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

Laboratory Record

OBJECT-ORIENTED MODELLING

Submitted in partial fulfilment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

M RAJASHEKAR REDDY

1BM22CS138

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
September 2024- January 2025

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 M Rajashekar Reddy (1BM22CS138) during the 5th Semester September 2024- January 2025.

Signature of the Faculty in Charge
Saritha A. N
Assistant Professor,
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

TABLE OF CONTENTS

CHAPTER - 1

HOTEL MANAGEMENT SYSTEM

PROBLEM STATEMENT

The Hotel Management System is a comprehensive software solution designed to streamline and automate the core operations of hotels, resorts, and lodging facilities. Managing hotel operations manually or through outdated systems often leads to booking conflicts, billing discrepancies, inefficient resource allocation, and poor customer experiences. This system addresses these challenges by integrating reservation management, housekeeping, billing, and customer service into a unified platform, improving operational efficiency and enhancing guest satisfaction.

- What is the system for?

The Hotel Management System is designed to automate and streamline various operations within a hotel. It helps manage room reservations, guest check-ins and check-outs, housekeeping schedules, billing, and customer feedback. By integrating these functions, the system ensures smooth operations and improves the guest experience.

- Where is it used?

This system is primarily used in hotels, resorts, inns, motels, and other accommodations offering hospitality services. It is also used by large hotel chains requiring centralized management across multiple locations.

- Who is it for?

The system is for hotel administrators, front desk staff, housekeeping teams, and guests. For administrators, it provides operational insights, while for staff, it reduces manual work. Guests benefit from faster service and a more pleasant experience.

- When is it required?

It is required during daily hotel operations, especially in high-demand periods like tourist seasons, holidays, or large-scale events. It is indispensable for hotels with a large volume of guests or multiple branches.

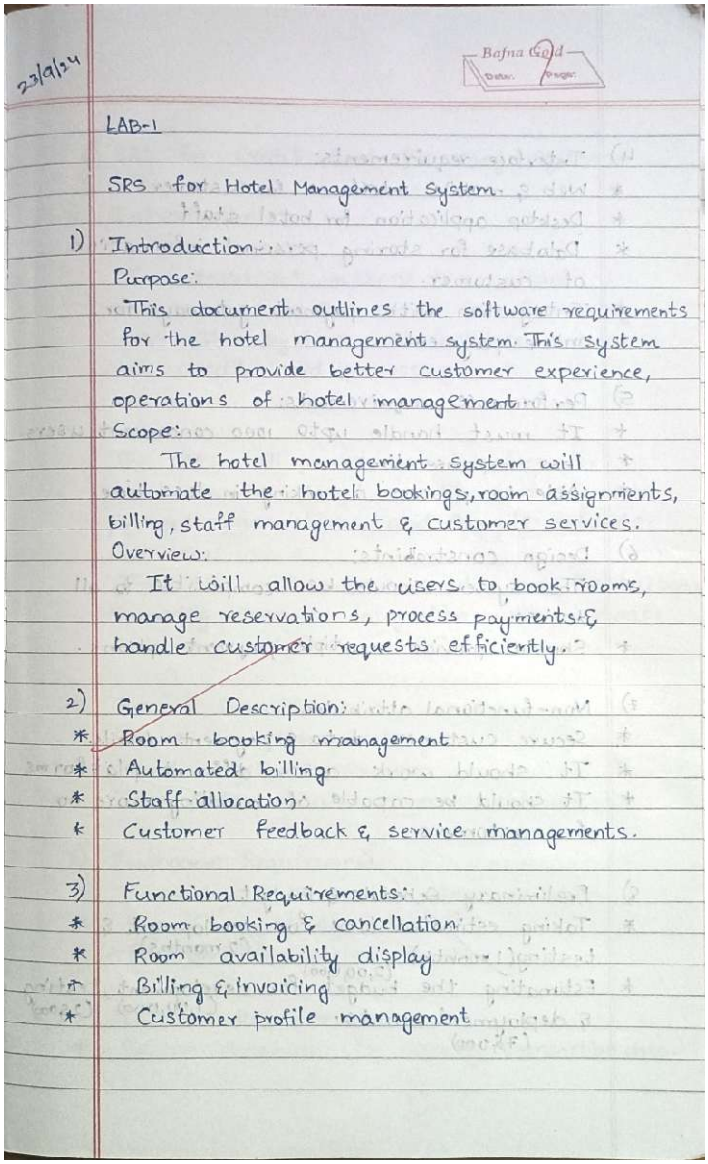
- Why is it needed?

The system is needed to enhance efficiency, reduce human errors, and optimize resource utilization. It helps hotels improve customer satisfaction by offering personalized services and faster processing. Additionally, it supports financial management by generating detailed reports.

- How is it done?

The system is implemented through specialized software that integrates hotel operations into a centralized platform. This platform includes modules for reservations, room allocation, housekeeping management, customer billing, and performance analytics. Cloud-based solutions allow access from multiple locations, enabling better management for chains.

SOFTWARE REQUIREMENTS SPECIFICATION



23/9/24

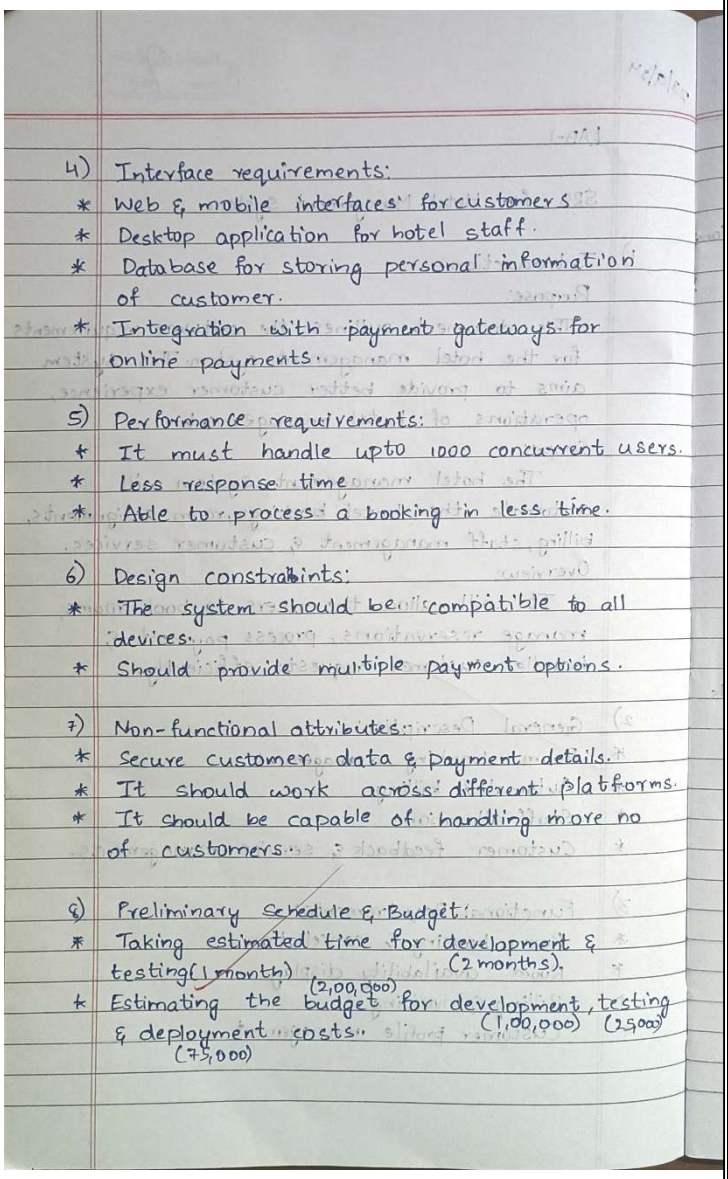
LAB-1

SRS for Hotel Management System.

1) Introduction:
Purpose:
This document outlines the software requirements for the hotel management system. This system aims to provide better customer experience, operations of hotel management.
Scope:
The hotel management system will automate the hotel bookings, room assignments, billing, staff management & customer services.
Overview:
It will allow the users to book rooms, manage reservations, process payments & handle customer requests efficiently.

2) General Description:
* Room booking management
* Automated billing
* Staff allocation
* Customer feedback & service managements.

3) Functional Requirements:
* Room booking & cancellation
* Room availability display
* Billing & invoicing
* Customer profile management



4) Interface requirements:
* Web & mobile interfaces for customers
* Desktop application for hotel staff.
* Database for storing personal information of customer.
* Integration with payment gateways for online payments.

5) Performance requirements:
* It must handle upto 1000 concurrent users.
* Less response time
* Able to process a booking in less time.

6) Design constraints:
* The system should be compatible to all devices.
* Should provide multiple payment options.

7) Non-functional attributes:
* Secure customer data & payment details.
* It should work across different platforms.
* It should be capable of handling more no of customers.

8) Preliminary Schedule & Budget:
* Taking estimated time for development & testing (1 month) (2 months).
* Estimating the budget (2,00,000) for development, testing & deployment costs. (1,00,000) (25,000) (75,000)

UML DIAGRAMS

CLASS DIAGRAM

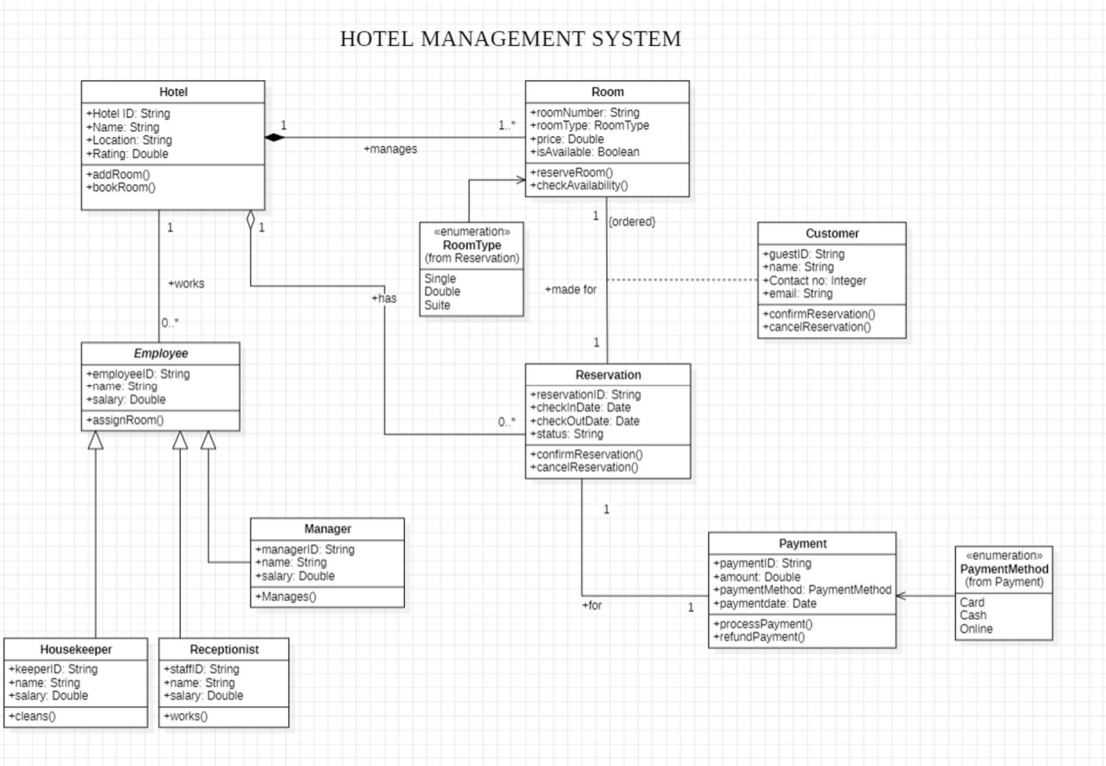
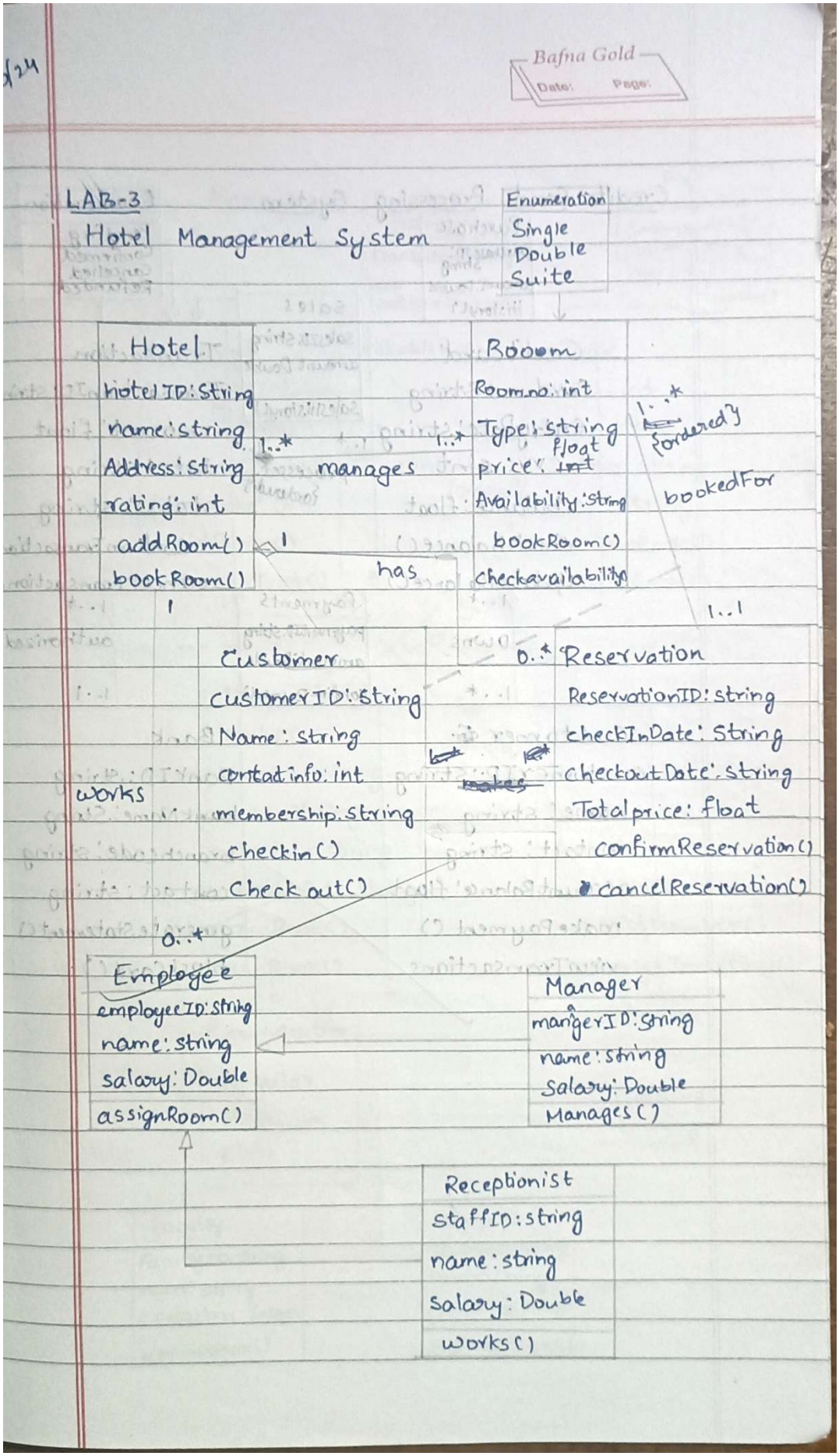


Figure 1.1: Class Diagram

The Hotel Management System Class Diagram outlines key entities and their relationships. The Hotel class manages details such as Hotel ID, Name, Location, and Rating, and oversees Room entities, which include attributes like room type, availability, and price. Customers can reserve rooms through the Reservation class, linked to Payment, which handles payment processing and refunds. The Employee class includes roles like Manager, Receptionist, and Housekeeper, each with specific responsibilities such as managing hotels, assisting guests, or cleaning rooms. Enumerations like RoomType (e.g., SINGLE, DOUBLE, SUITE) and PaymentMethod (e.g., Cash, Card, Online) define room categories and payment options. These interconnected classes ensure smooth hotel operations, efficient staff management, and an enhanced customer experience.


LAB-3
Hotel Management System
Enumeration
Single
Double
Suite
Hotel
hotelID: String
name: string
Address: string
rating: int
addRoom()
bookRoom()
1..*
manages
1
has
Room
Room.no: int
Type: string
price: float
Availability: String
bookRoom()
checkavailability
1..*
ordered
bookedFor
Customer
customerID: String
Name: string
contactinfo: int
membership: string
checkin()
Check out()
works
0..*
Reservation
ReservationID: string
checkInDate: String
checkoutDate: string
Total price: float
confirmReservation()
cancelReservation()
1..1
Employee
employeeID: string
name: string
salary: Double
assignRoom()
Manager
mangerID: String
name: string
Salary: Double
Manages()
Receptionist
staffID: string
name: string
salary: Double
works()

STATE DIAGRAM

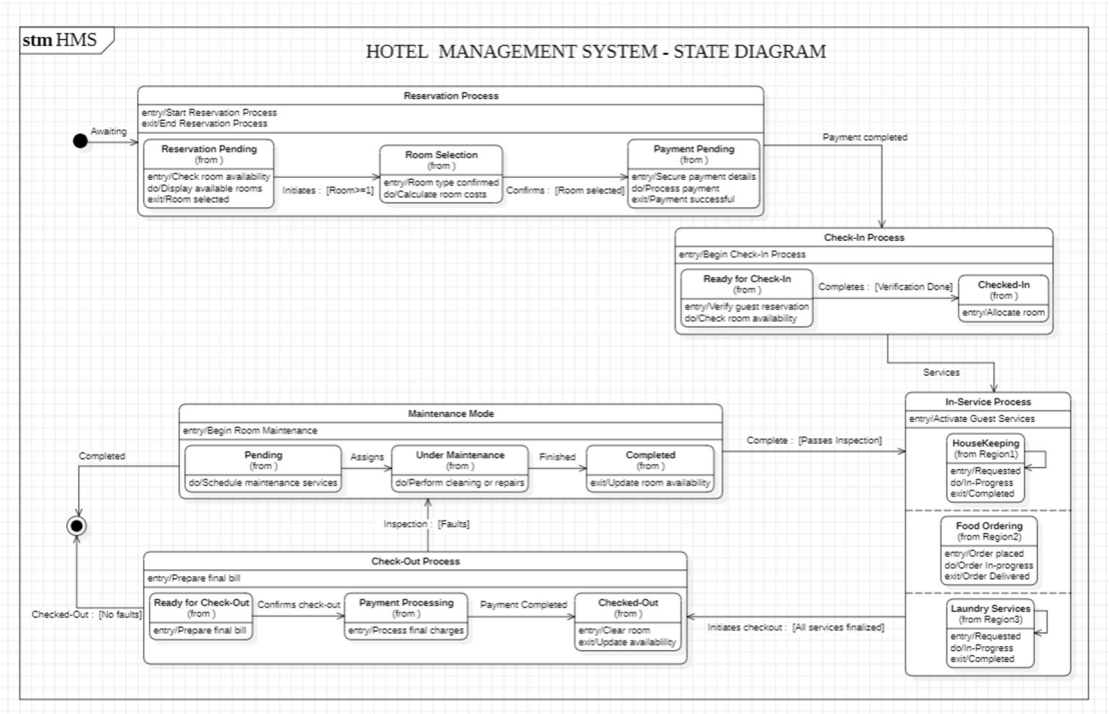
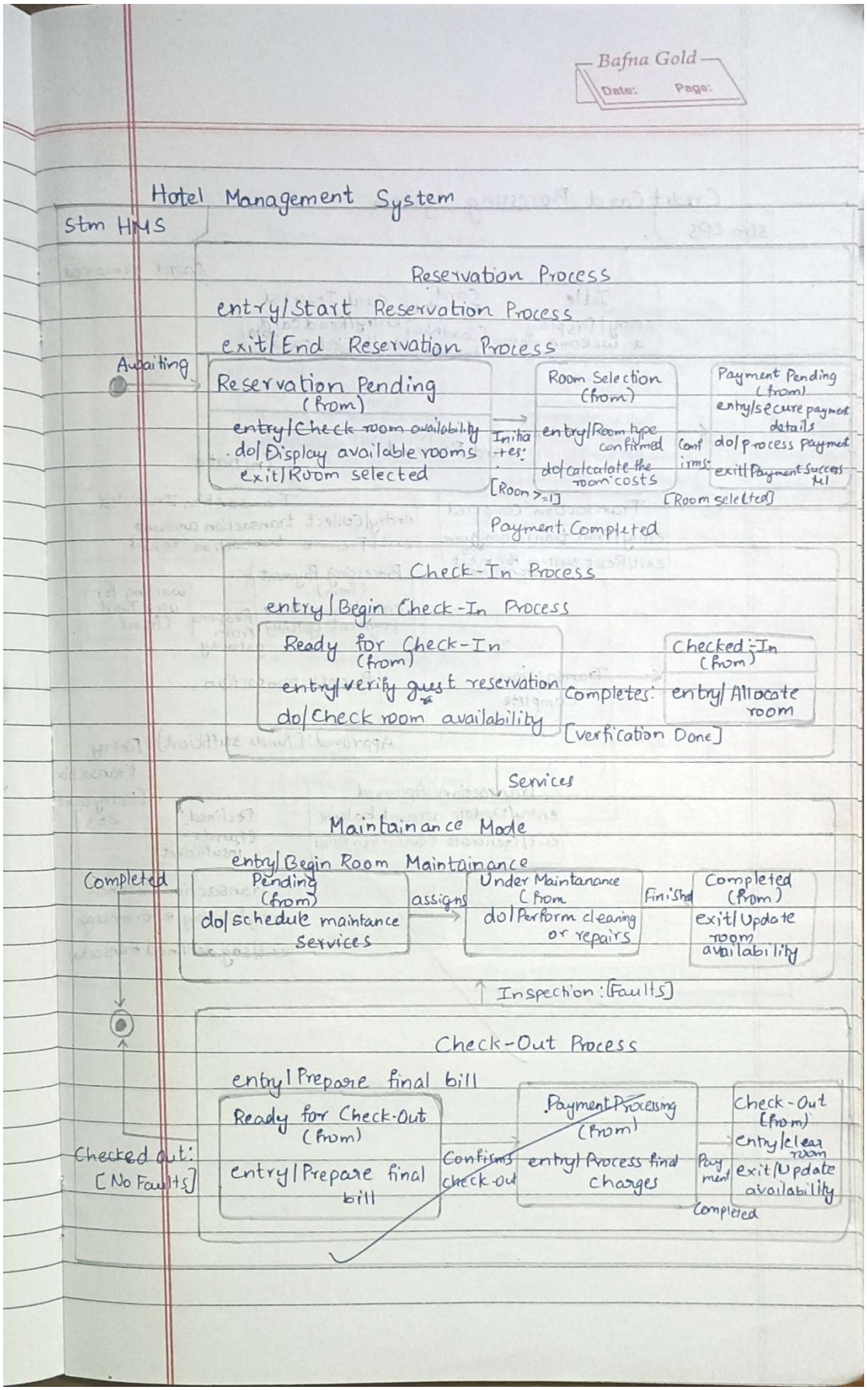


Figure 1.2: State Diagram

The hotel management system state diagram outlines the dynamic workflow of hotel operations, starting from reservations to guest services. The reservation process begins in the awaiting reservation state, where the system awaits new requests. Once a reservation is initiated, it transitions to reservation pending, where room availability is checked. Guests then move to the room selection state, choosing their preferred room, followed by payment pending for payment processing. Upon successful payment, the system transitions to payment completed, confirming the reservation. Finally, when guests arrive, they enter the checked-in state, where rooms are assigned.

