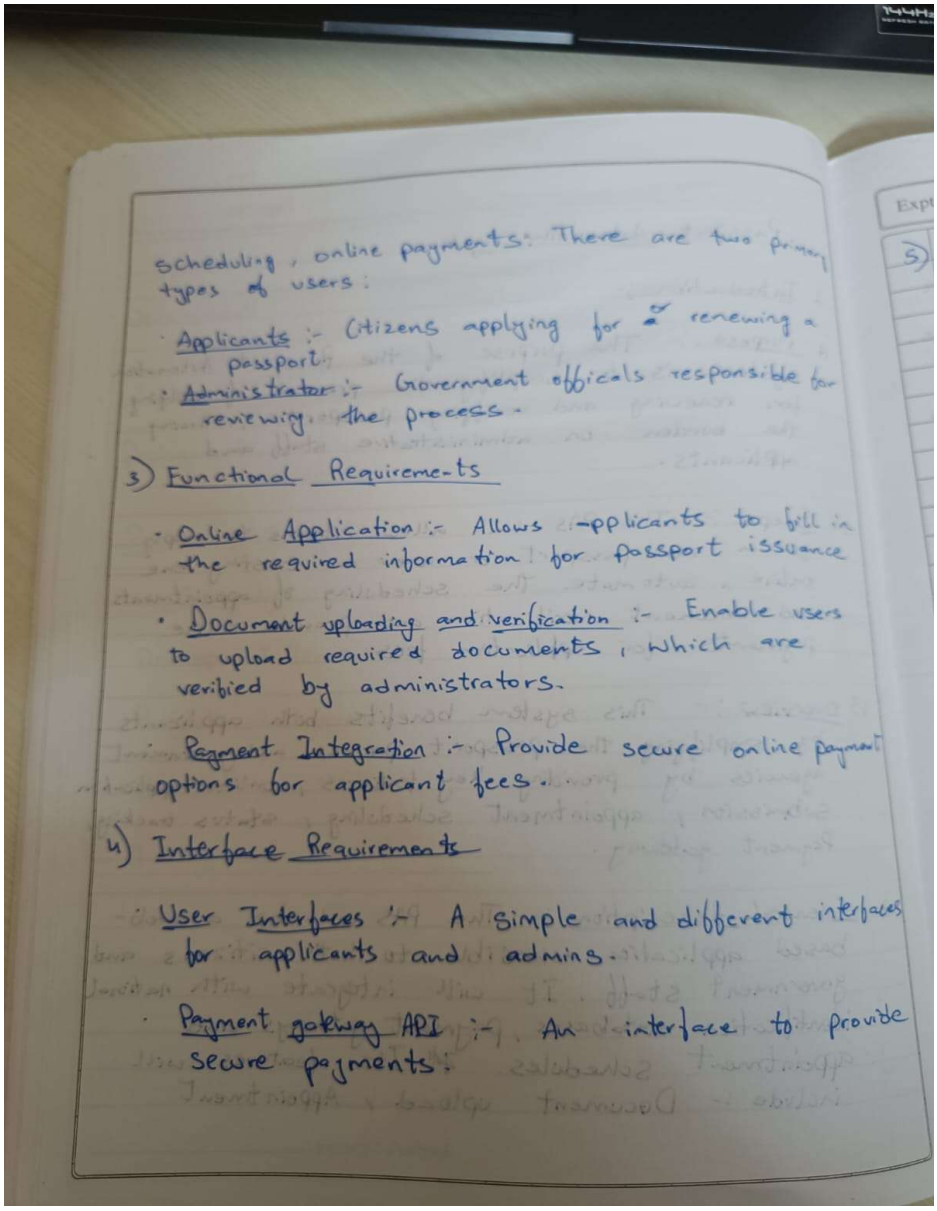
1.2 Scope :- The PAS will allow citizens to apply for a new passport or renew an existing one online, automate the scheduling of appointments and document verification and provide online payments for application fees.

1.3 Overview :- This system benefits both applicants by simplifying the passport process and government agencies by providing key features, online application submission, appointment scheduling, status tracking, Payment gateway.

2. General description :- The PAS will be a web-based application accessible to both citizens and government staff. It will integrate with national identification databases, payment gateways and appointment schedules. Its features will include :- Document upload, Appointment

Teacher's Signature : _____

Fig 5.1



scheduling, online payments. There are two primary types of users:

- Applicants :- Citizens applying for or renewing a passport.
- Administrator :- Government officials responsible for reviewing the process.