The advanced in-service process state captures nested service flows provided after check-in. Key nested states include housekeeping, food ordering, and laundry services, each managing specific guest needs. Within these states, services transition through requested, in-progress, and completed phases, ensuring efficient handling of guest requests.

This state diagram integrates high-level reservation and check-in processes with detailed guest service workflows, ensuring seamless operations, enhanced guest satisfaction, and efficient resource management.


Bafna Gold
Date: Page:
Hotel Management System
Stm HMS
Reservation Process
entry/ Start Reservation Process
exit/ End Reservation Process
Awaiting
Reservation Pending (from)
entry/ Check room availability
do/ Display available rooms
exit/ Room selected
Initiates
[Room >=1]
Room Selection (from)
entry/ Room type confirmed
do/ calculate the room costs
Confirms
[Room selected]
Payment Pending (from)
entry/ secure payment details
do/ process payment
exit/ Payment Successful
Payment Completed
Check-In Process
entry/ Begin Check-In Process
Ready for Check-In (from)
entry/ verify guest reservation
do/ Check room availability
Completes
[verification Done]
Checked-In (from)
entry/ Allocate room
Services
Maintainance Mode
entry/ Begin Room Maintainance
Completed
Pending (from)
do/ schedule maintance services
assigns
Under Maintainance (from
do/ Perform cleaning or repairs
Finishes
Completed (from)
exit/ Update room availability
Inspection: [Faults]
Check-Out Process
entry/ Prepare final bill
Checked out: [No Faults]
Ready for Check-Out (from)
entry/ Prepare final bill
Confirms check-out
Payment Processing (from)
entry/ Process final charges
Payment Completed
Check-Out (from)
entry/ clear room
exit/ Update availability

USE CASE DIAGRAM

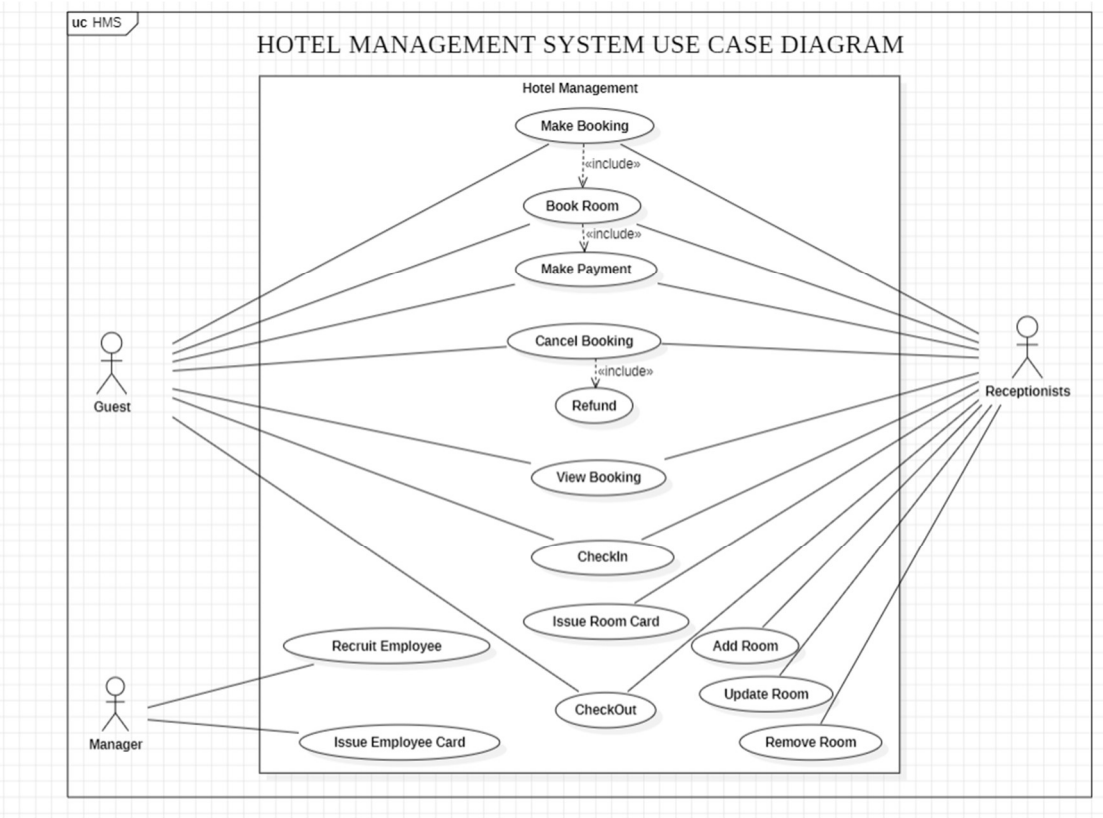
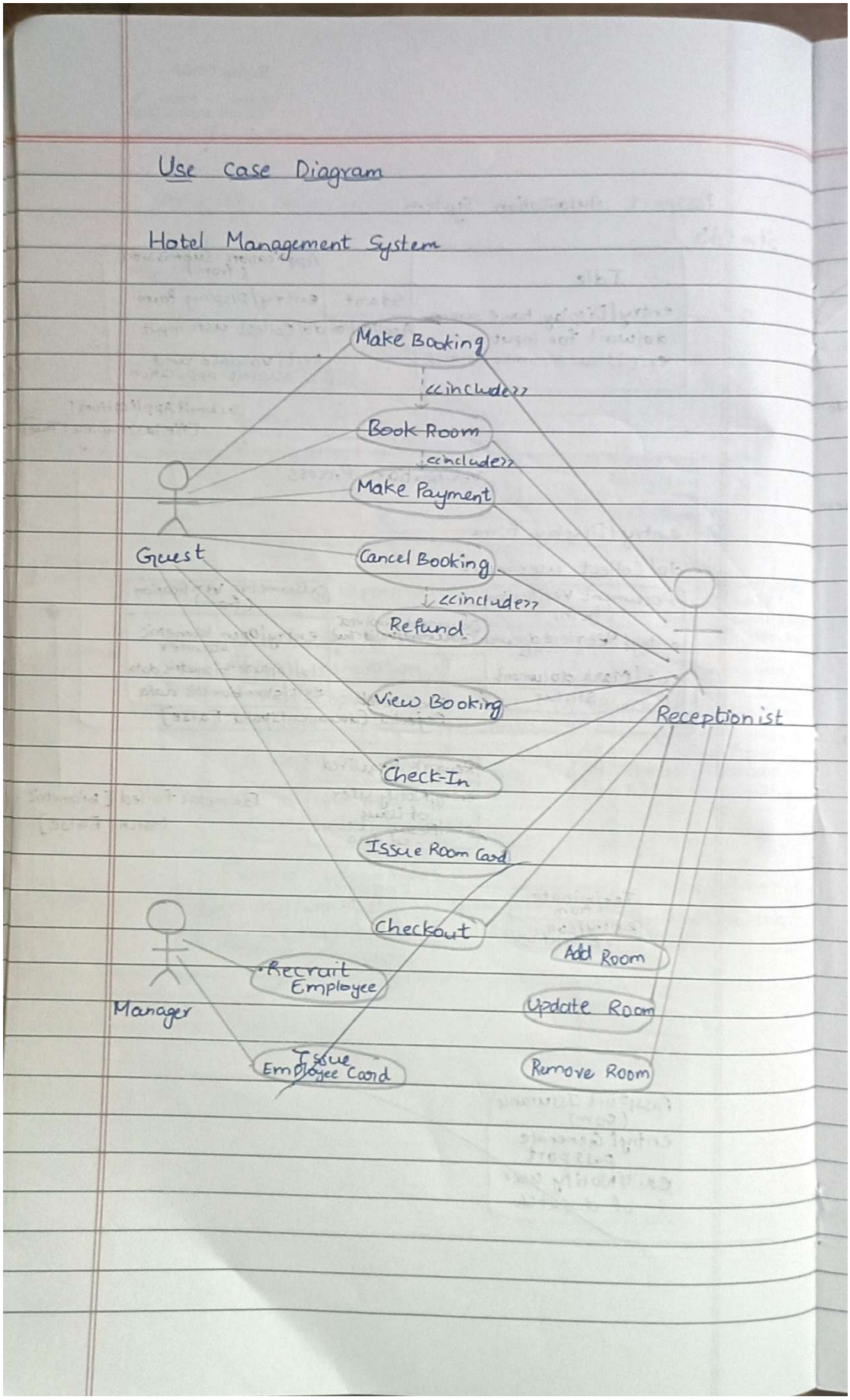


Figure 1.3: Use Case Diagram

The Hotel Management System includes various actors such as the Manager, Guest, and Receptionist, each responsible for different aspects of hotel operations. Guests can make, update, view, and cancel bookings, as well as make payments and check in or out. Receptionists assist with guest check-in, issue room keys, and facilitate checkouts. Managers oversee the hotel's operations, including adding and updating rooms, managing employee records, and issuing employee ID cards.

The use cases are organized with parent-child relationships, where Hotel Management encompasses multiple tasks, such as Make Booking, Book Room, Update Booking, and Refund. Receptionists and managers have specific roles, with receptionists handling guest interactions like check-ins and room key issuance, while managers focus on employee management and room inventory. This diagram outlines the key functionalities and interactions between actors and the system, illustrating the roles and responsibilities within hotel operations.


Use case Diagram
Hotel Management System
Make Booking
<<include>>
Book Room
<<include>>
Make Payment
Guest
Cancel Booking
<<include>>
Refund
View Booking
Receptionist
Check-In
Issue Room Card
Checkout
Add Room
Recruit Employee
Update Room
Manager
Issue Employee Coord
Remove Room

SEQUENCE DIAGRAM

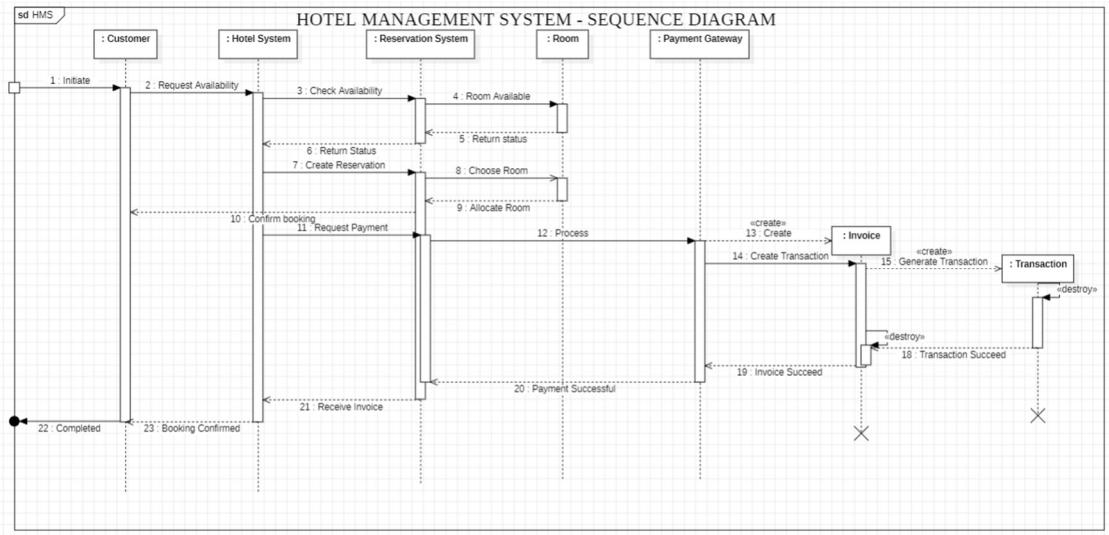
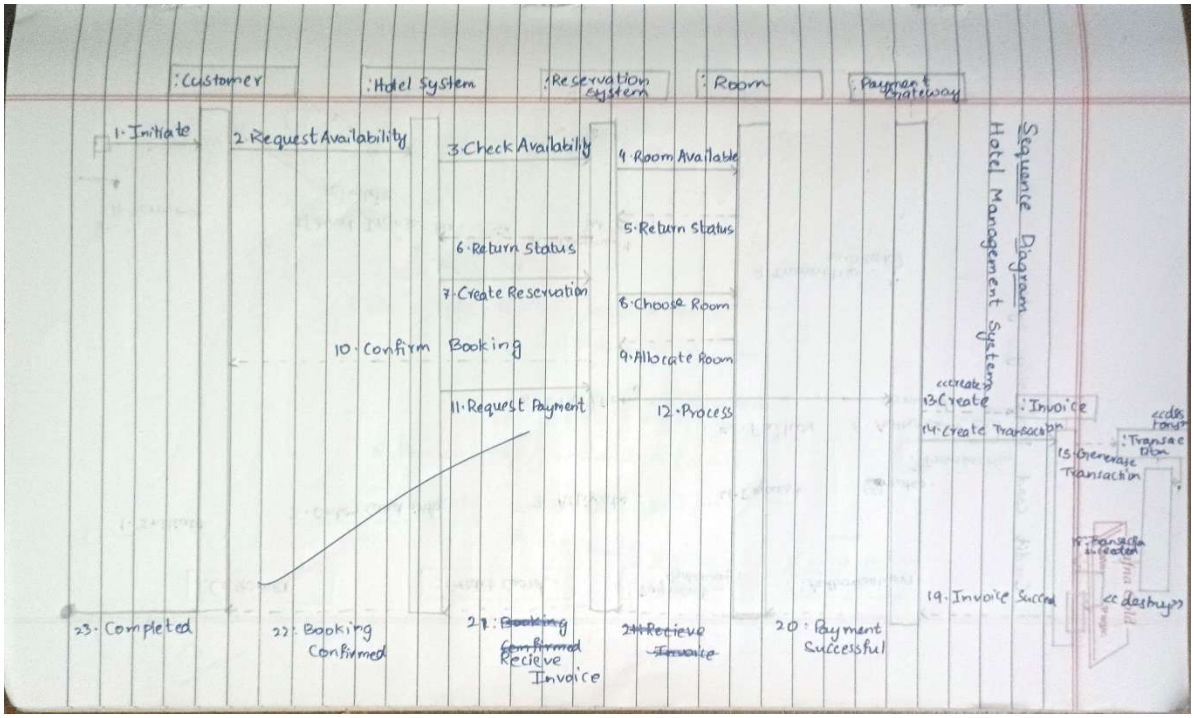


Figure 1.4: Sequence Diagram

In Scenario 1, the process begins with the customer checking room availability by sending a request to the Hotel System. The system then queries the Reservation System for available rooms based on the requested dates. Once the customer selects a room and provides payment details, the Hotel System processes the payment through the Payment Gateway, generating an invoice and transaction receipt for the customer. The process ends with the Hotel System confirming the booking.

Scenario 2 starts when the customer sends a cancellation request to the Hotel System. The system checks the reservation status and updates it to "Cancelled" in the Reservation System, releasing the allocated room. If applicable, a refund process is initiated. The Simple Sequence Diagram outlines interactions between objects like the User, Authentication System, Database, Application, and UI during a login process. The Advanced Sequence Diagram details how transient objects (Order and Order Line) interact with passive objects (Product and Inventory) to complete and process an order.


Sequence Diagram
Hotel Management System
:Customer
:Hotel System
:Reservation System
:Room
:Payment Gateway
1. Initiate
2 Request Availability
3. Check Availability
4. Room Available
5. Return Status
6. Return Status
7. Create Reservation
8. Choose Room
9. Allocate Room
10. Confirm Booking
11. Request Payment
12. Process
<<create>>
13. Create
:Invoice
14. Create Transaction
15. Generate Transaction
19. Invoice Succ
<<destroy>>
23. Completed
22: Booking Confirmed
21: Recieve Invoice
20: Payment Successful

ACTIVITY DIAGRAM

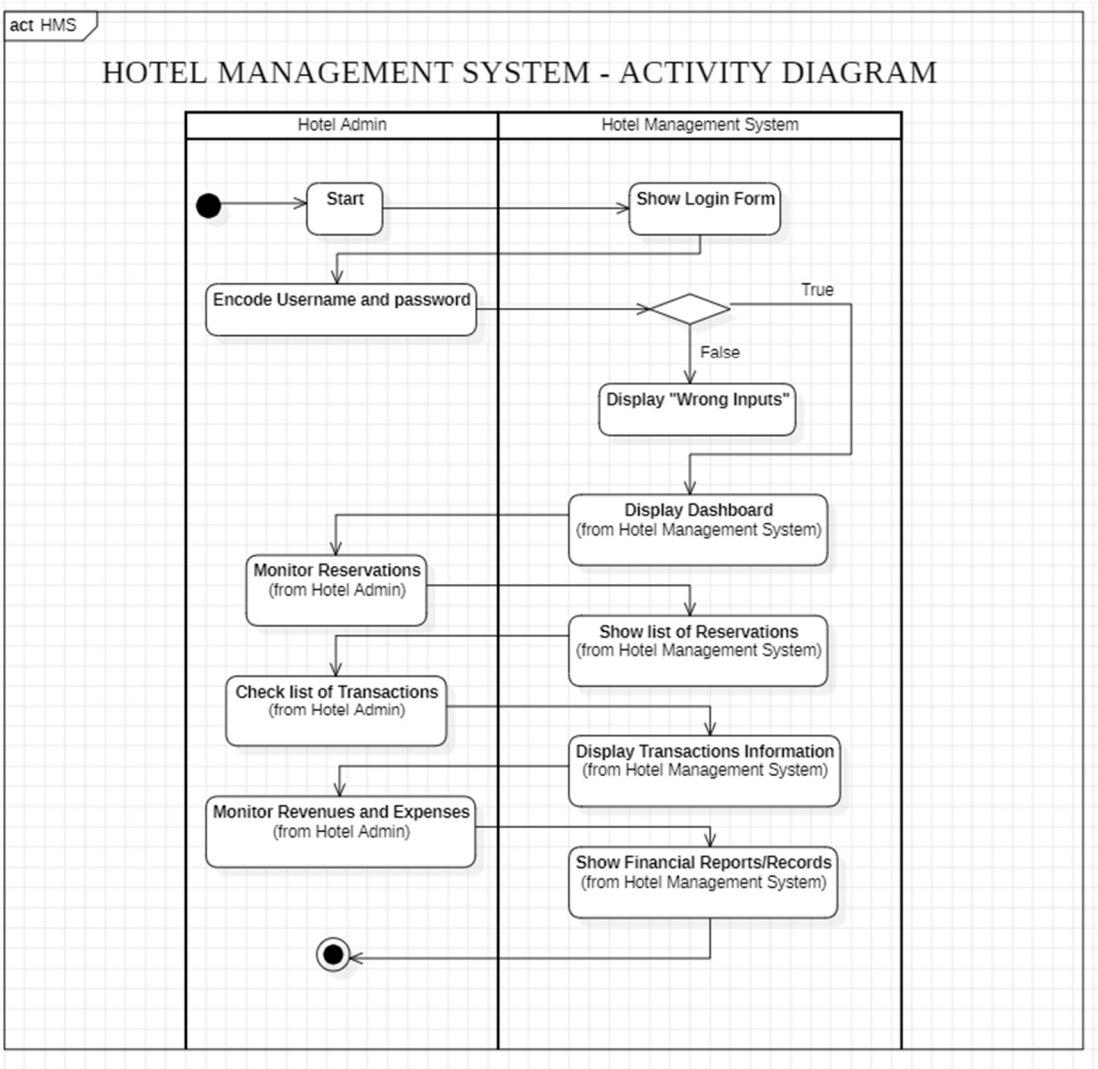


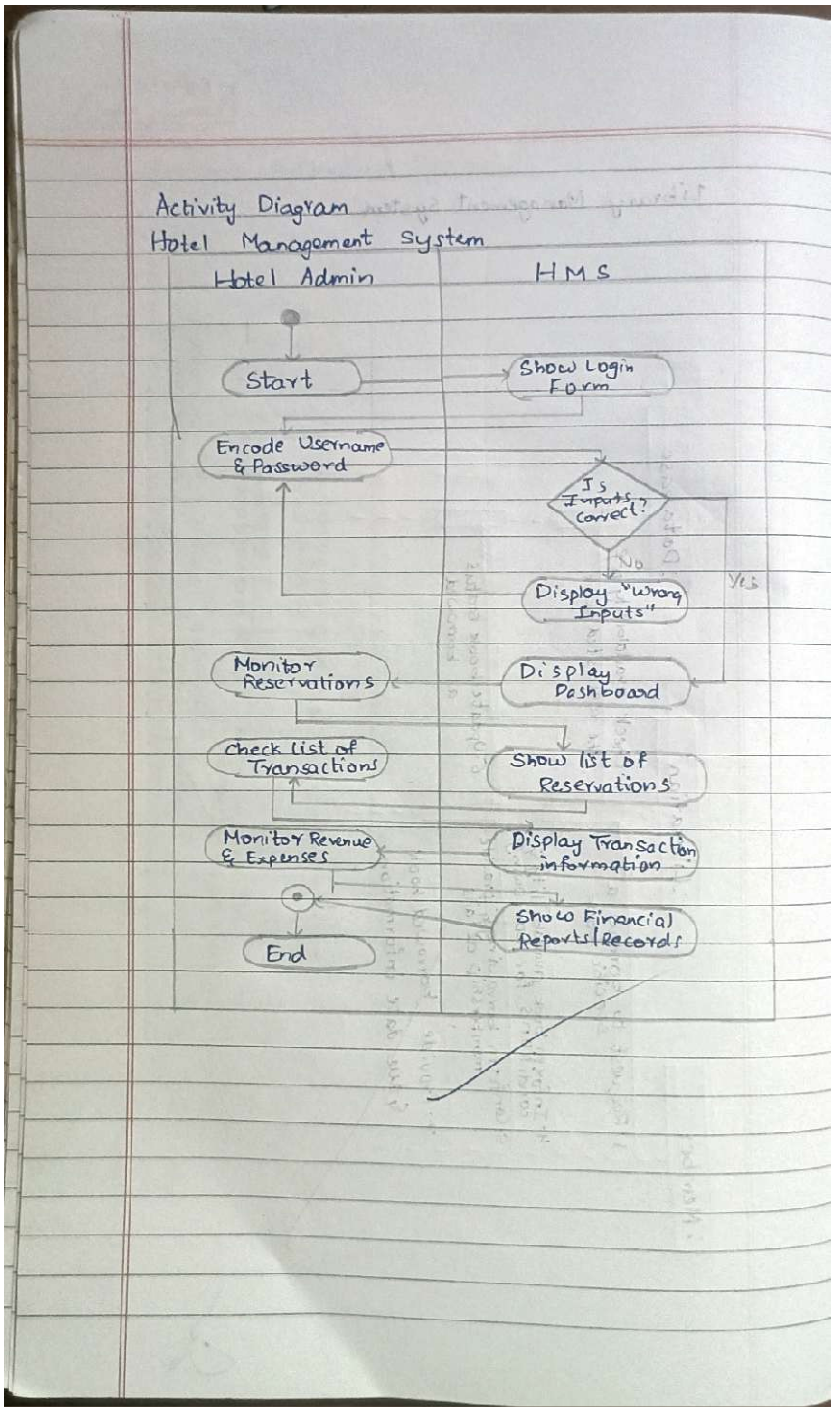
Figure 1.5: Activity Diagram

This swimlane diagram divides the actions into two swimlanes: the hotel admin and the hotel management system.

In the hotel admin swimlane, the hotel admin starts by logging in, entering their credentials, and then performs tasks such as monitoring reservations, checking transaction records, and analyzing financial performance, including revenues and expenses. The admin's actions guide the flow of the system.

In the hotel management system swimlane, the system first presents the login form, then processes login attempts by displaying an error message for incorrect credentials or a dashboard if successful. The system also displays various types of information to the admin, such as the list of reservations, transaction details, and

financial reports. These actions are prompted by the admin's requests and actions. This structure helps visualize how the admin interacts with the system to monitor and manage the hotel's operations.



CHAPTER – 2

CREDIT CARD PROCESSING SYSTEM

PROBLEM STATEMENT

The Credit Card Processing System is a secure and efficient solution designed to facilitate electronic payments in both physical and digital environments. Traditional payment methods or poorly integrated systems can lead to transaction delays, security vulnerabilities, and increased fraud risks. This system ensures seamless transaction processing by encrypting data, verifying payment details in real-time, and integrating with banking networks. It helps businesses provide fast, reliable, and secure payment options, enhancing trust and customer convenience.

- What is the system for?

The Credit Card Processing System facilitates secure, quick, and reliable electronic transactions. It ensures smooth payment processing by handling tasks like authorization, fraud detection, transaction settlement, and record-keeping.

- Where is it used?

This system is widely used in retail stores, online e-commerce platforms, restaurants, hotels, and banks. It is also employed by payment gateways and financial institutions for large-scale payment processing.

- Who is it for?

The system benefits customers using credit cards for purchases, merchants accepting card payments, and banks or financial institutions issuing and processing the cards. It also serves payment gateway providers that facilitate secure transactions.

- When is it required?

It is required during every transaction involving credit cards, whether for shopping, bill payments, subscriptions, or online services. It plays a critical role in environments where seamless and secure payment is a priority.

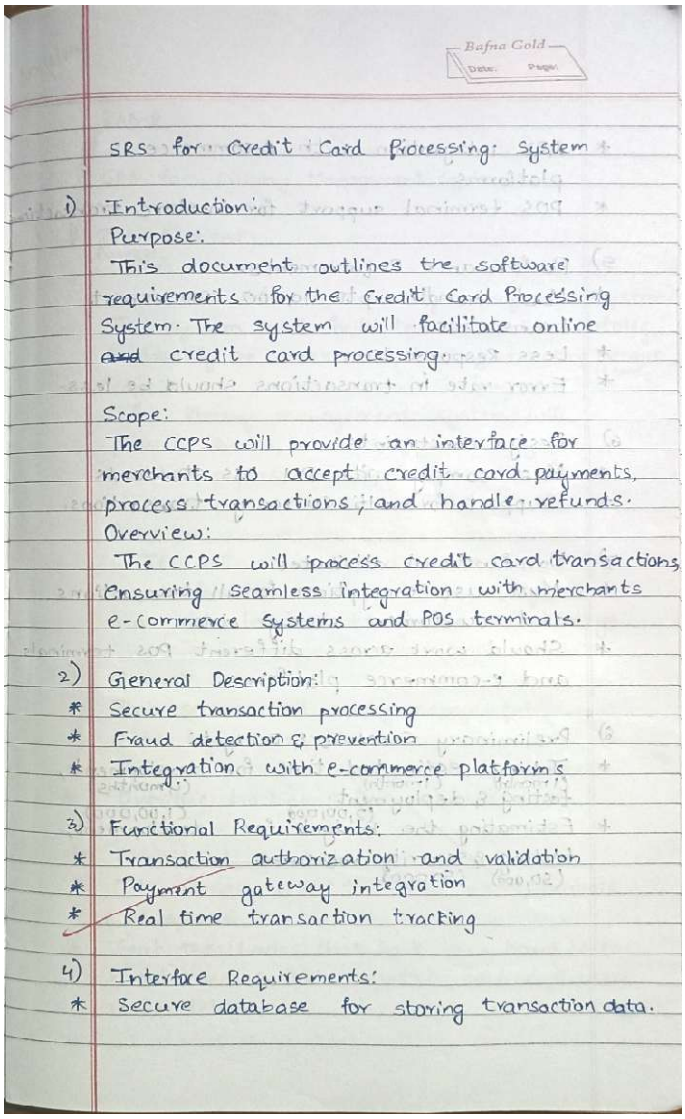
- Why is it needed?

It is needed to enhance customer convenience, ensure secure transactions, and build trust between merchants and consumers. The system also minimizes fraud risks, reduces manual reconciliation tasks, and ensures compliance with global payment standards like PCI DSS.

- How is it done?

The system uses POS terminals, online payment gateways, or mobile apps to initiate transactions. Data encryption ensures security, while backend systems verify card details with issuing banks in real time. Advanced fraud detection mechanisms monitor suspicious activities to safeguard transactions

SOFTWARE REQUIREMENTS SPECIFICATION



SRS for Credit Card Processing System

1) Introduction:
Purpose:
This document outlines the software requirements for the Credit Card Processing System. The system will facilitate online credit card processing.

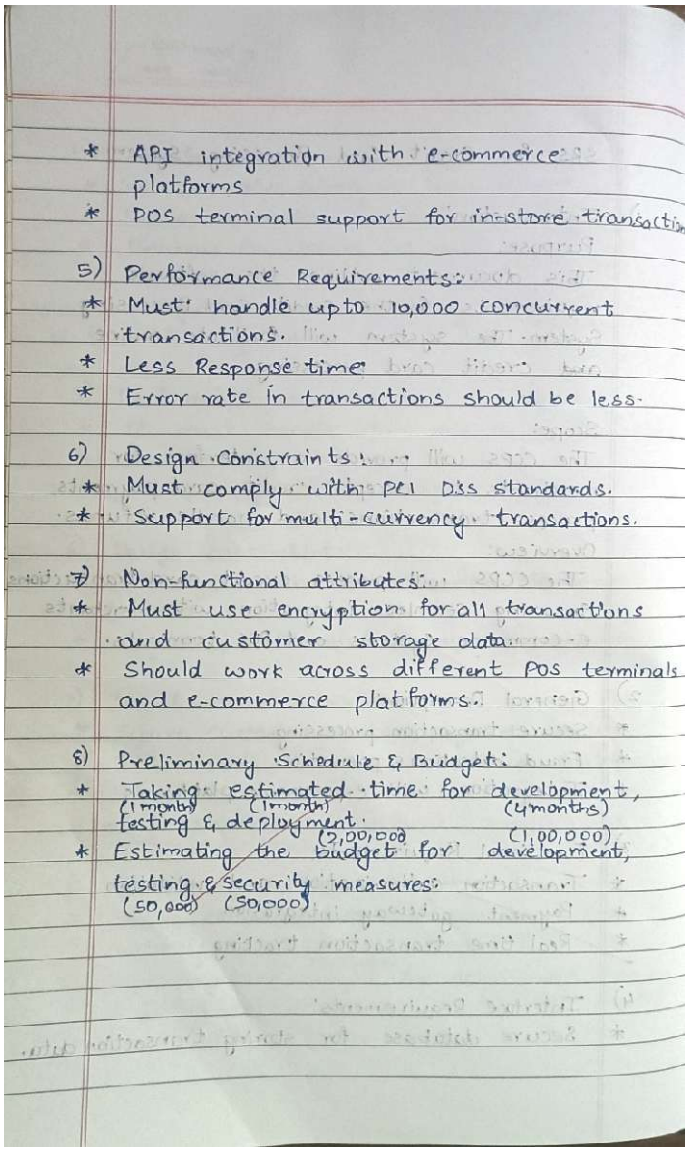
Scope:
The CCPS will provide an interface for merchants to accept credit card payments, process transactions, and handle refunds.

Overview:
The CCPS will process credit card transactions, ensuring seamless integration with merchants e-commerce systems and POS terminals.

2) General Description:
* Secure transaction processing
* Fraud detection & prevention
* Integration with e-commerce platforms

3) Functional Requirements:
* Transaction authorization and validation
* Payment gateway integration
* Real time transaction tracking

4) Interface Requirements:
* Secure database for storing transaction data.



* API integration with e-commerce platforms
* POS terminal support for in-store transaction

5) Performance Requirements:
* Must handle upto 10,000 concurrent transactions.
* Less Response time
* Error rate in transactions should be less.

6) Design Constraints:
* Must comply with PCI DSS standards.
* Support for multi-currency transactions.

7) Non-functional attributes:
* Must use encryption for all transactions and customer storage data.
* Should work across different POS terminals and e-commerce platforms.

8) Preliminary Schedule & Budget:
* Taking estimated time for development, (1 month) testing & deployment. (1 month) (2,00,000) (4 months) (1,00,000)
* Estimating the budget for development, testing & security measures: (50,000) (50,000)

UML DIAGRAMS

CLASS DIAGRAM

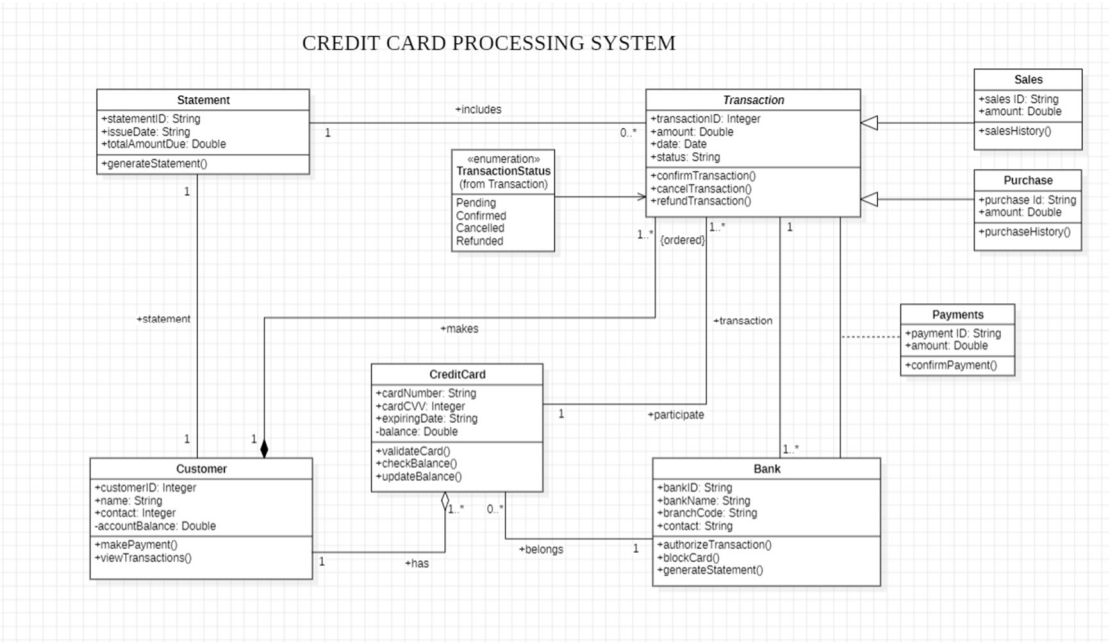
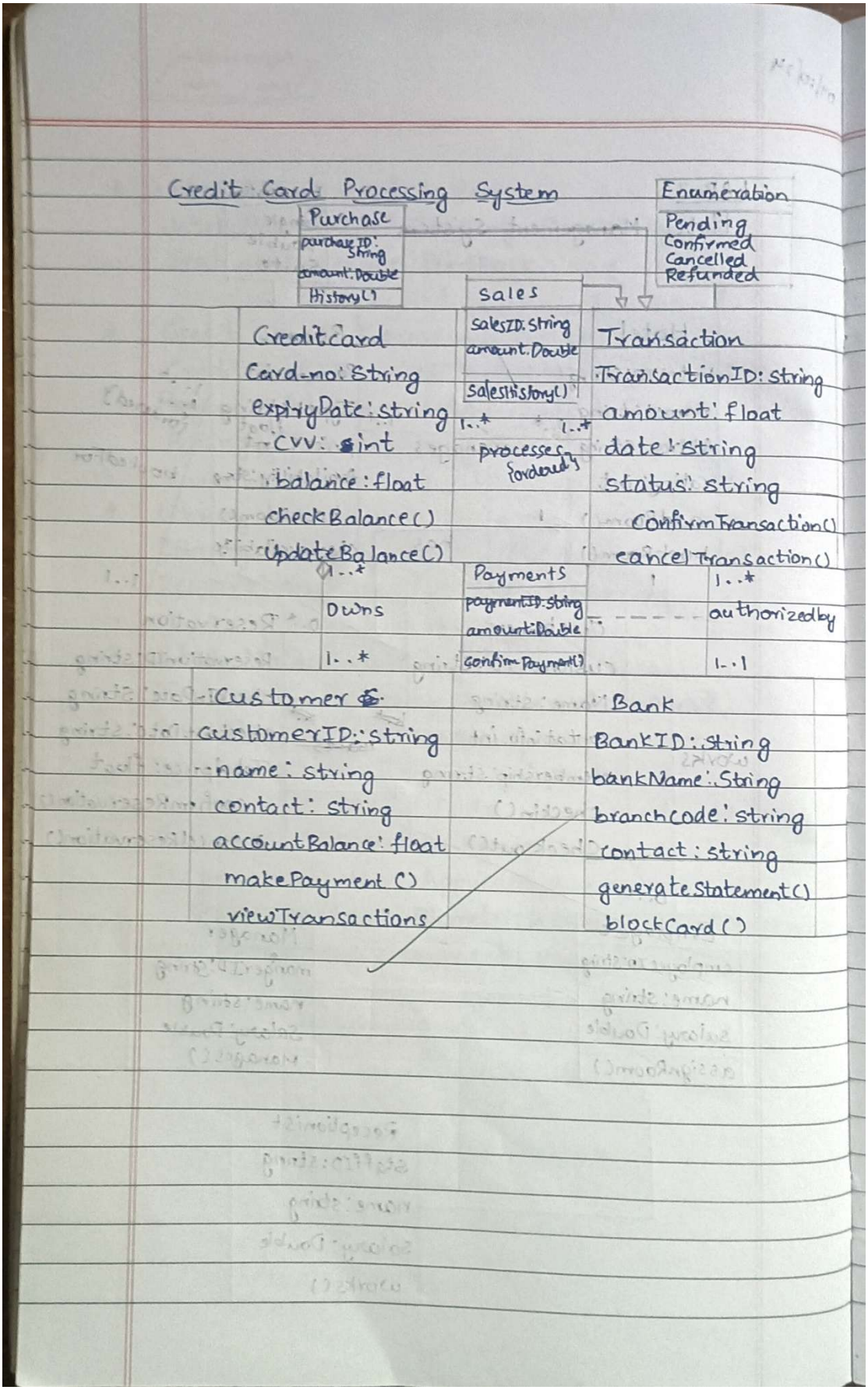


Figure 2.1: Class Diagram

The Credit Card Processing System Class Diagram defines the entities and relationships necessary to facilitate secure and efficient transaction management. The Customer class represents individuals using credit cards, with methods for making payments, viewing transactions, and managing account details. Customers may be classified as RegularCustomer or PremiumCustomer, inheriting additional privileges. Credit cards, represented by the CreditCard class, store card details such as balance, type, and expiration date, with methods for validation, balance checks, and blocking.

Transactions are managed by the Transaction class, which tracks attributes like transaction ID, amount, and status, and includes methods to initiate, confirm, cancel, and refund transactions. Statements, managed by the Statement class, aggregate transactions and provide detailed summaries for customers. Banks, represented by the Bank class, oversee credit card authorization, transaction settlement, and statement generation. Enumerations like TransactionStatus (e.g., Pending, Confirmed) and CardType (e.g., Platinum, Gold) further define the system's functionality. Interactions between these entities ensure smooth processing, customer satisfaction, and robust financial operations.