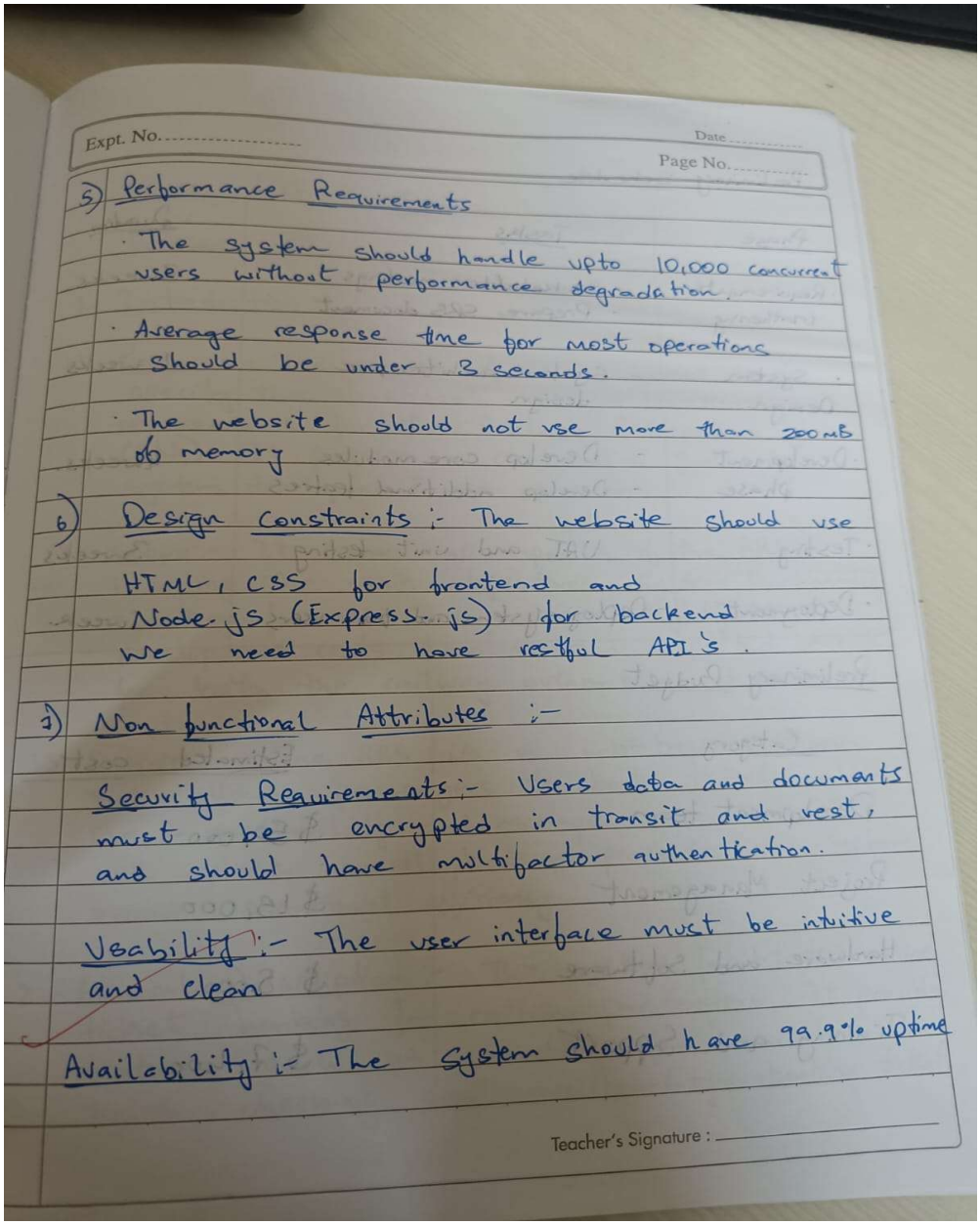
3) Functional Requirements

- Online Application :- Allows applicants to fill in the required information for passport issuance.
- Document uploading and verification :- Enable users to upload required documents, which are verified by administrators.
- Payment Integration :- Provide secure online payment options for applicant fees.

4) Interface Requirements

- User Interfaces :- A simple and different interfaces for applicants and admins.
- Payment gateway API :- An interface to provide secure payments.

Fig 5.2



Expt. No..... Date.....
Page No.....

5) Performance Requirements

- The system should handle upto 10,000 concurrent users without performance degradation.
- Average response time for most operations should be under 3 seconds.
- The website should not use more than 200 MB of memory

6) Design Constraints :- The website should use HTML, CSS for frontend and Node.js (Express.js) for backend we need to have restful API's.

7) Non functional Attributes :-

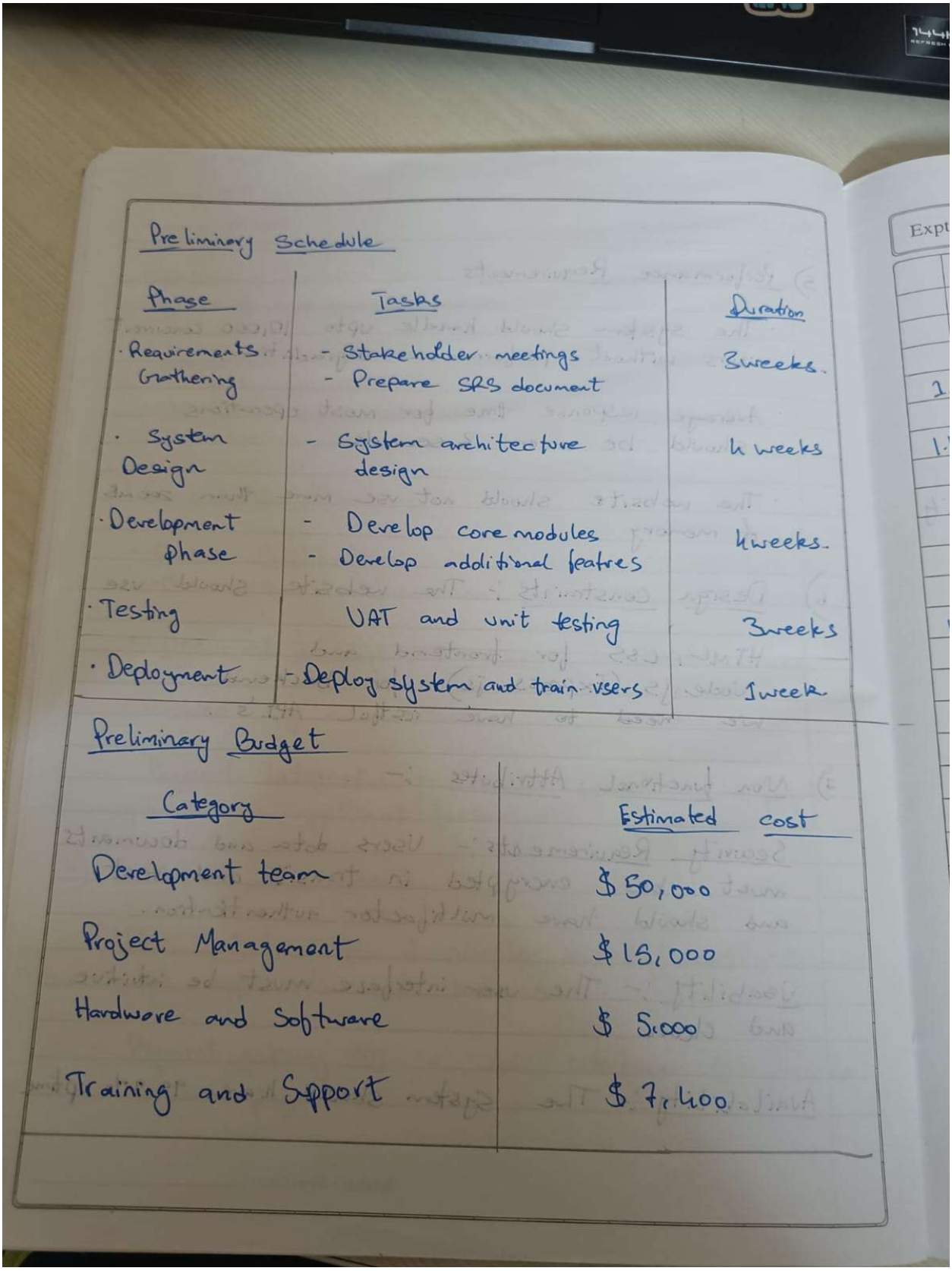
Security Requirements :- Users data and documents must be encrypted in transit and rest, and should have multifactor authentication.

Usability :- The user interface must be intuitive and clean

Availability :- The system should have 99.9% uptime

Teacher's Signature :

Fig 5.3



Preliminary Schedule

Phase	Tasks	Duration
· Requirements Gathering	- Stakeholder meetings - Prepare SRS document	3 weeks.
· System Design	- System architecture design	4 weeks
· Development phase	- Develop core modules - Develop additional features	4 weeks.
· Testing	UAT and unit testing	3 weeks
· Deployment	- Deploy system and train users	1 week

Preliminary Budget

Category	Estimated cost
Development team	$ 50,000
Project Management	$ 15,000
Hardware and Software	$ 5,000
Training and Support	$ 7,400