Credit Card Processing System
Enumeration
Pending
Confirmed
Cancelled
Refunded
Purchase
purchaseID: String
amount: Double
History()
Sales
salesID: String
amount: Double
salesHistory()
CreditCard
Card-no: String
expiryDate: String
CVV: int
balance: float
checkBalance()
updateBalance()
Transaction
TransactionID: String
amount: float
date: String
status: String
confirmTransaction()
cancelTransaction()
1..*
processes
1..*
1..*
Owns
1..*
Payments
paymentID: String
amount: Double
confirmPayment()
1..*
authorizedby
1..1
Customer
customerID: String
name: String
contact: String
accountBalance: float
makePayment()
viewTransactions
Bank
BankID: String
bankName: String
branchCode: String
contact: String
generateStatement()
blockCard()

STATE DIAGRAM

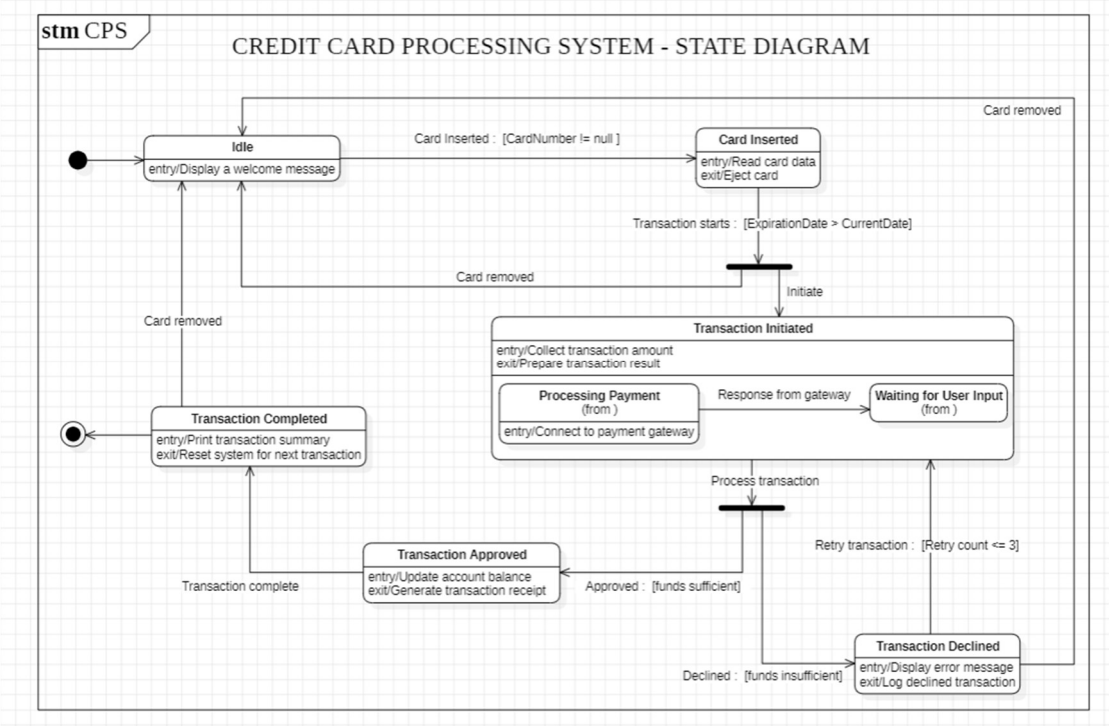
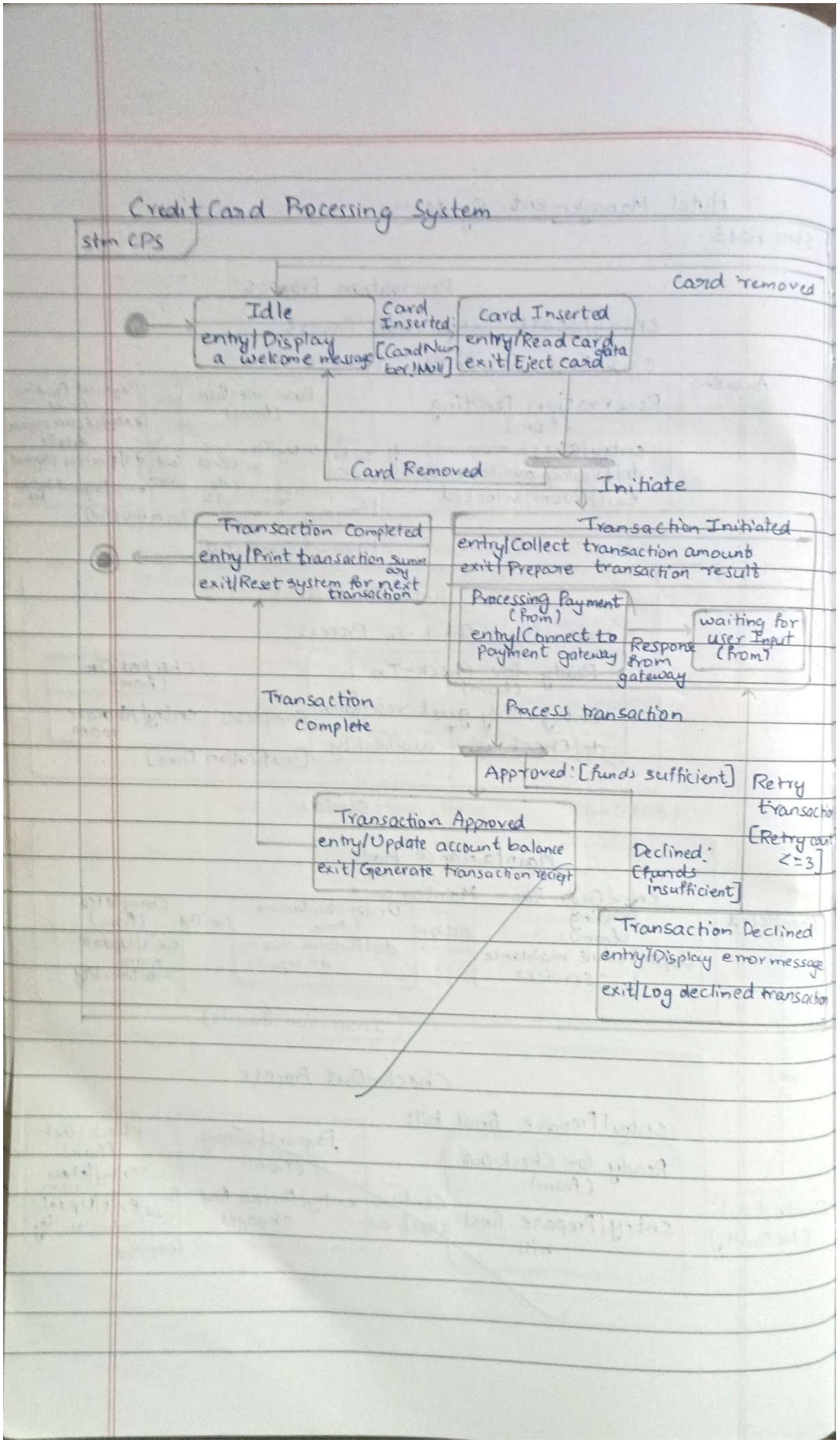


Figure 2.2: State Diagram

The Credit Card Processing System State Diagram outlines the flow of transactions, starting with the Idle state, where the system awaits card insertion. Once a card is inserted, the system transitions to the Card Inserted state, where card data is read. The process moves to Transaction Initiated, involving amount input and data collection, followed by Processing Payment, where the system connects with the payment gateway. If additional input is required, the system transitions to Waiting for User Input.

Based on the gateway's response, the transaction is either Approved, updating the account balance, or Declined, displaying an error message. The final state, Transaction Completed, resets the system for the next transaction, printing a transaction summary. Key events like card insertion, amount entry, and gateway responses trigger transitions, while activities such as reading card data, updating balances, and error handling ensure smooth processing.

This diagram provides a comprehensive view of the transaction workflow, detailing state transitions, triggered events, and activities, ensuring efficient and secure credit card processing.


Credit Card Processing System
stm CPS
Card removed
Idle
entry/ Display a welcome message
Card Inserted: [CardNumber.IsValid]
Card Inserted
entry/ Read card data
exit/ Eject card
Card Removed
Initiate
Transaction Completed
entry/ Print transaction summary
exit/ Reset system for next transaction
Transaction Initiated
entry/ Collect transaction amount
exit/ Prepare transaction result
Processing Payment (From)
entry/ Connect to payment gateway
Response from gateway
waiting for user Input (From)
Transaction complete
Process transaction
Approved: [funds sufficient]
Retry transaction
[Retry count <=3]
Transaction Approved
entry/ Update account balance
exit/ Generate transaction receipt
Declined: [funds insufficient]
Transaction Declined
entry/ Display error message
exit/ Log declined transaction

USE CASE DIAGRAM

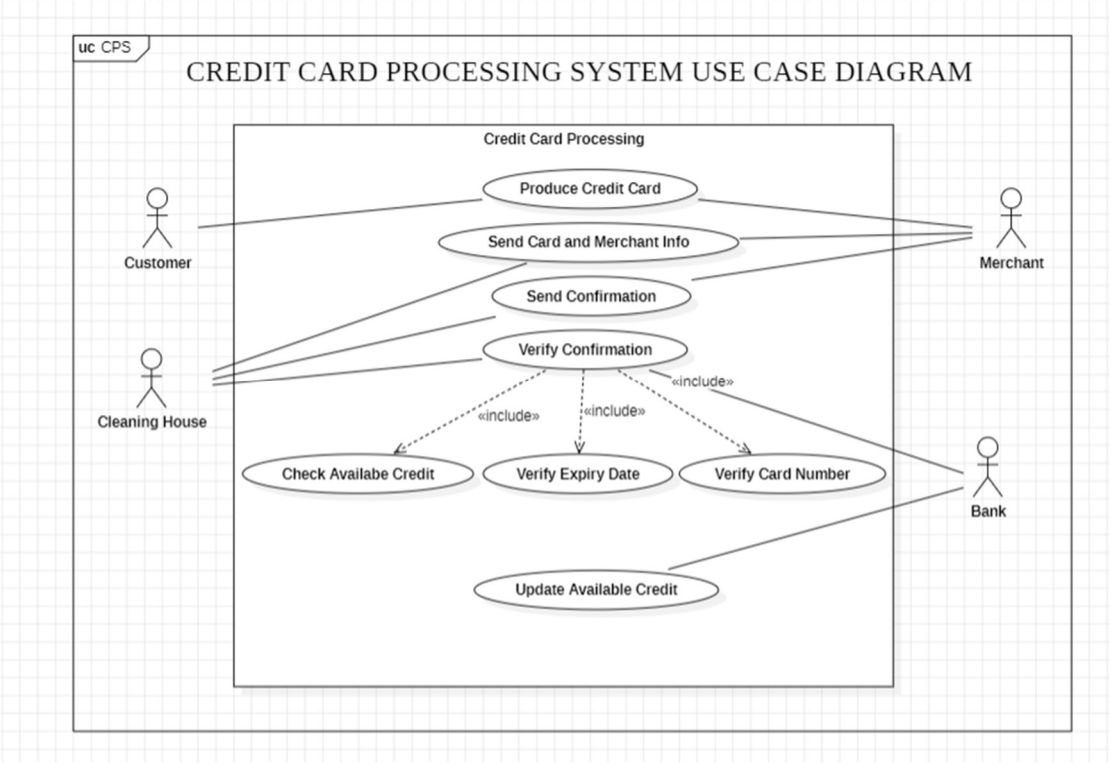
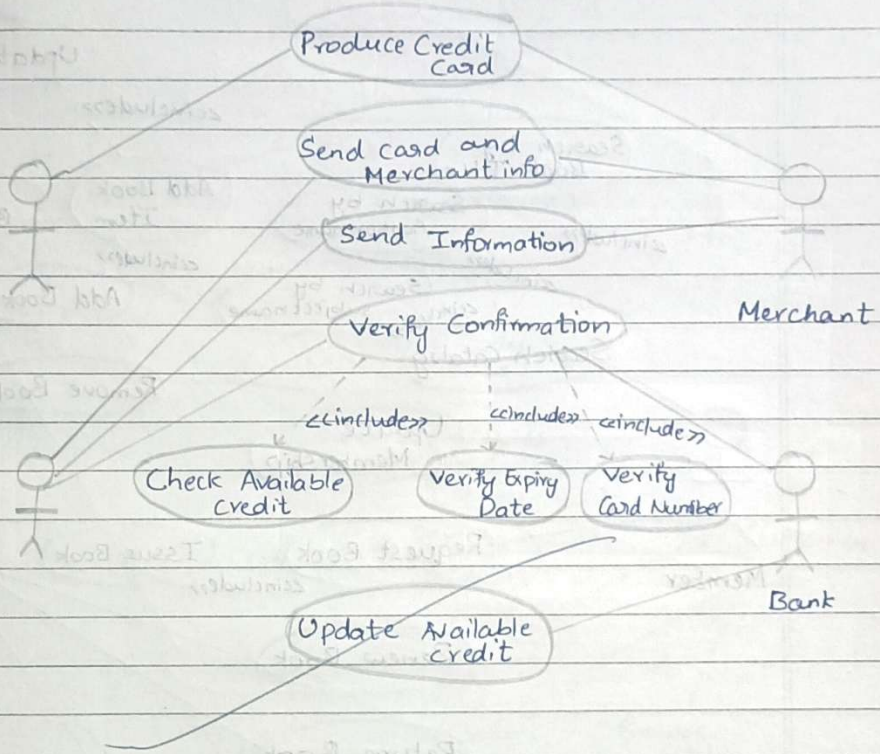


Figure 2.3: Use Case Diagram

The Credit Card Processing System involves several key actors: Customer, Merchant, Payment Processor, and Bank. Customers, including Regular, Elite, and Premium cardholders, initiate transactions with merchants. The merchant processes the transaction by sending card details to the Payment Processor, which handles the financial verification, including checking available credit and verifying the card's validity. The Bank issues the credit card and maintains the customer's account, while the Payment Processor ensures funds are authorized and settled.

The main use case, Credit Card Processing, encompasses several steps, such as Present Credit Card, Send Card and Merchant Info, and Send Confirmation, which are followed by a Verify Confirmation step that includes verifying the card number and expiry date. This process can be extended with optional fraud checks. The system also includes use cases for checking and updating available credit. The diagram helps visualize the relationships between the actors and use cases, including Includes and Extends relationships, and demonstrates the flow of information and actions in a typical credit card transaction.

Credit Card Processing System



SEQUENCE DIAGRAM

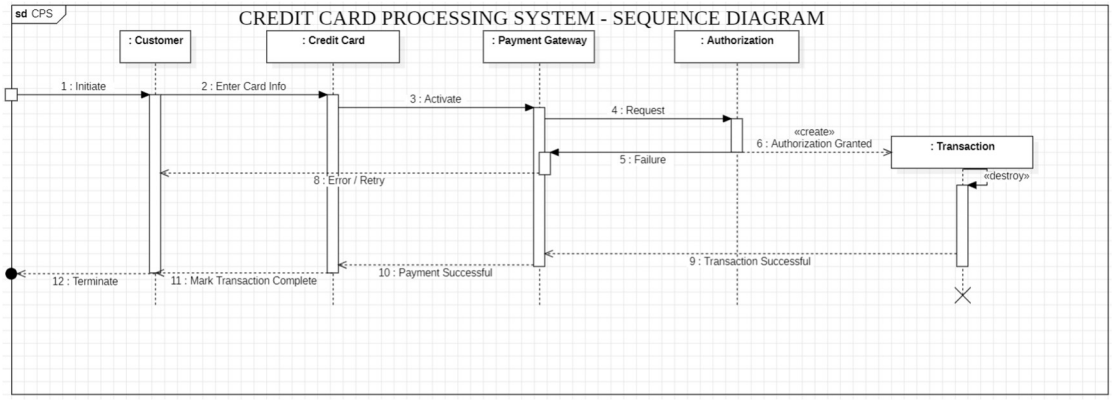
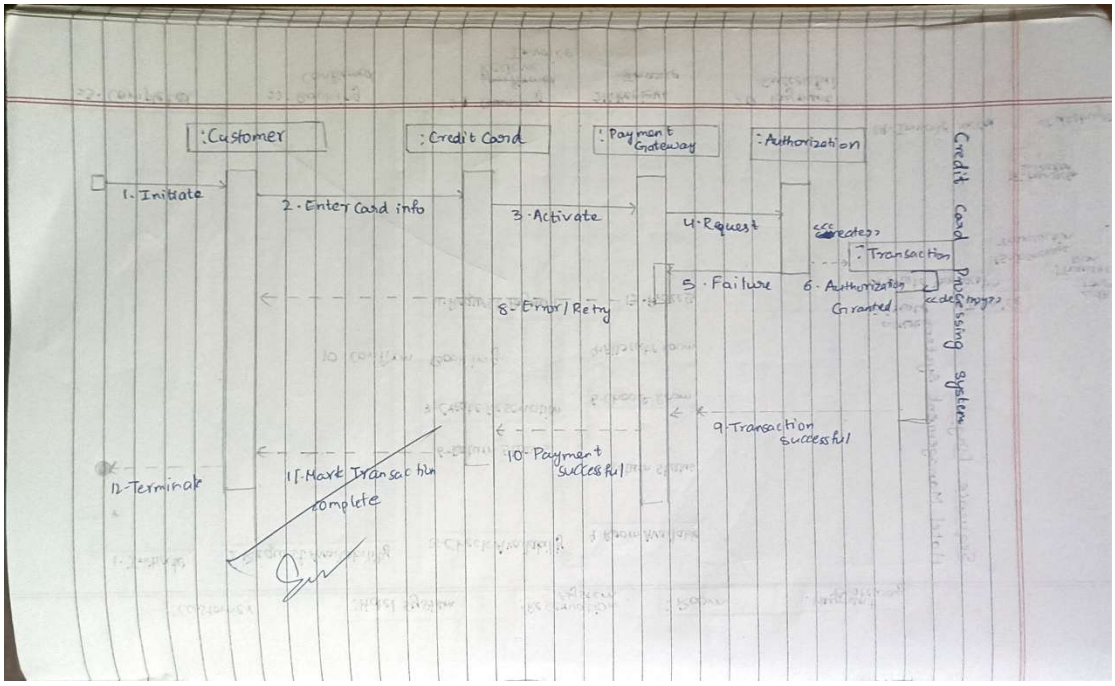


Figure 2.4: Sequence Diagram

In this sequence diagram, the customer initiates the transaction process and enters credit card information, which activates the credit card for use. The credit card sends a request for authorization to the payment gateway. If the transaction is authorized, the payment gateway grants authorization, and a transaction record is created. If the authorization fails due to issues like invalid card details or insufficient funds, a failure response is sent, and the process may be retried. Once authorization is successful, the credit card notifies the customer of the successful payment, marks the transaction complete, and the process terminates.

This sequence diagram highlights the interactions between the customer, credit card, payment gateway, and authorization during a typical credit card transaction. It illustrates the sequence of messages exchanged, and how the system manages different stages, including authorization, transaction creation, and failure handling.