Fig 5.4

Class Diagram

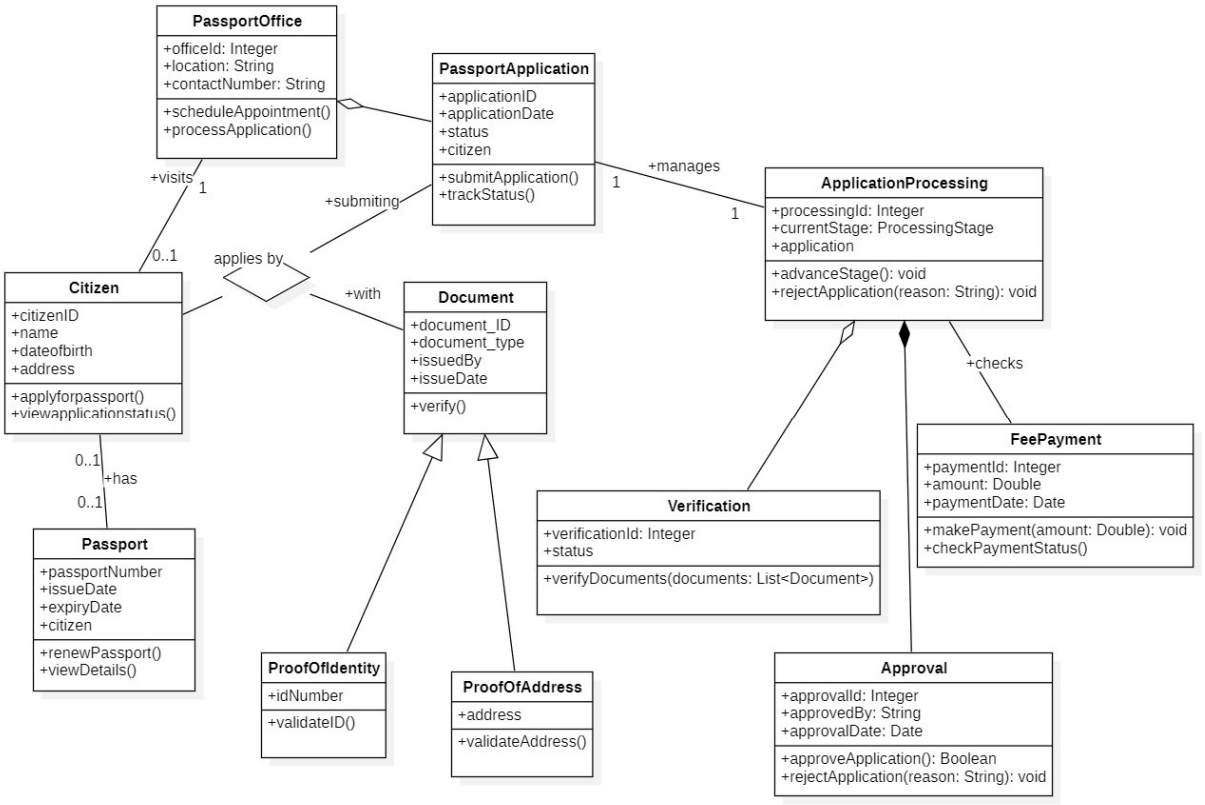


Fig 5.5

The class diagram illustrates the structure and relationships within a Passport Automation System. It highlights the key classes, their attributes, and the associations between them.

Classes:

- **PassportOffice**: Manages passport-related operations, including scheduling appointments and processing applications through its methods.
- **PassportApplication**: Represents a citizen's application for a passport, allowing submission and tracking of application status.
- **Citizen**: Represents an individual applying for a passport, with methods to apply for a passport and check application status.

- **Document**: Represents supporting documents for passport applications, with a method to verify their authenticity.
- **ProofOfIdentity**: A specialized document type representing identity proof, including a method to validate the ID.
- **ProofOfAddress**: A specialized document type representing address proof, including a method to validate the address.
- **Passport**: Represents the issued passport with attributes such as passport number, issue date, and expiry date, and methods to renew and view passport details.
- **ApplicationProcessing**: Handles the processing of passport applications, with methods to advance or reject the application at different stages.
- **Verification**: Represents the verification process for submitted documents, with a method to verify a list of documents.
- **FeePayment**: Represents the payment process for the application fee, with methods to make payments and check payment status.
- **Approval**: Manages the final approval or rejection of passport applications, with methods to approve or reject the application.

Relationships:

- **PassportOffice - Citizen**: A citizen can visit a passport office to apply for a passport.
- **Citizen - PassportApplication**: A citizen submits a passport application to apply for a passport. **PassportApplication - ApplicationProcessing**: Each application is managed by a specific processing instance.
- **PassportApplication - Document**: A passport application requires supporting documents submitted by the citizen.
- **Document - ProofOfIdentity/ProofOfAddress**: Proof of identity and proof of address are specialized types of documents.
- **ApplicationProcessing - Verification**: The application processing stage involves verification of submitted documents.
- **ApplicationProcessing - FeePayment**: Fee payment is checked during the application processing stage.

- **ApplicationProcessing - Approval**: The application is either approved or rejected during the processing stage.
- **Citizen - Passport**: A citizen may be issued a passport upon successful application approval.

Methods:

Each class would likely have methods for interacting with its attributes and related classes. For example:

The **PassportOffice** class includes methods such as scheduleAppointment() for booking appointments and processApplication() for handling passport applications. The **PassportApplication** class provides the methods submitApplication() for submitting a new application and trackStatus() for checking the current status of the application. The **Citizen** class has the methods applyForPassport() to initiate the application process and viewApplicationStatus() to monitor the progress of their passport application.

The **Document** class includes the method verify() to authenticate the validity of documents. Its specialized subclasses

The class diagram provides a clear and concise representation of the classes and their relationships within the Passport Automation System, laying the foundation for the object-oriented design and implementation of the system.

State Diagram

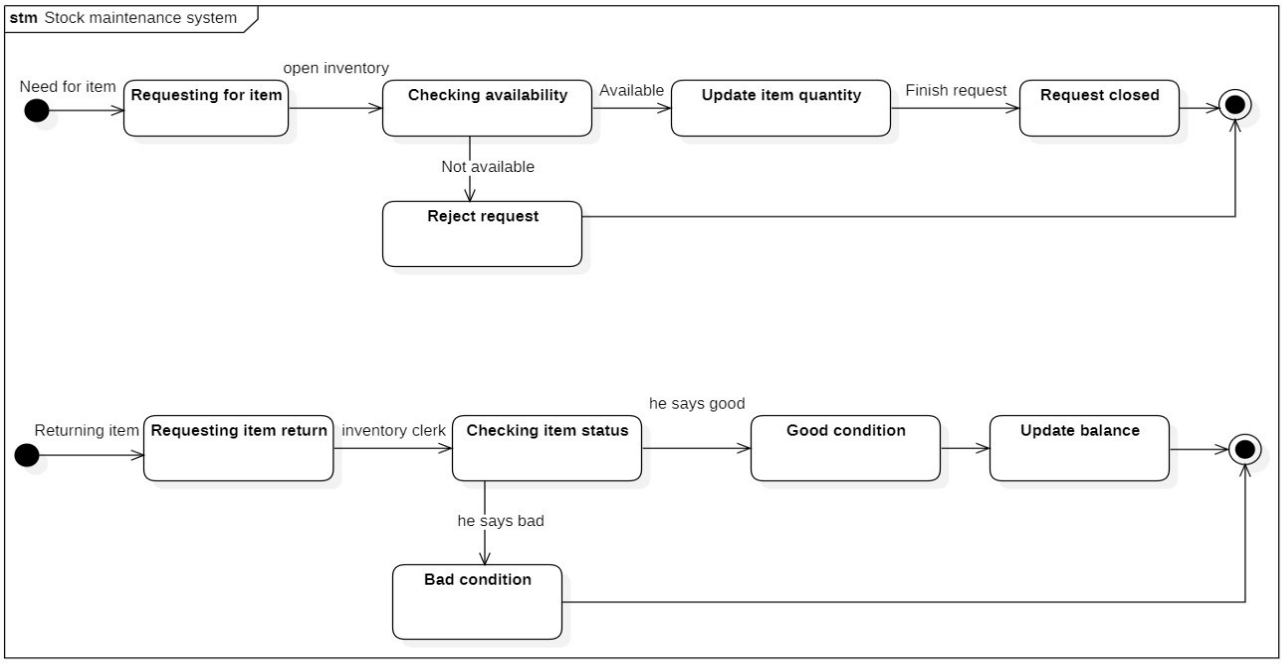


Fig 5.6

The statechart diagram illustrates the workflow for managing stock items within a Stock Maintenance System. It outlines the states and transitions involved in both requesting and returning items.

States:

- **Need for Item:** The initial state where a request for an item is made.
- **Requesting for Item:** The state where the request for an item is being processed.
- **Checking Availability:** The state where the system checks the availability of the requested item.
- **Available:** The state where the requested item is available in stock.
- **Update Item Quantity:** The state where the system updates the item quantity after the item is issued or returned.
- **Finish Request:** The state where the request is successfully completed.
- **Request Closed:** The final state after the request is processed.
- **Reject Request:** The state where the request for an item is rejected due to unavailability.
- **Returning Item:** The state where a request for returning an item is made.

- **Requesting Item Return:** The state where the request for returning an item is being processed.
- **Checking Item Status:** The state where the condition of the returned item is being checked.
- **Good Condition:** The state where the returned item is in good condition.
- **Bad Condition:** The state where the returned item is in bad condition.
- **Update Balance:** The state where the stock balance is updated after the item is returned.

Transitions:

- From "Need for Item" to "Requesting for Item": When a request for an item is made.
- From "Requesting for Item" to "Checking Availability": After the request is received.
- From "Checking Availability" to "Available": If the item is available in stock.
- From "Checking Availability" to "Not Available": If the item is not available in stock.
- From "Available" to "Update Item Quantity": If the item is available, the quantity is updated.
- From "Update Item Quantity" to "Finish Request": After the quantity is updated.
- From "Finish Request" to "Request Closed": Upon successful completion of the request.
- From "Checking Availability" to "Reject Request": If the item is not available.
- From "Returning Item" to "Requesting Item Return": When a request for returning an item is made.
- From "Requesting Item Return" to "Checking Item Status": After the return request is received.
- From "Checking Item Status" to "Good Condition": If the returned item is in good condition.
- From "Checking Item Status" to "Bad Condition": If the returned item is in bad condition.
- From "Good Condition" to "Update Balance": If the item is in good condition, the stock balance is updated.

The statechart diagram provides a clear and concise overview of the stock management process, highlighting the different stages involved in requesting and returning items, as well as the possible outcomes and transitions between states.