:Customer
:Credit Card
:Payment Gateway
:Authorization
Credit Card Processing System
1. Initiate
2. Enter Card info
3. Activate
4. Request
<<create>>
:Transaction
5. Failure
6. Authorization Granted
<<destroy>>
8. Error/Retry
9. Transaction successful
10. Payment successful
11. Mark Transaction complete
12. Terminate

ACTIVITY DIAGRAM

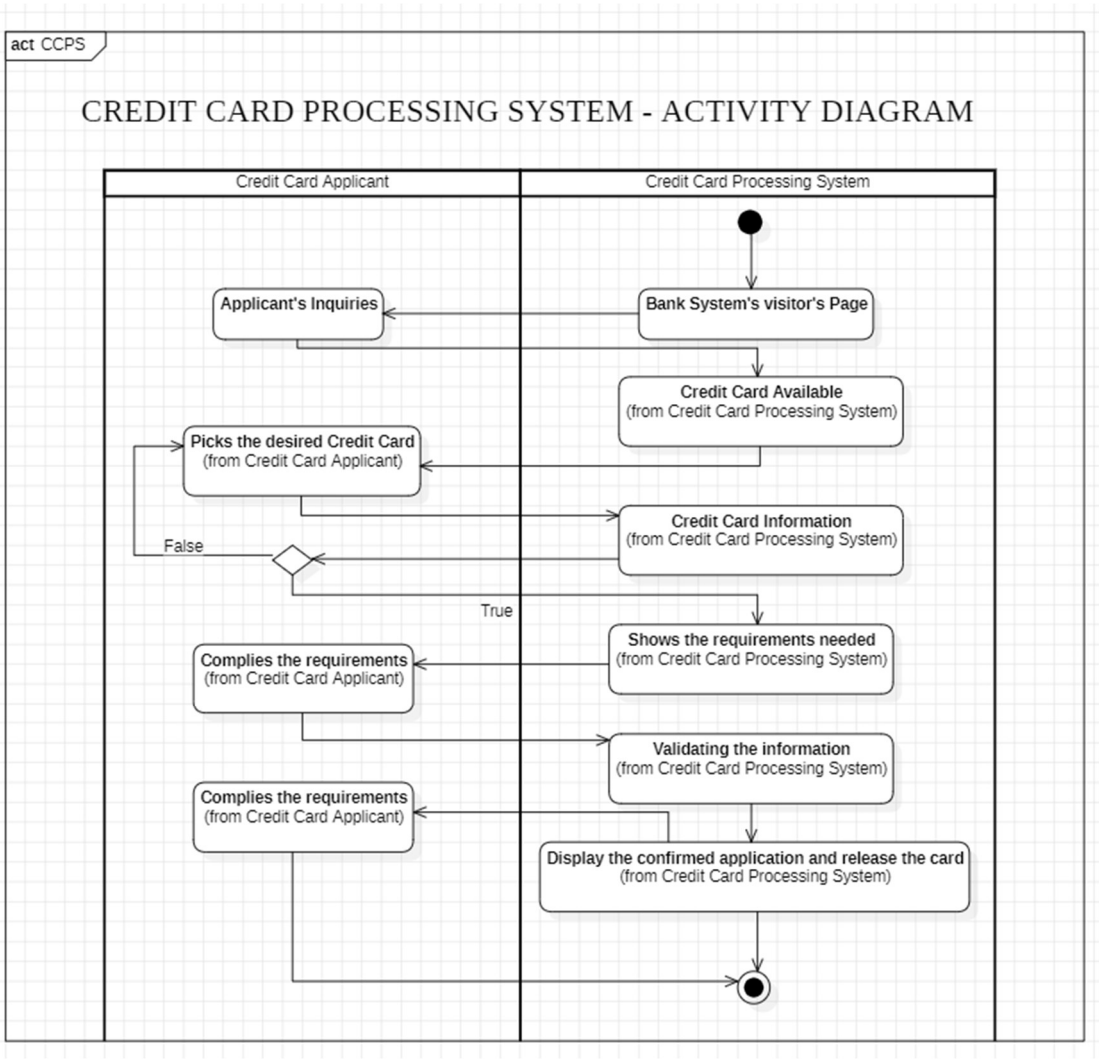
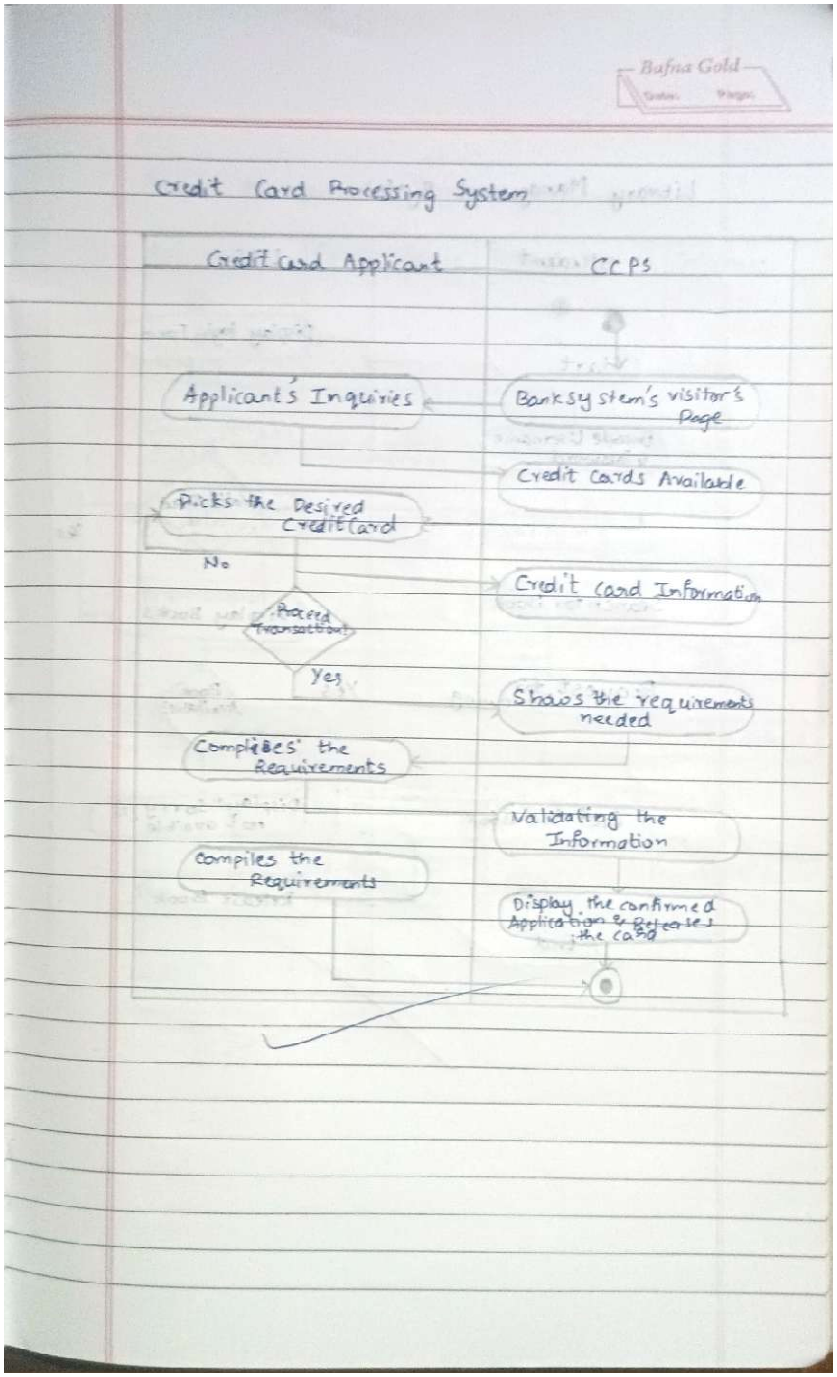


Figure 2.5: Activity Diagram

In this swimlane diagram, the actions are divided between the Credit Card Applicant and the Credit Card Processing System.

In the Credit Card Applicant swimlane, the applicant starts by inquiring about the available credit card options. After reviewing the options, the applicant selects their desired credit card and then complies with the requirements by providing the necessary information and documents for the application process.

In the Credit Card Processing System swimlane, the system first displays the available credit card options on the bank's visitor page. It then presents details about the selected credit card, including its features, and shows the requirements that need to be fulfilled by the applicant. After receiving the applicant's information, the system validates it and, upon successful validation, confirms the application and informs the applicant about the issuance of the credit card. This process helps visualize the sequence of steps and interactions between the applicant and the credit card system.


Credit Card Processing System
Credit card Applicant
CCPS
Applicant's Inquiries
Banksystem's visitor's Page
Credit cards Available
Picks the Desired Credit card
No
Credit card Information
Yes
Shows the requirements needed
Complies the Requirements
Validating the Information
compiles the Requirements
Display the confirmed

CHAPTER – 3

LIBRARY MANAGEMENT SYSTEM

PROBLEM STATEMENT

The Library Management System is an advanced software solution tailored to manage library operations effectively, from cataloging resources to tracking lending and returns. Manual or outdated library systems can result in misplaced items, inaccurate records, and inefficient services. This system centralizes resource management, enables real-time tracking of borrowed items, and provides users with easy access to materials through search features. It significantly reduces administrative workload and enhances the overall library experience for both staff and patrons.

- What is the system for?

The Library Management System is designed to digitize and automate library operations, including book cataloging, issuing, returning, member management, and inventory tracking. It enhances user access to library resources and simplifies administrative tasks.

- Where is it used?

It is used in public libraries, school and university libraries, private book collections, and corporate research libraries where a systematic organization of resources is essential.

- Who is it for?

The system is for librarians, library staff, students, researchers, and general users who borrow or access resources. It provides librarians with efficient tools for managing collections and offers users easy access to materials.

- When is it required?

It is required whenever libraries deal with large collections of books, journals, and digital media or experience high user traffic. It is particularly useful in academic settings where timely access to resources is crucial.

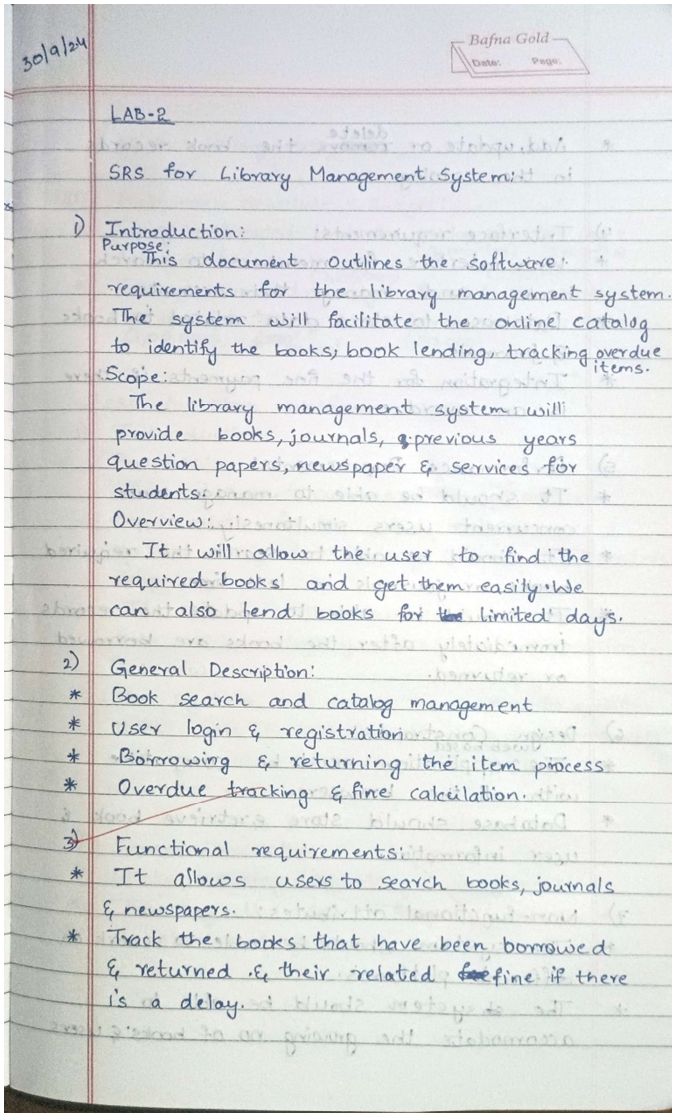
- Why is it needed?

It is needed to improve resource management, reduce the workload of library staff, enhance user experience, and ensure accurate tracking of borrowed items. It also enables data-driven decision-making through analytics on resource usage.

- How is it done?

The system is implemented through software that digitizes library catalogs, automates lending processes, and tracks user activity. Features like search engines, RFID tags, and mobile apps make the system more accessible and user-friendly.

SOFTWARE REQUIREMENTS SPECIFICATION



30/9/24

Bafna Gold
Date: Page:

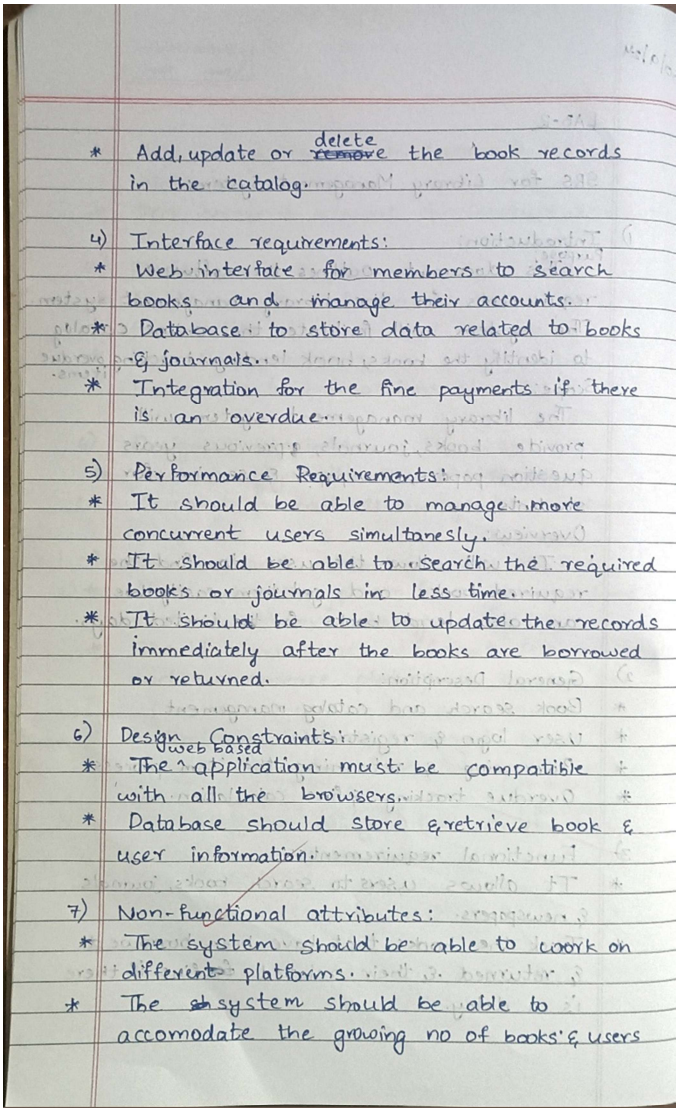
LAB-2

SRS for Library Management System:

1) Introduction:
Purpose: This document outlines the software requirements for the library management system. The system will facilitate the online catalog to identify the books, book lending, tracking overdue items.
Scope:
The library management system will provide books, journals, previous years question papers, newspaper & services for students.
Overview:
It will allow the user to find the required books and get them easily. We can also lend books for ~~the~~ limited days.

2) General Description:
* Book search and catalog management
* User login & registration
* Borrowing & returning the item process
* Overdue tracking & fine calculation.

3) Functional requirements:
* It allows users to search books, journals & newspapers.
* Track the books that have been borrowed & returned & their related ~~fine~~ fine if there is a delay.



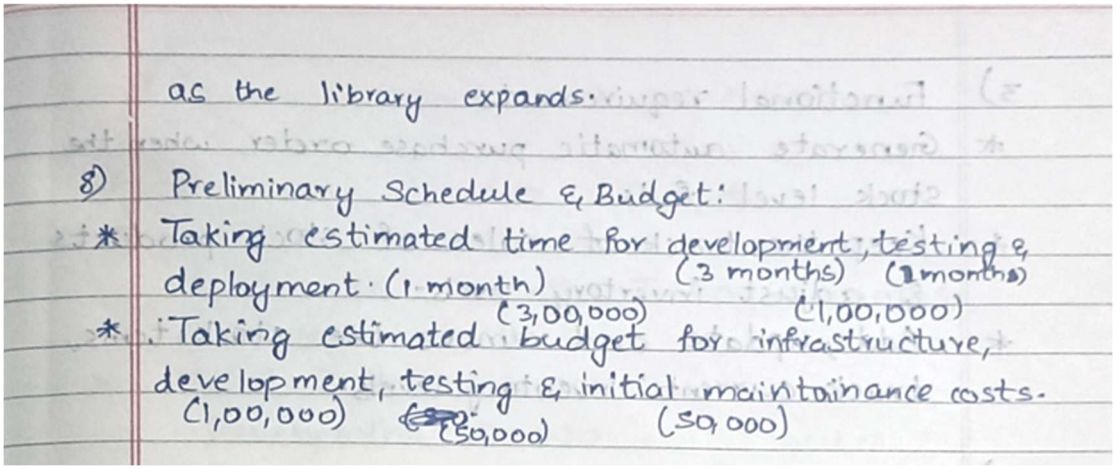
* Add, update or ~~remove~~ delete the book records in the catalog.

4) Interface requirements:
* Web interface for members to search books and manage their accounts.
* Database to store data related to books & journals.
* Integration for the fine payments if there is an overdue.

5) Performance Requirements:
* It should be able to manage more concurrent users simultanesly.
* It should be able to search the required books or journals in less time.
* It should be able to update the records immediately after the books are borrowed or returned.

6) Design Constraints:
* The web based application must be compatible with all the browsers.
* Database should store & retrieve book & user information.

7) Non-functional attributes:
* The system should be able to work on different platforms.
* The ~~sh~~ system should be able to accomodate the growing no of books & users



as the library expands.

8) Preliminary Schedule & Budget:
* Taking estimated time for development, testing & deployment. (1 month) (3 months) (1 months)
* Taking estimated budget for infrastructure, development, testing & initial maintainance costs. (3,00,000) (1,00,000) (1,00,000) ~~(50,000)~~ (50,000) (50,000)

UML DIAGRAMS

CLASS DIAGRAM

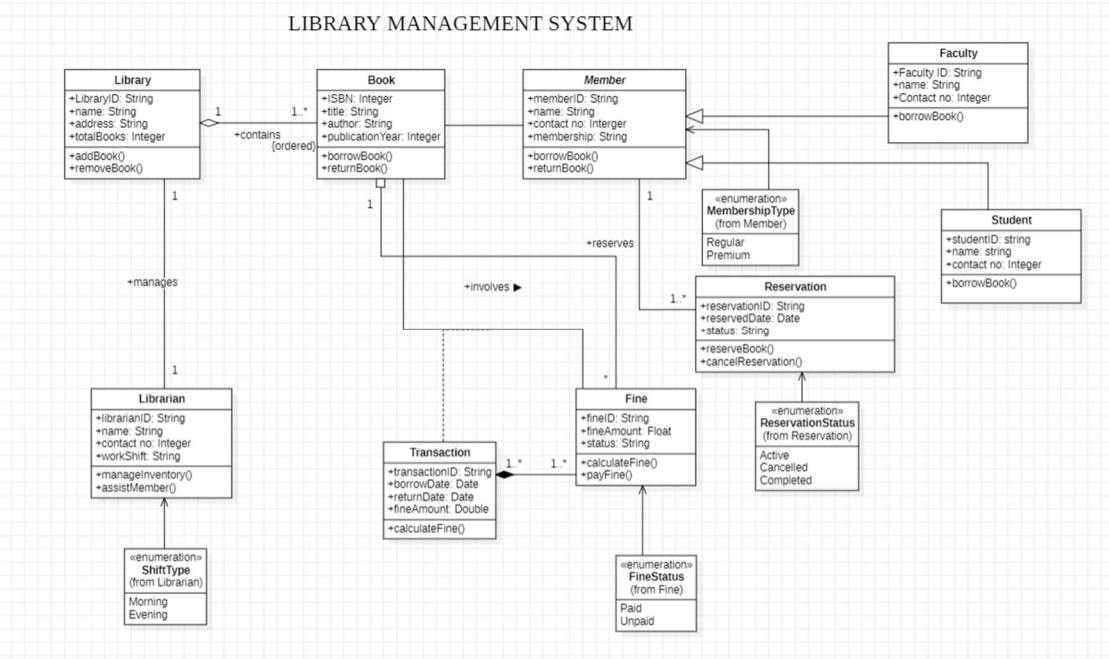
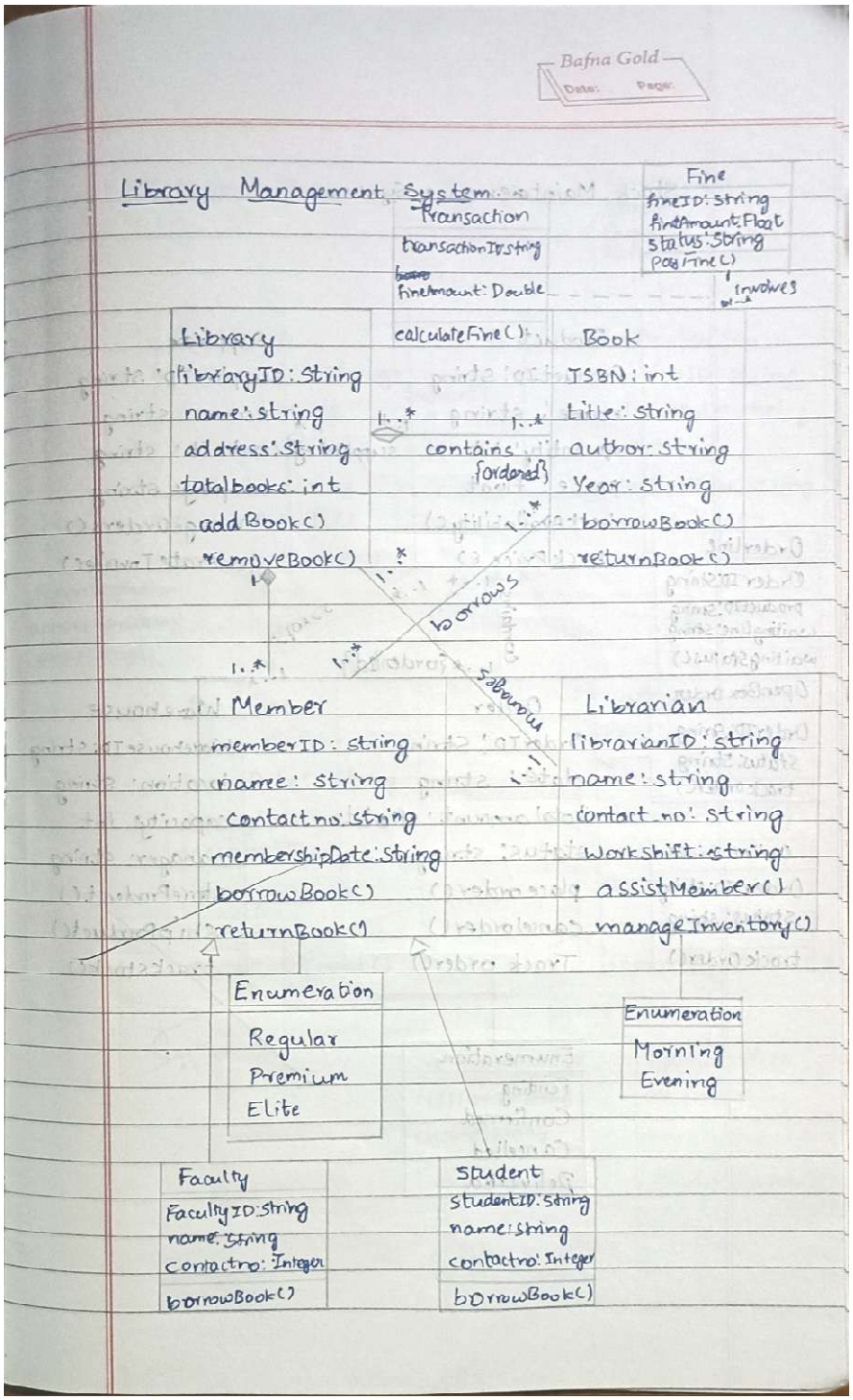


Figure 3.1: Class Diagram

The Library Management System Class Diagram outlines the structure and relationships essential for efficient library operations. The Library class serves as the central entity, managing books, with methods for adding and removing them. Books are represented by the Book class, which tracks attributes like ISBN, title, author, and publication year, and includes methods for borrowing and returning.