Use Case Diagram

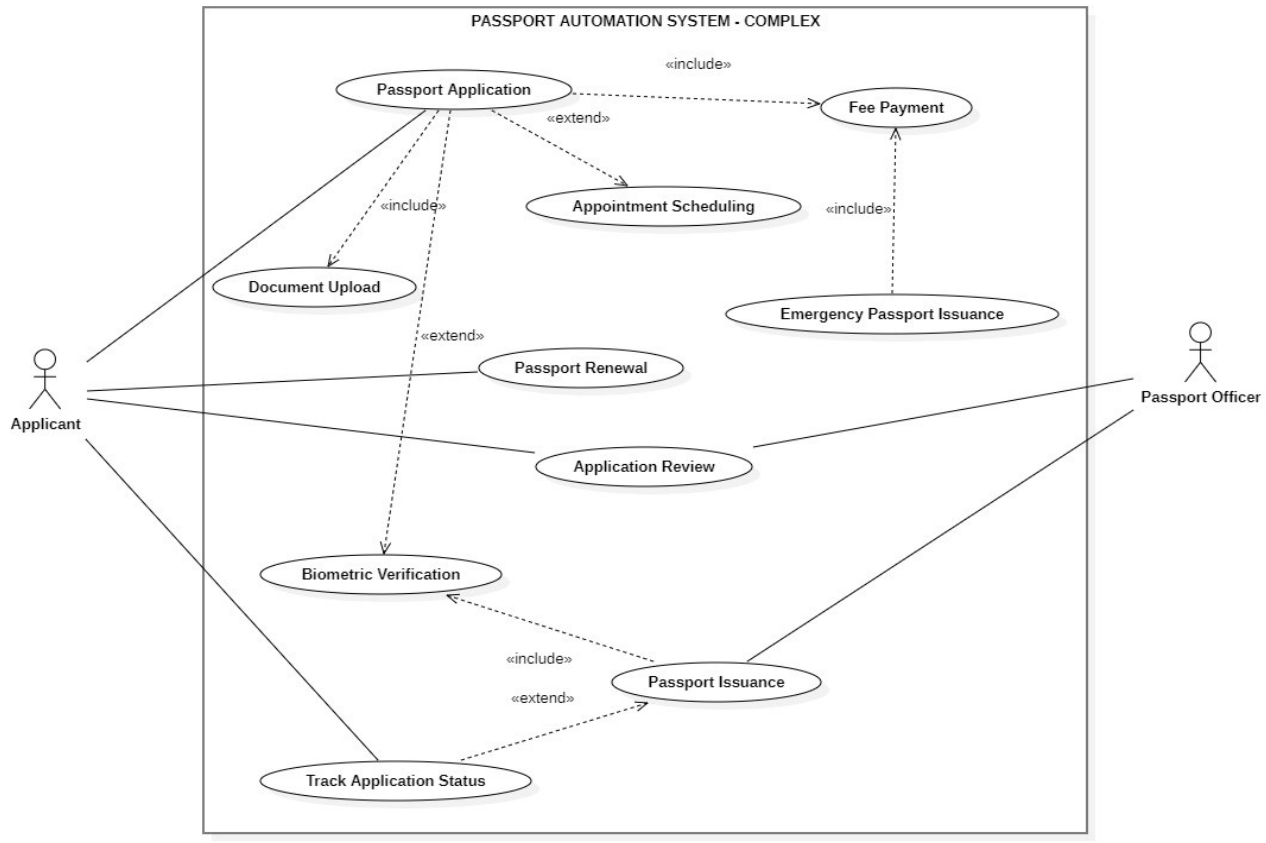


Fig 5.7

The use case diagram illustrates the various functionalities and interactions within a complex Passport Automation System. It highlights the different use cases available to both Applicants and Passport Officers, as well as the relationships between these use cases.

Actors:

- **Applicant:** Represents the individual applying for a passport.
- **Passport Officer:** Represents the officer responsible for processing passport applications.

Use Cases:

- **Passport Application:** The core process of applying for a passport, including submitting the application form and required documents.

- **Fee Payment:** The process of paying the required fees for the passport application.
- **Appointment Scheduling:** Scheduling an appointment for biometric verification and document submission.
- **Document Upload:** The process of uploading supporting documents electronically as part of the application.
- **Passport Renewal:** The process of renewing an existing passport.
- **Emergency Passport Issuance:** Handles emergency passport applications requiring expedited processing.
- **Application Review:** The process of reviewing the application and supporting documents.
- **Biometric Verification:** The process of capturing biometric data (fingerprints and photograph) of the applicant.
- **Passport Issuance:** The final stage of the process where the passport is issued to the applicant.
- **Track Application Status:** Allows applicants to track the status of their passport application online.

Relationships:

- **Include:** "Fee Payment" is included in "Passport Application," indicating that fee payment is an integral part of the application process.
- **Include:** "Appointment Scheduling" is included in "Passport Application," indicating that scheduling an appointment is part of the application process.
- **Include:** "Document Upload" is included in "Passport Application," indicating that document upload is part of the application process.
- **Include:** "Emergency Passport Issuance" includes "Fee Payment," indicating that fee payment is required for emergency passport applications.
- **Include:** "Biometric Verification" is included in "Passport Issuance," indicating that biometric verification is necessary before passport issuance.
- **Include:** "Track Application Status" is included in all other use cases, indicating that applicants can track the status throughout the process.
- **Extend:** "Passport Renewal" extends "Passport Application," indicating that passport renewal follows a similar process to a new passport application with some variations.

- **Extend:** "Emergency Passport Issuance" extends "Passport Application," indicating that emergency passport issuance follows the general application process with expedited procedures.

The use case diagram provides a comprehensive overview of the passport automation system's functionalities, highlighting the interactions between different actors and use cases, as well as the relationships between various components of the system.

Sequence Diagram

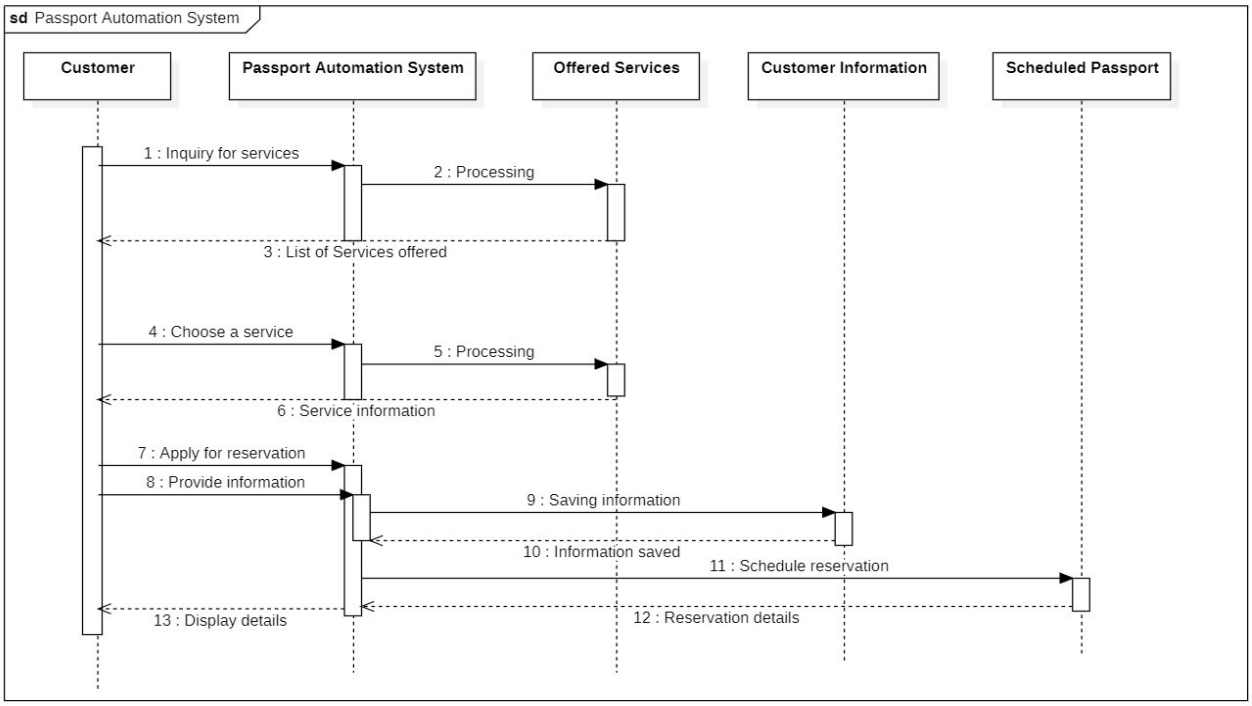


Fig 5.8

The sequence diagram illustrates the flow of interactions involved in a Passport Automation System, specifically focusing on the process of scheduling a passport service. It outlines the sequence of messages exchanged between the Customer, Passport Automation System, Offered Services, Customer Information, and Scheduled Passport components.

Actors:

- **Customer:** Represents the individual who is requesting a passport service.
- **Passport Automation System:** Represents the system that manages the passport application process.
- **Offered Services:** Represents the module responsible for displaying available passport services.
- **Customer Information:** Represents the module responsible for collecting and storing customer information.

- **Scheduled Passport:** Represents the module responsible for scheduling appointments and storing reservation details.

Sequence of Events:

1. **Inquiry for services:** The Customer initiates the process by inquiring about available passport services.
2. **Processing:** The Passport Automation System receives the inquiry and begins processing it.
3. **List of Services offered:** The Offered Services module provides a list of available passport services to the Passport Automation System.
4. **Choose a service:** The Customer selects a specific service from the list.
5. **Processing:** The Passport Automation System processes the service selection.
6. **Service information:** The Passport Automation System provides information about the selected service to the Customer.
7. **Apply for reservation:** The Customer applies for a reservation for the selected service.
8. **Provide information:** The Customer provides the necessary information for the reservation, such as personal details and appointment preferences.
9. **Saving information:** The Customer Information module receives and saves the customer's information.
10. **Information saved:** The Customer Information module confirms that the information has been saved.
11. **Schedule reservation:** The Scheduled Passport module schedules the reservation based on the customer's information and available slots.
12. **Reservation details:** The Scheduled Passport module provides the reservation details (date, time, location) to the Customer.
13. **Display details:** The Passport Automation System displays the reservation details to the Customer.