Members, classified as RegularMember, PremiumMember, or EliteMember, interact with the system to borrow, return, and reserve books. Borrowing activities are managed through the BorrowTransaction class, which records transaction details, including borrow and return dates, and calculates fines through the Fine class if applicable. Reservations are handled by the Reservation class, which tracks reservation details and links to borrow transactions.

The Librarian class oversees inventory management and assists members, operating in shifts defined by ShiftType enumeration. Enumerations like MembershipType (e.g., Regular, Premium) and ReservationStatus (e.g., Active, Completed) refine the system's functionality. These interconnected components ensure seamless library operations, member satisfaction, and proper resource utilization.


Bafna Gold
Date: Page:
Library Management System
Transaction
transactionID: string
fineAmount: Double
calculateFine()
Fine
fineID: string
fineAmount: Float
status: String
payFine()
involves
Library
libraryID: String
name: string
address: String
totalbooks: int
addBook()
removeBook()
1..*
contains
(ordered)
Book
ISBN: int
title: string
author: string
year: string
borrowBook()
returnBook()
borrows
manages
Member
memberID: string
namee: string
contactno: string
membershipDate: String
borrowBook()
returnBook()
Librarian
librarianID: string
name: string
contact_no: string
workshift: string
assistMember()
manageInventory()
Enumeration
Regular
Premium
Elite
Enumeration
Morning
Evening
Faculty
FacultyID: string
name: string
contactno: Integer
borrowBook()
Student
studentID: string
name: string
contactno: Integer
borrowBook()

STATE DIAGRAM

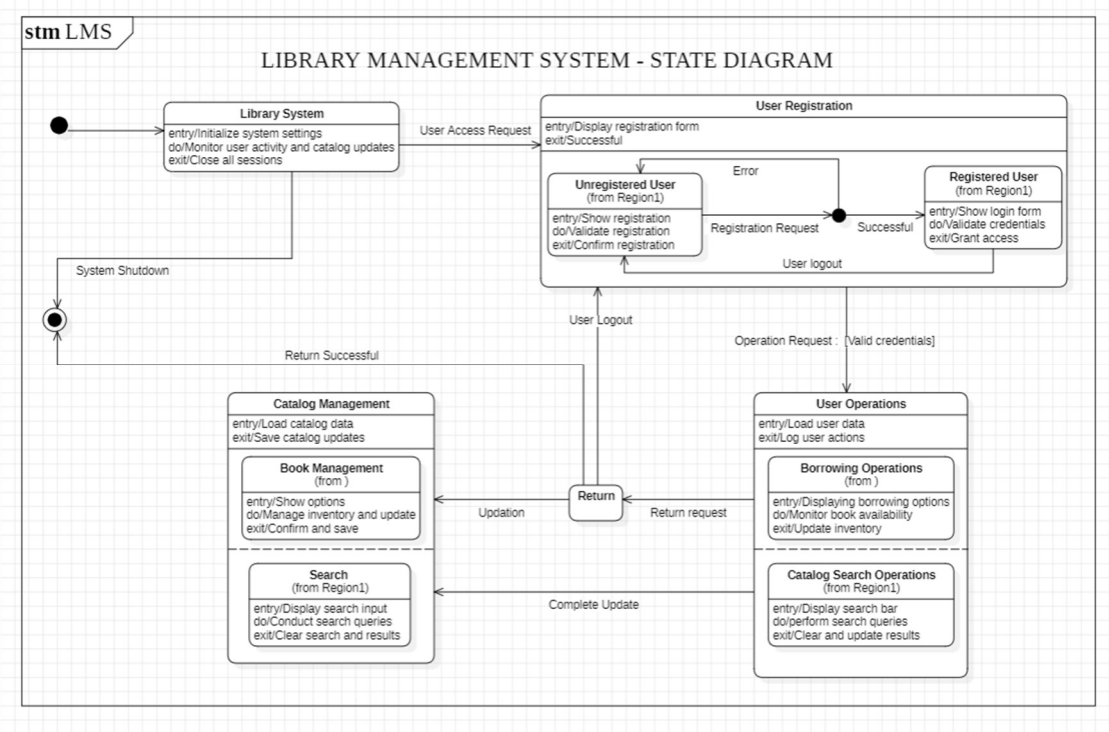


Figure 3.2: State Diagram

The Library Management System State Diagram offers a structured overview of the system's functionality, highlighting its states, transitions, and key activities. The initial state, Library System, initializes system settings and awaits user interaction. From here, users may proceed to User Registration, where new users register, transitioning from Unregistered User to Registered User after successful validation and confirmation.

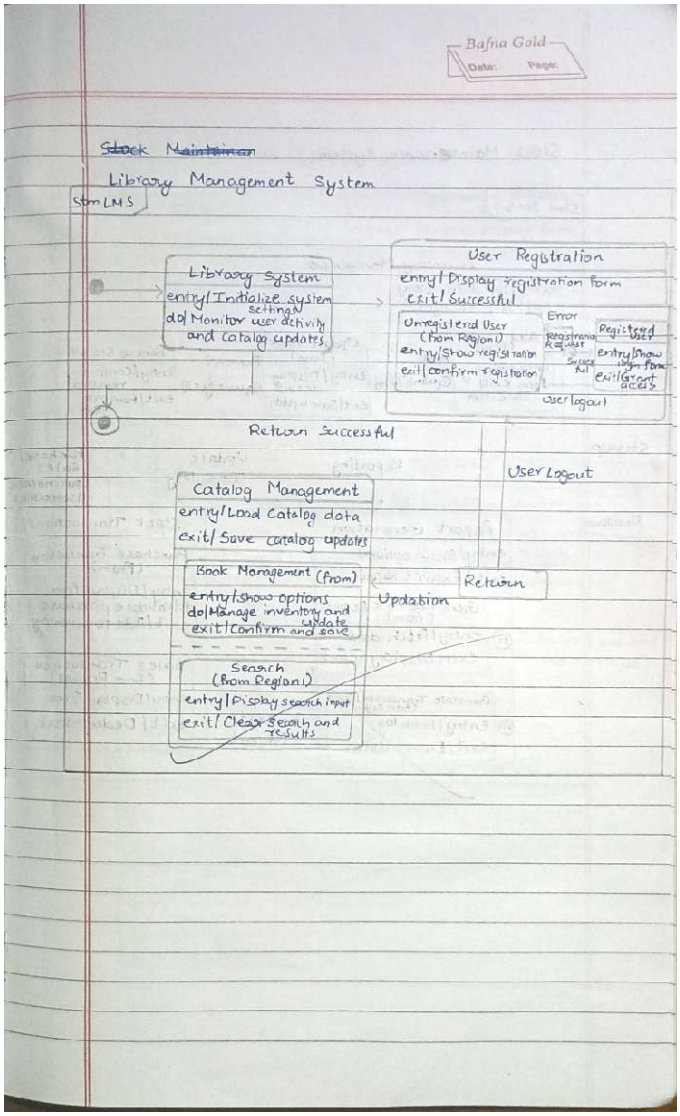
In the Registered User state, users log in by providing credentials. Successful validation grants access, enabling transitions to states like Catalog Management, where library data is managed by adding, removing, or updating books, and Book Management, which handles borrowing, returning, and reserving books. Users can also enter the Search state to explore the catalog through query-based operations.

The User Operations state allows registered users to borrow, return, or check account details. The Return state handles the book return process, updating inventory upon successful completion. Errors such as invalid credentials or failed operations direct the system to the Error state, where corrective actions can be taken.

Transitions like User Access Request, Registration Request, Operation Request, and Return Successful facilitate seamless movement between states. Activities

in each state include displaying forms, validating data, managing inventory, and saving updates, ensuring the system operates efficiently.

This state diagram effectively captures the core functionalities and user interactions within the library system, providing a clear framework for managing users, books, and catalog data.



USE CASE DIAGRAM

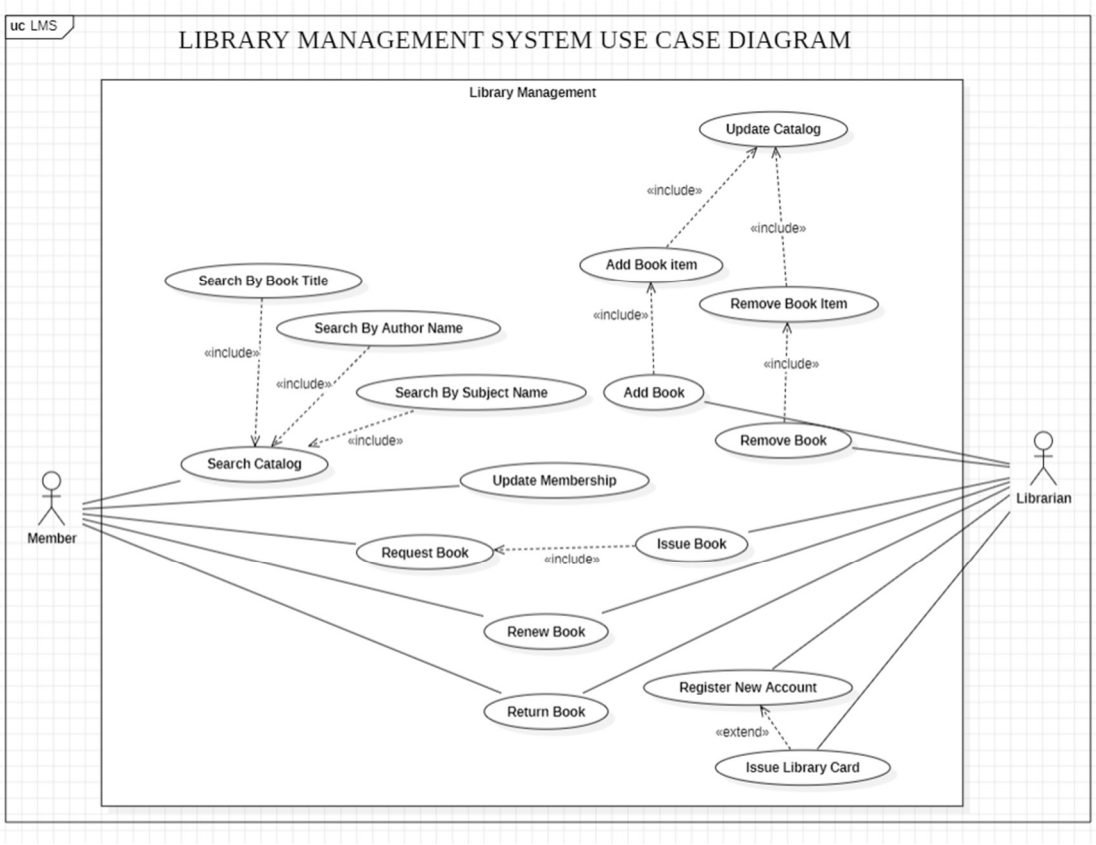
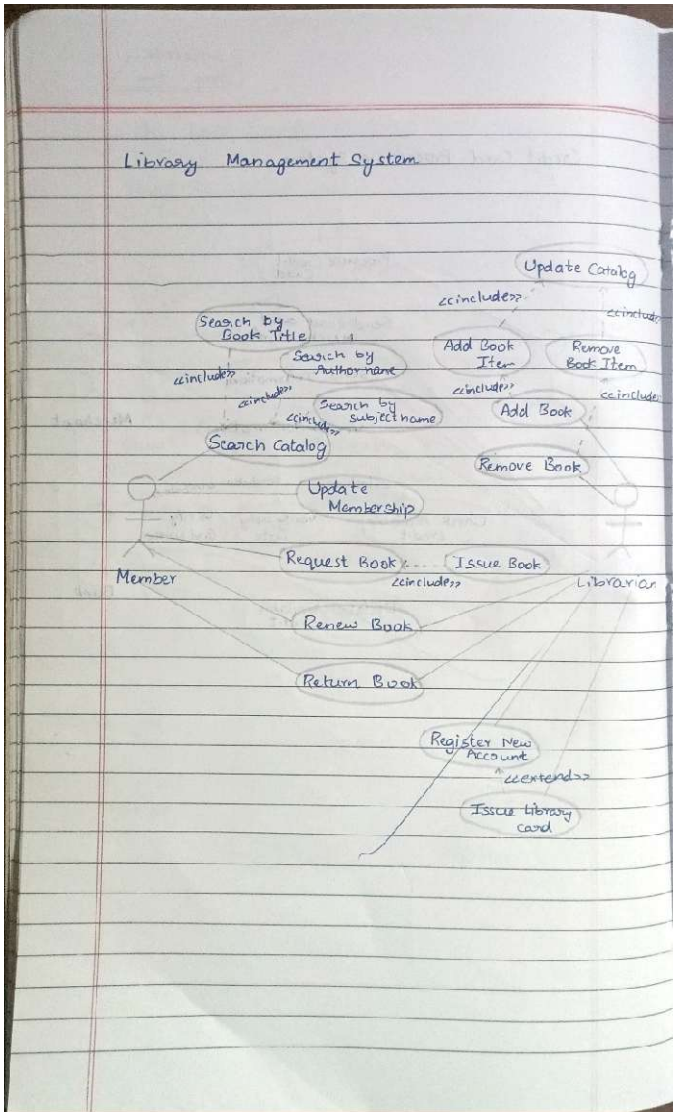


Figure 3.3: Use Case Diagram

The Library Management System involves two primary actors: Member and Librarian. Members, including Regular, Elite, and Premium types, interact with the library system to access various services such as borrowing books, renewing loans, and making requests. Librarians manage the library's operations, including maintaining the catalog, issuing books, and updating membership details. The Library Management use case encompasses tasks like updating the catalog, adding or removing books, and editing book details, which are included in sub-use cases such as Add Book Item and Remove Book Item.

Key use cases include Search Catalog, which is further divided into Search By Book Title, Search By Author Name, and Search By Subject Name, as well as tasks like Request Book, Renew Book, and Return Book. Register New Account and Issue Library Card are additional functions that librarians handle to onboard new members. The diagram also highlights the relationships between use cases, such as the Includes relationship, where one use case depends on others, and Inheritance, where more

specific member types inherit the privileges of the general Member actor. This structure allows for a streamlined representation of the library's operations and interactions.



SEQUENCE DIAGRAM

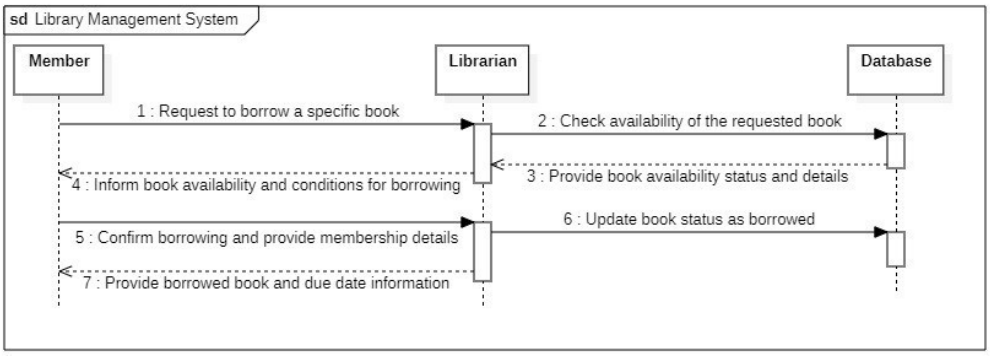
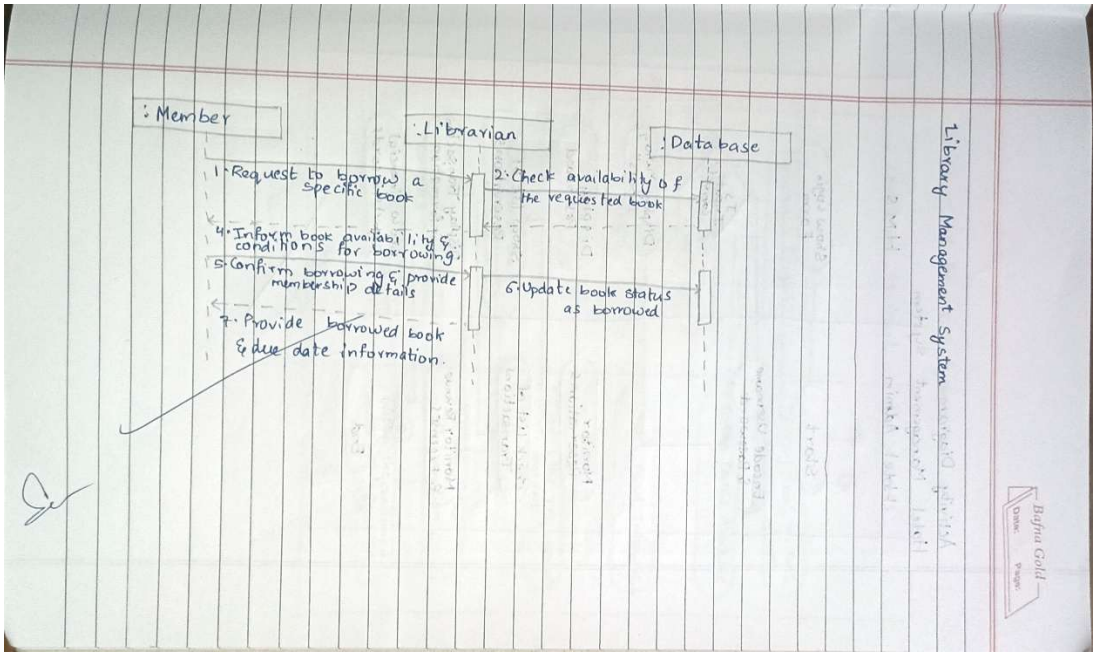


Figure 3.4: Sequence Diagram

This sequence diagram illustrates the process of borrowing and returning a book in a library system. The Customer initiates the borrowing process by requesting a book, which the Library System checks for availability in the Inventory. Once the book is found and available, a Transaction object is created to manage the borrowing process. The Library System then requests the Inventory to lend the book, updating the inventory records and informing the Customer of the successful borrowing. After the Customer returns the book, the Library System updates the inventory, and the process concludes with a successful return notification.

The diagram emphasizes the interaction between the Customer, Library System, Inventory, and Book objects, showing the flow of requests and responses. It also tracks the creation and removal of the Transaction object, ensuring proper management of the borrowing and returning process.