The sequence diagram provides a clear and concise overview of the steps involved in scheduling a passport service, highlighting the interactions between the different components of the system and the flow of information throughout the process.

Activity Diagram

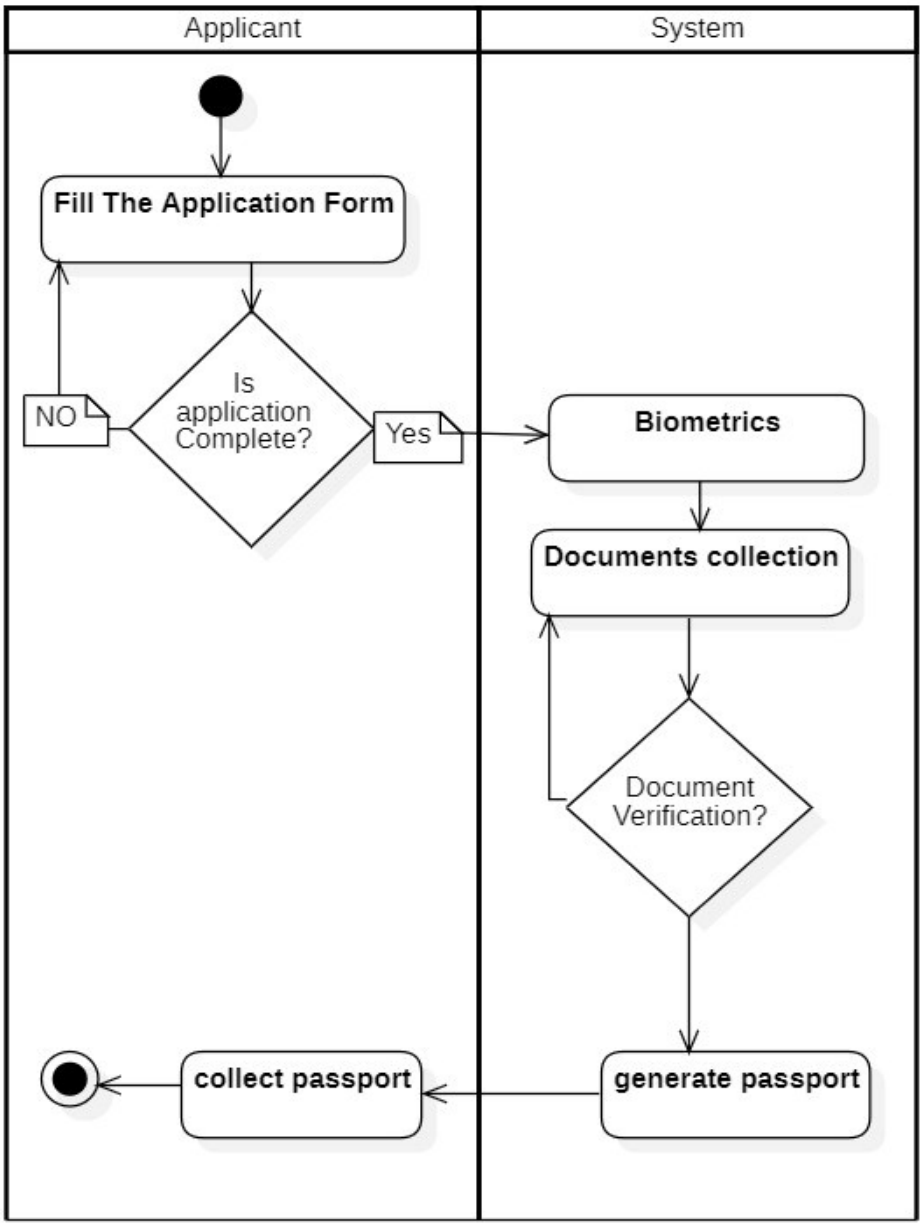


Fig 5.9

The activity diagram illustrates the workflow for processing a passport application. It outlines the steps involved from the applicant's perspective and the system's actions.

Actors:

- **Applicant:** The individual applying for the passport.
- **System:** Represents the passport processing system.

Activities:

- **Action1:** The initial state where the application process begins.
- **Fill The Application Form:** The applicant fills out the passport application form with the required information.
- **Is application Complete:** The system checks if the applicant has filled out the application form completely.
    - **No:** If the application is incomplete, the applicant is prompted to complete the form.
    - **Yes:** If the application is complete, the process moves to the next step.
- **Biometrics:** The applicant undergoes biometric data capture (e.g., fingerprints, photograph).
- **Documents collection:** The applicant submits the required supporting documents (e.g., proof of address, identification).
- **Document Verification:** The system verifies the authenticity of the submitted documents.
- **generate passport:** If all checks are successful, the system generates the passport.
- **collect passport:** The applicant collects the issued passport.

The activity diagram provides a clear and concise overview of the passport application process, highlighting the key steps involved and the interactions between the applicant and the system.