ACTIVITY DIAGRAM

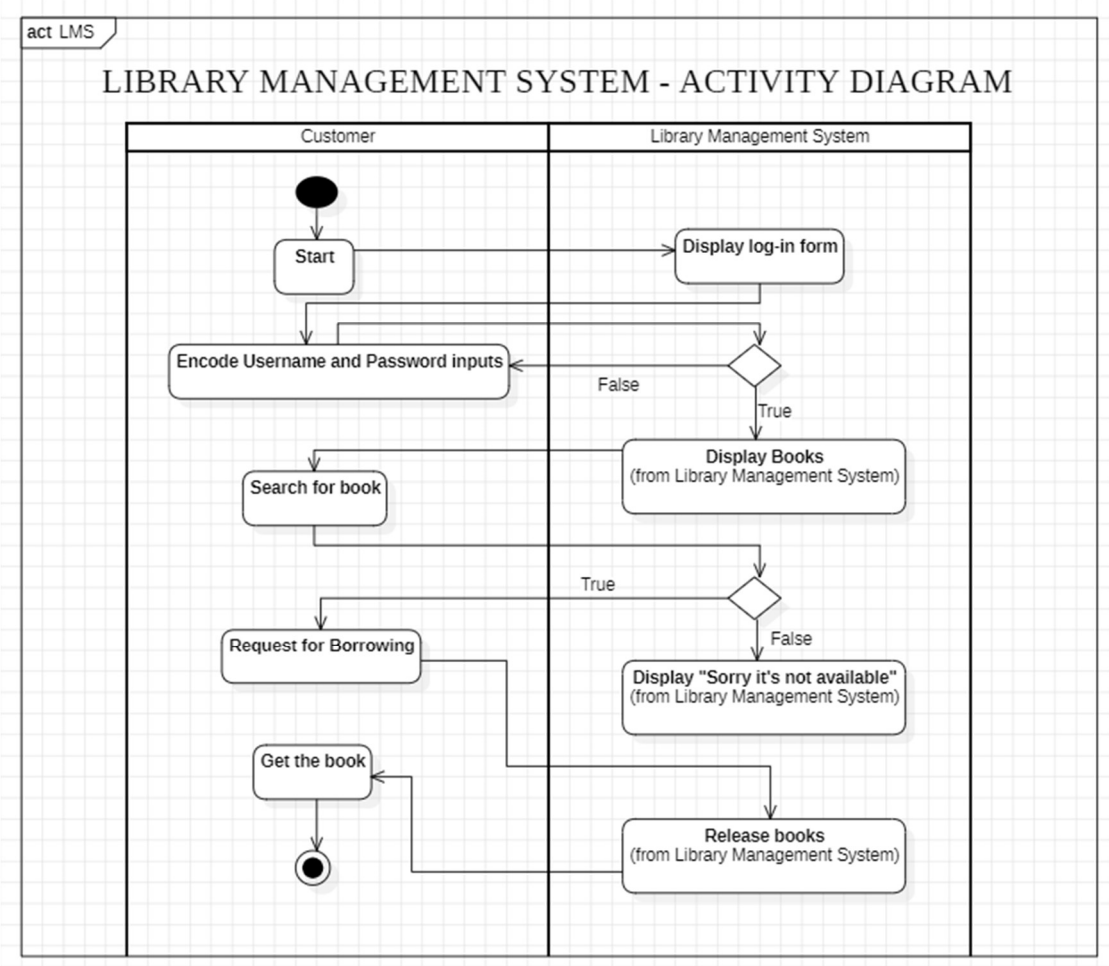
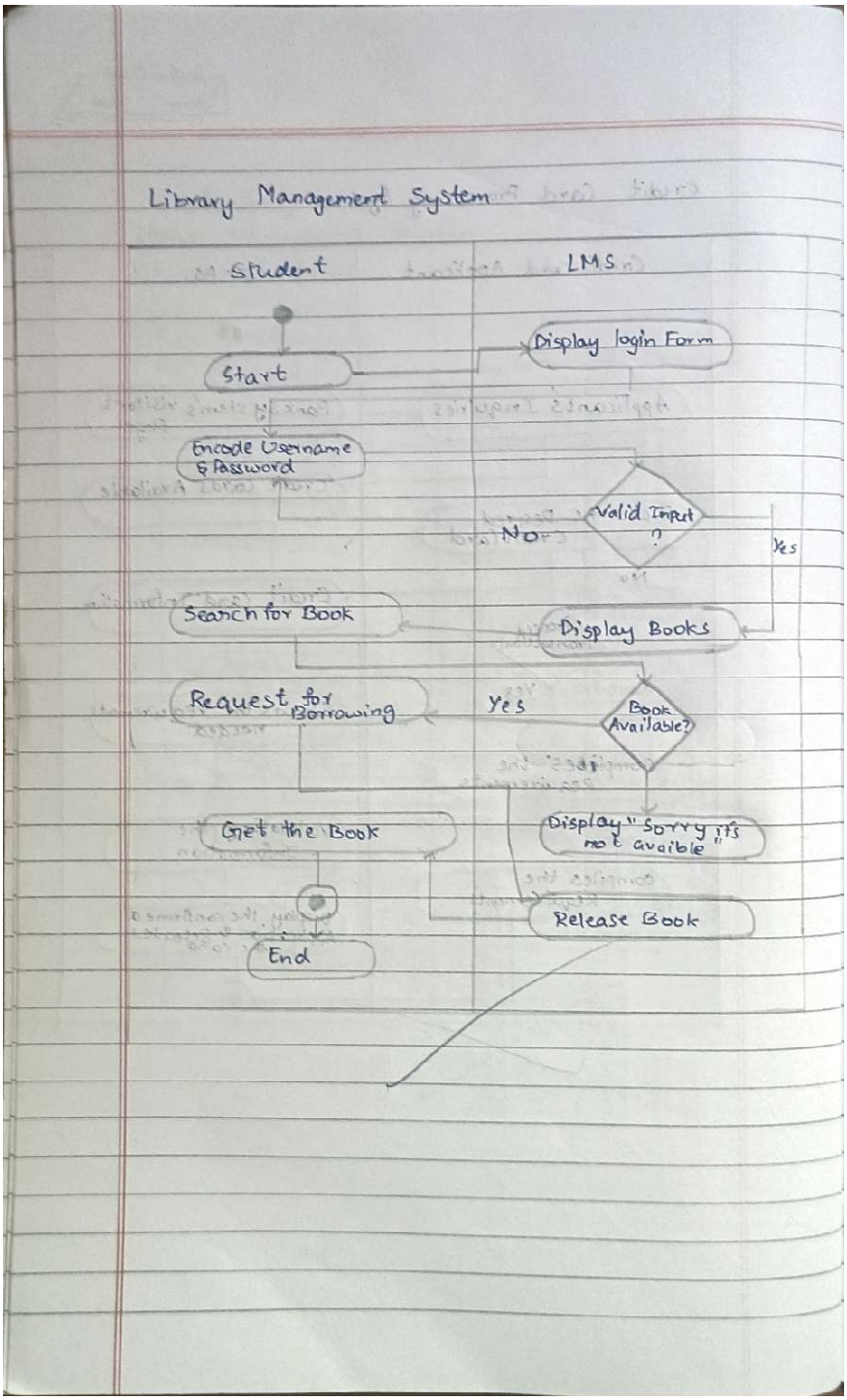


Figure 3.5: Activity Diagram

In this swimlane diagram, the actions are divided between the Customer and the Library Management System.

In the Customer swimlane, the user begins by interacting with the system in its initial state. The user then inputs their login credentials, searching for a specific book in the library catalog. Once the desired book is found, the user requests to borrow it and ultimately receives the book from the library.

In the Library Management System swimlane, the system first displays the login form to the user. After successful login, it shows the list of available books. If the requested book is unavailable, the system provides a message informing the user of its unavailability. Lastly, the system handles the process of releasing the borrowed books back into the catalog once returned by the customer. This diagram highlights the interaction flow between the user and the library system during the borrowing process.


Library Management System
Student
LMS
Start
Display login Form
Encode Username & Password
Valid Input ?
No
Yes
Search for Book
Display Books
Request for Borrowing
Yes
Book Available?
Get the Book
Display "Sorry it's not avaible"
Release Book
End

CHAPTER – 4

STOCK MAINTENANCE SYSTEM

PROBLEM STATEMENT

The Stock Maintenance System is a robust solution developed to monitor and manage inventory levels, track stock movement, and ensure timely replenishments. Traditional or manual inventory systems are prone to errors, leading to overstocking, stockouts, and financial losses. This system automates inventory tracking, integrates with procurement and sales platforms, and generates alerts for low stock levels. By optimizing stock management, it helps businesses reduce costs, improve operational efficiency, and meet customer demands consistently.

- What is the system for?

The Stock Maintenance System is for tracking and managing inventory, monitoring stock levels, generating restocking alerts, and maintaining purchase and sales records. It ensures smooth operations in supply chain and retail management.

- Where is it used?

It is used in warehouses, retail stores, manufacturing plants, distribution hubs, and e-commerce platforms to handle inventory efficiently.

- Who is it for?

The system is for inventory managers, procurement teams, supply chain professionals, and business owners who need to monitor stock levels and ensure product availability.

- When is it required?

It is required in any scenario where inventory management is critical, such as during high sales periods, product launches, or time-sensitive manufacturing operations.

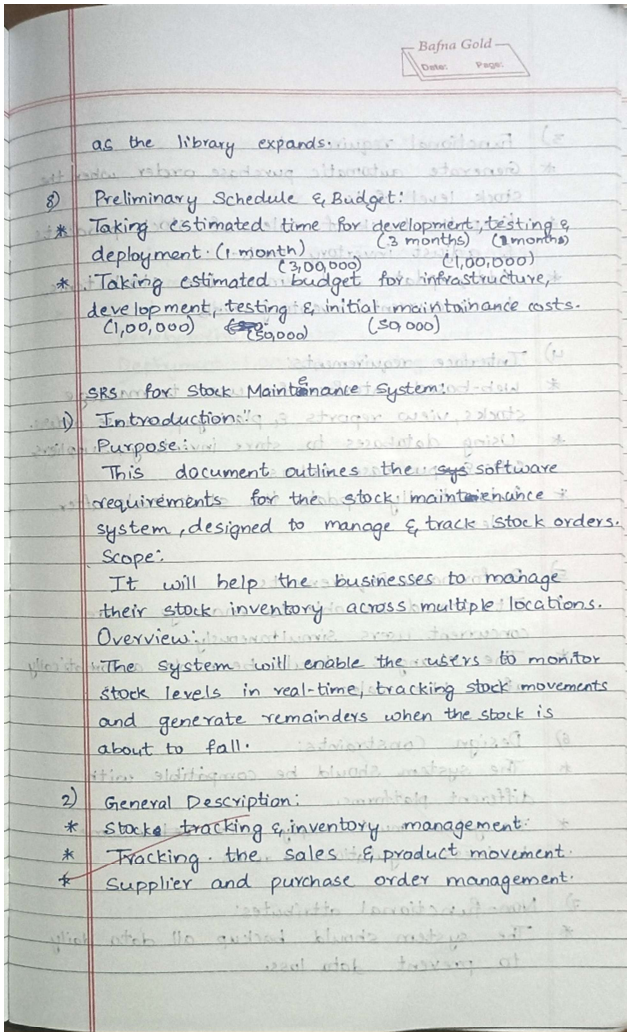
- Why is it needed?

It is needed to optimize inventory levels, prevent stockouts or overstocking, reduce storage costs, and ensure timely product availability. It also helps maintain accurate records for auditing and forecasting.

- How is it done?

The system uses inventory management software that integrates with point-of-sale systems, procurement modules, and sales platforms. Features like barcode scanning, automated alerts, and real-time reporting ensure efficient stock handling.

SOFTWARE REQUIREMENTS SPECIFICATION



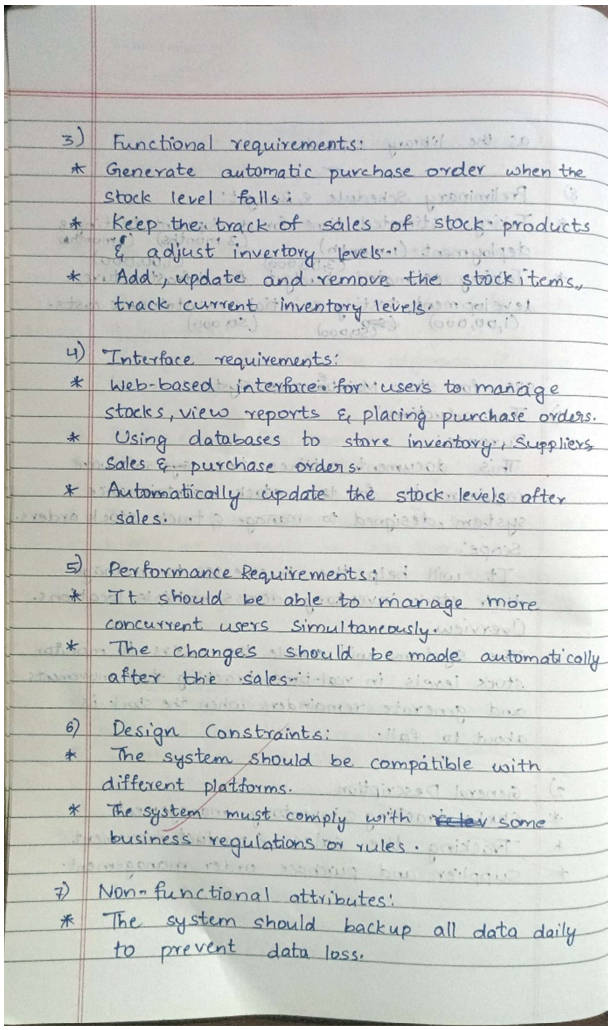
as the library expands.

8) Preliminary Schedule & Budget:
* Taking estimated time for development, testing & deployment. (1 month) (3 months) (1 months)
* Taking estimated budget for infrastructure, (3,00,000) (1,00,000)
development, testing & initial maintainance costs.
(1,00,000) (50,000) (50,000)

SRS for Stock Maintainance System:

1) Introduction:
Purpose:
This document outlines the ~~sys~~ software requirements for the stock maintainance system, designed to manage & track stock orders.
Scope:
It will help the businesses to manage their stock inventory across multiple locations.
Overview:
The system will enable the users to monitor stock levels in real-time, tracking stock movements and generate remainders when the stock is about to fall.

2) General Description:
* Stock tracking & inventory management.
* Tracking the sales & product movement.
* Supplier and purchase order management.



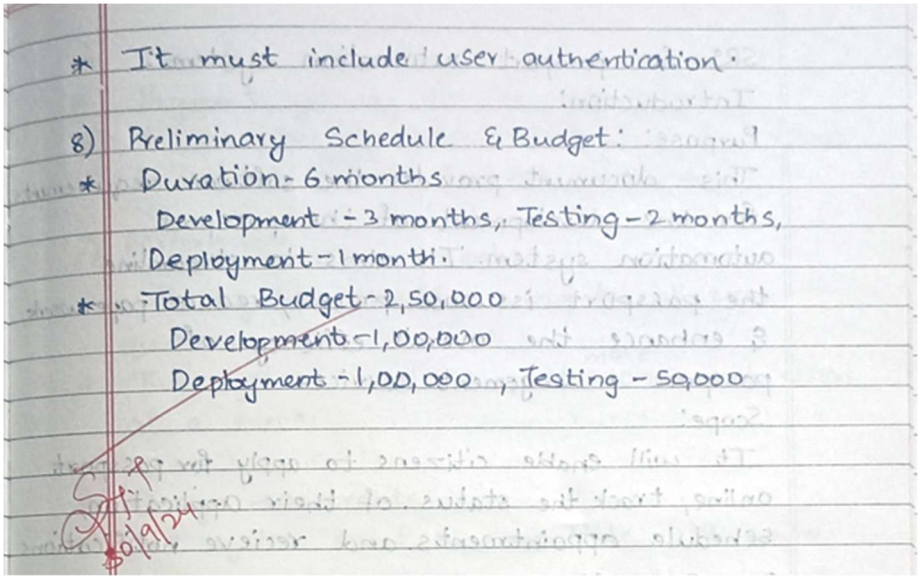
3) Functional requirements:
* Generate automatic purchase order when the stock level falls.
* Keep the track of sales of stock products & adjust invertory levels.
* Add, update and remove the stock items, track current inventory levels.

4) Interface requirements:
* Web-based interface for users to manage stocks, view reports & placing purchase orders.
* Using databases to store inventory, Suppliers Sales & purchase orders.
* Automatically update the stock levels after sales.

5) Performance Requirements:
* It should be able to manage more concurrent users simultaneously.
* The changes should be made automatically after the sales.

6) Design Constraints:
* The system should be compatible with different platforms.
* The system must comply with ~~rele~~ some business regulations or rules.

7) Non-functional attributes:
* The system should backup all data daily to prevent data loss.



* It must include user authentication.

8) Preliminary Schedule & Budget:
* Duration - 6 months
Development - 3 months, Testing - 2 months,
Deployment - 1 month.
* Total Budget - 2,50,000
Development - 1,00,000
Deployment - 1,00,000, Testing - 50,000

10/9/24

UML DIAGRAMS

CLASS DIAGRAM

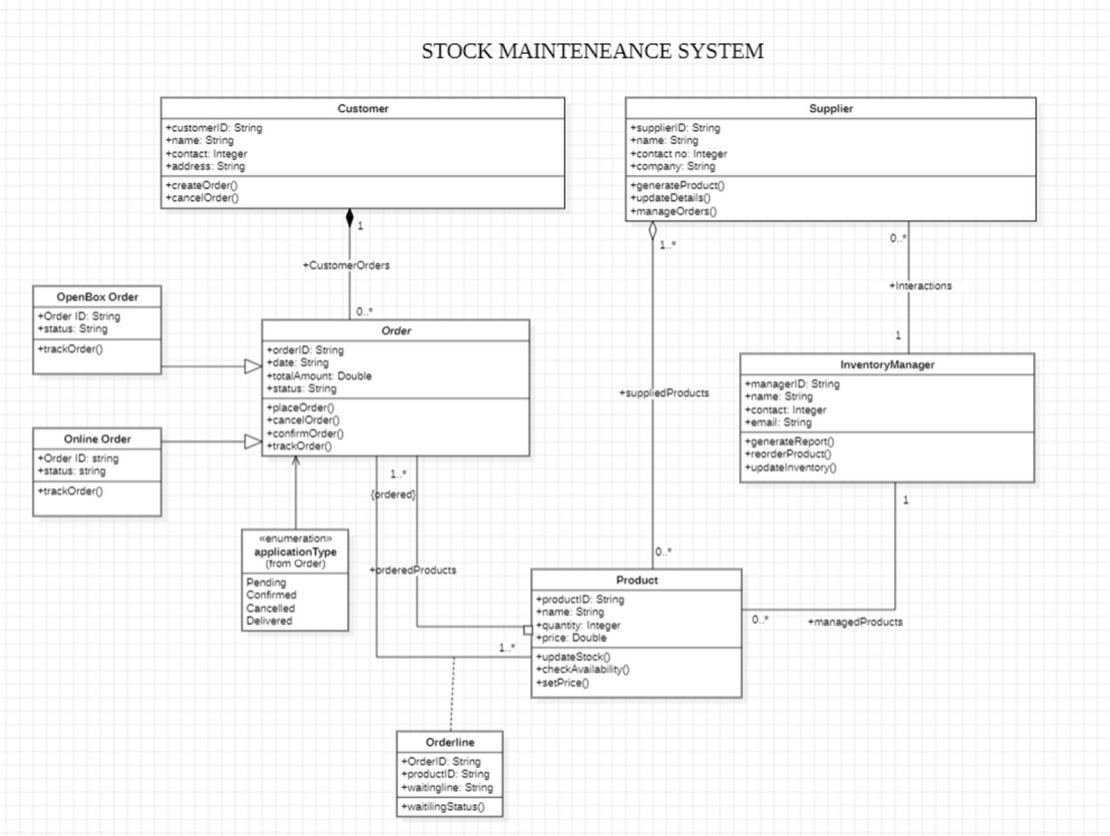
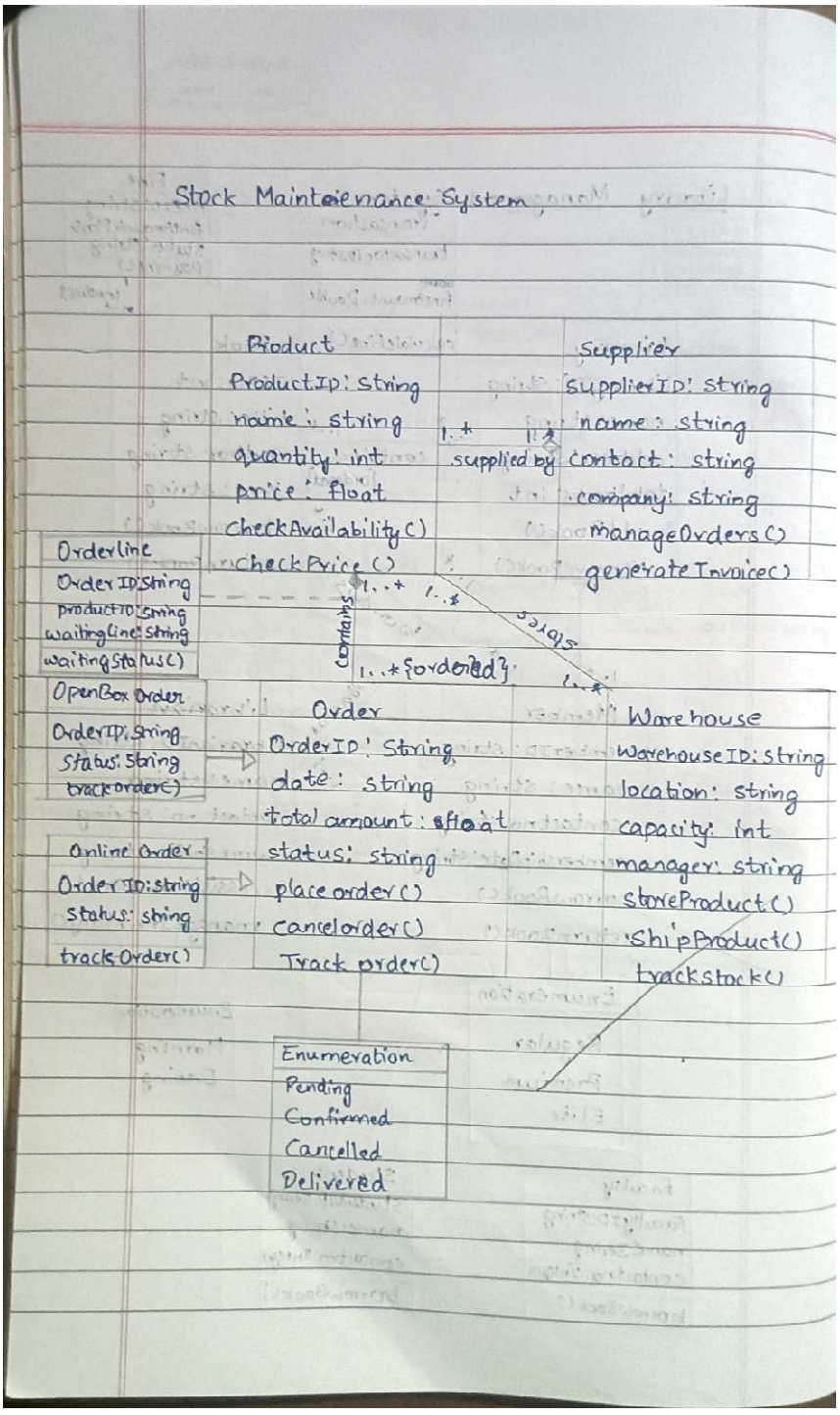


Figure 4.1: Class Diagram

The Stock Maintenance System Class Diagram outlines the core entities and their relationships to efficiently manage products, orders, and inventory. The Customer class represents buyers, enabling them to create, cancel, and update orders. Orders are handled through the Order class, which includes specialized types such as OnlineOrder and OpenBoxOrder. Each order consists of multiple OrderLine items, linked to individual Product entities that store details like quantity, price, and availability.

Suppliers, represented by the Supplier class, provide products and manage updates. The InventoryManager class oversees stock, monitors orders, generates reports, and ensures timely restocking. Customers' orders are aggregated in the CustomerOrders class for better tracking. The system leverages enumerations such as OrderStatus (e.g., Pending, Confirmed, Delivered) to track the lifecycle of orders. Associations between these entities ensure smooth workflows, from product supply to order completion, providing an optimized and seamless stock management process.


Stock Maintenance System
Product
ProductID: String
name: string
quantity: int
price: float
CheckAvailability()
checkPrice()
Supplier
SupplierID: String
name: string
contact: string
company: string
manageOrders()
generateInvoice()
1..*
supplied by
1..1
contains
1..*
1..*
stores
1..*
{ordered}
Orderline
OrderID: String
productID: String
waitingLine: String
waitingStatus()
OpenBox Order
OrderID: String
Status: String
trackorder()
Online Order
OrderID: String
Status: string
trackOrder()
Order
OrderID: String
date: string
total amount: float
status: string
placeorder()
cancelorder()
Track order()
Warehouse
WarehouseID: String
location: string
capacity: int
manager: string
storeProduct()
ShipProduct()
trackstock()
Enumeration
Pending
Confirmed
Cancelled
Delivered

STATE DIAGRAM

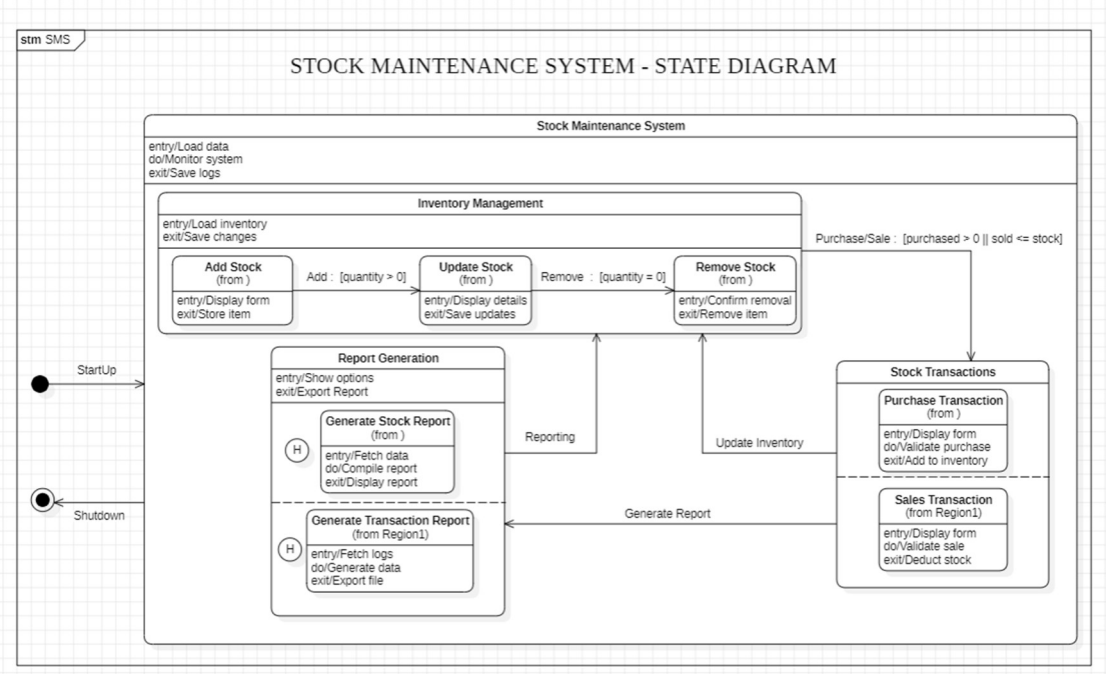


Figure 4.2: State Diagram

The Stock Maintenance System State Diagram provides a structured representation of the key states and transitions in managing stock and inventory processes. The system begins in the Stock Maintenance System state, where it initializes settings, monitors activity, and loads necessary data.

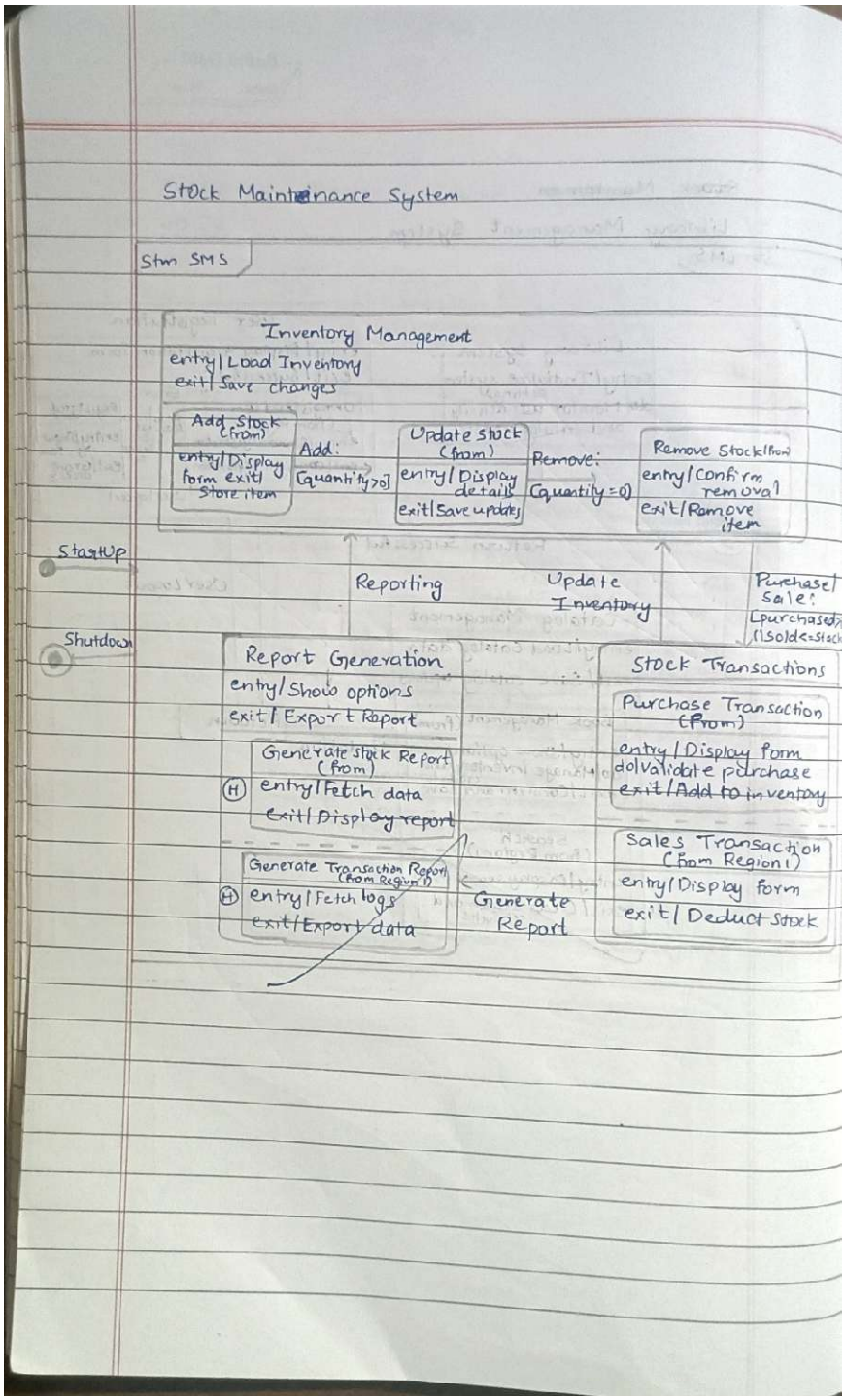
The transition to Inventory Management occurs when the user performs inventory-related tasks. This state encapsulates specific operations like Add Stock, Update Stock, and Remove Stock. In the Add Stock state, new stock items are added, while in the Update Stock state, the quantity of existing items is modified. The Remove Stock state is responsible for deleting items from the inventory. Activities in these states include displaying forms, storing items, saving updates, and confirming removals.

The Report Generation state facilitates the creation of detailed reports. It includes two substates: Generate Stock Report and Generate Transaction Report. In these states, data is fetched, compiled into reports, and displayed or exported as needed.

The Stock Transactions superstate encompasses two primary states: Purchase Transaction and Sales Transaction. During Purchase Transaction, purchases are validated, and inventory is updated by adding items. In Sales Transaction, sales are validated, and stock quantities are deducted accordingly. Activities involve displaying transaction forms, validating processes, and adjusting inventory data.

Transitions such as Add Stock, Update Stock, Remove Stock, Generate Stock Report, and Generate Transaction Report facilitate movement between states based on user actions. Core activities across states include loading inventory, saving changes, fetching data, and exporting reports.

This diagram provides a comprehensive view of the Stock Maintenance System, covering inventory operations, transaction processing, and reporting functionalities, ensuring streamlined and efficient management.



USE CASE DIAGRAM

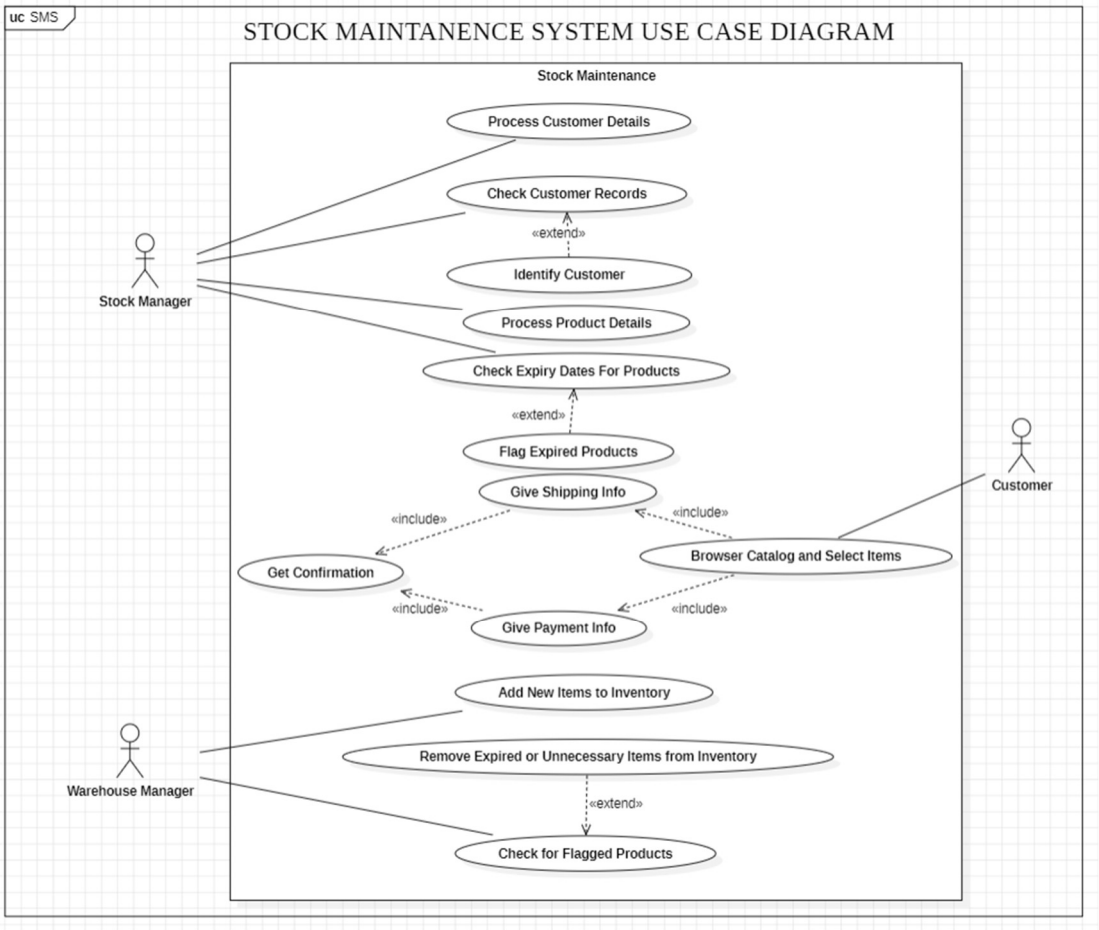
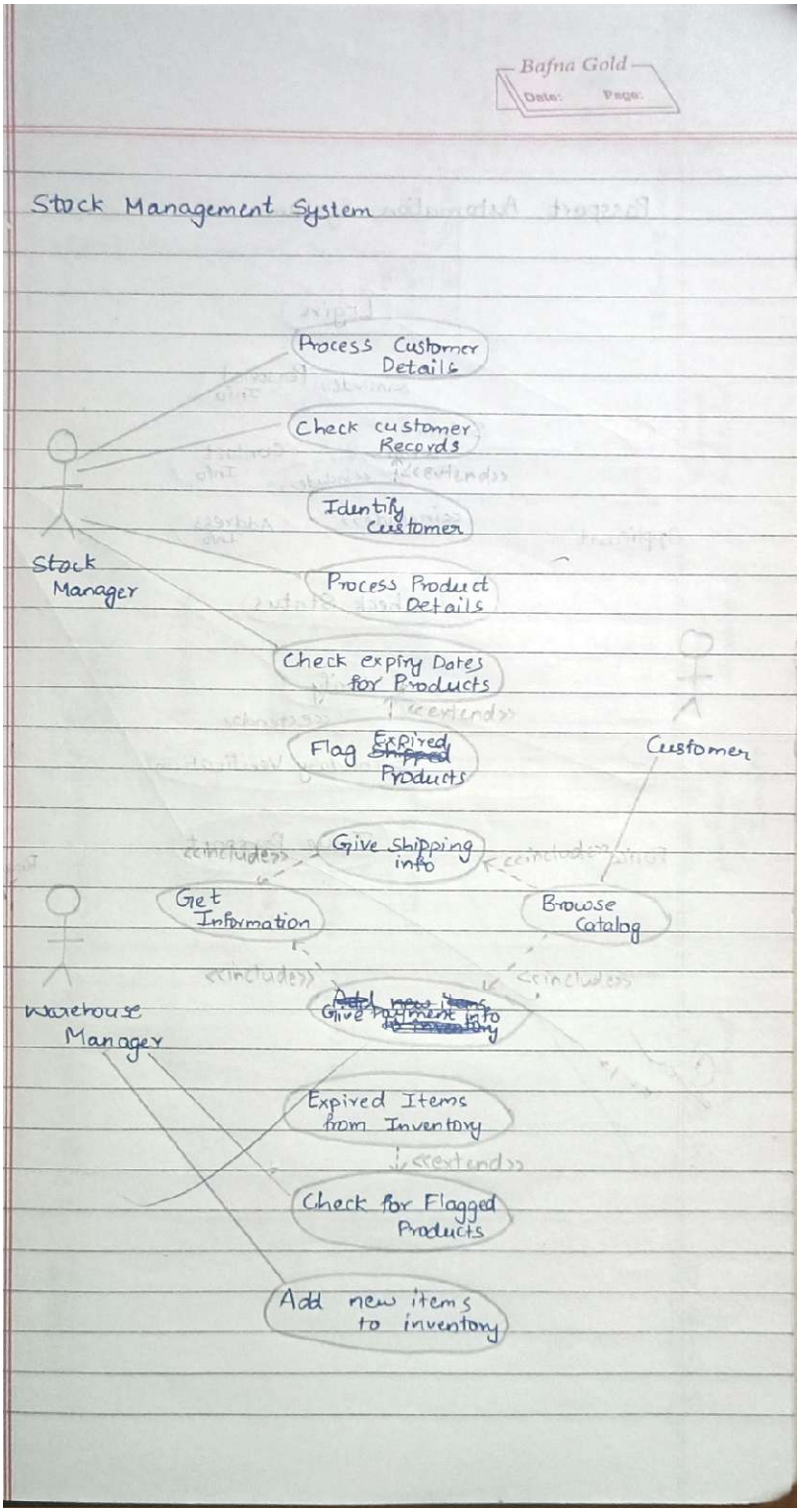


Figure 4.3: Use Case Diagram

The Stock Maintenance System involves several key actors: the Stock Manager, Control Manager, Warehouse Manager, Inventory Manager, and Customer. These actors interact with various use cases to manage stock, process customer orders, maintain accurate inventory records, and handle the flow of products. The main use case, Stock Maintenance, includes managing customer and product details, such as processing customer information and tracking product inventory. Extensions like Check Customer Records and Identify Regular Customer allow for enhanced customer management, potentially enabling loyalty programs and targeted promotions.

Key functions within the system include Give Shipping Info for providing delivery details to customers and Get Confirmation for confirming orders, which includes Give Payment Info and Browser Catalog and Select Items. The system also allows the Stock Manager and Inventory Manager to add new items to inventory, delete

expired or obsolete items, and flag products nearing expiration. The relationships between use cases, such as <<include>> and <<extend>>, help define dependencies and optional extensions to manage tasks like verifying customer records or flagging expired products. This use case diagram provides a comprehensive view of how stock management and customer interactions are integrated.



SEQUENCE DIAGRAM

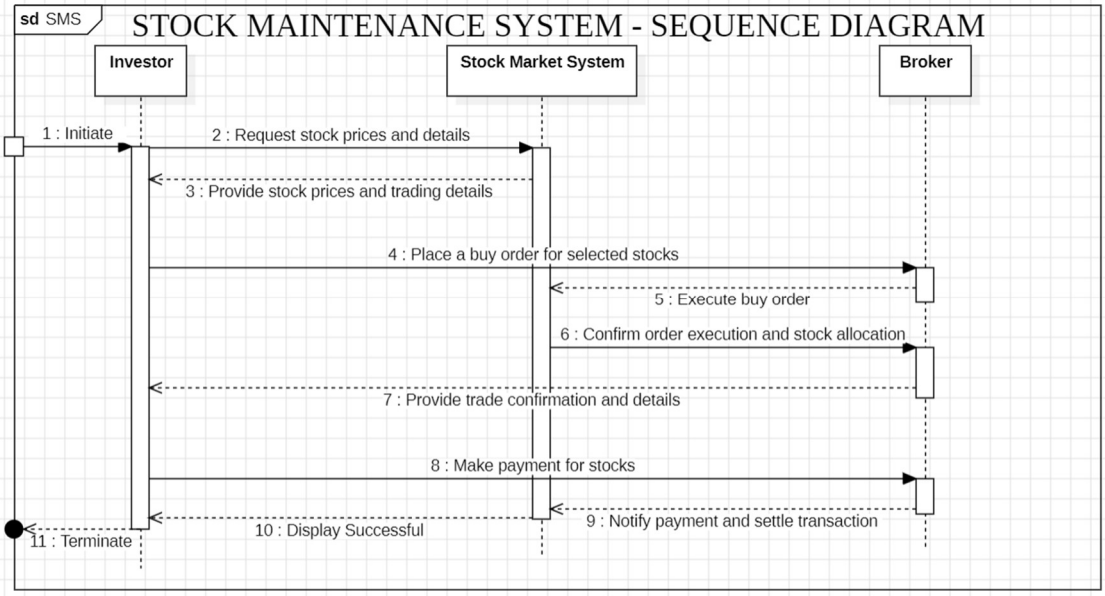
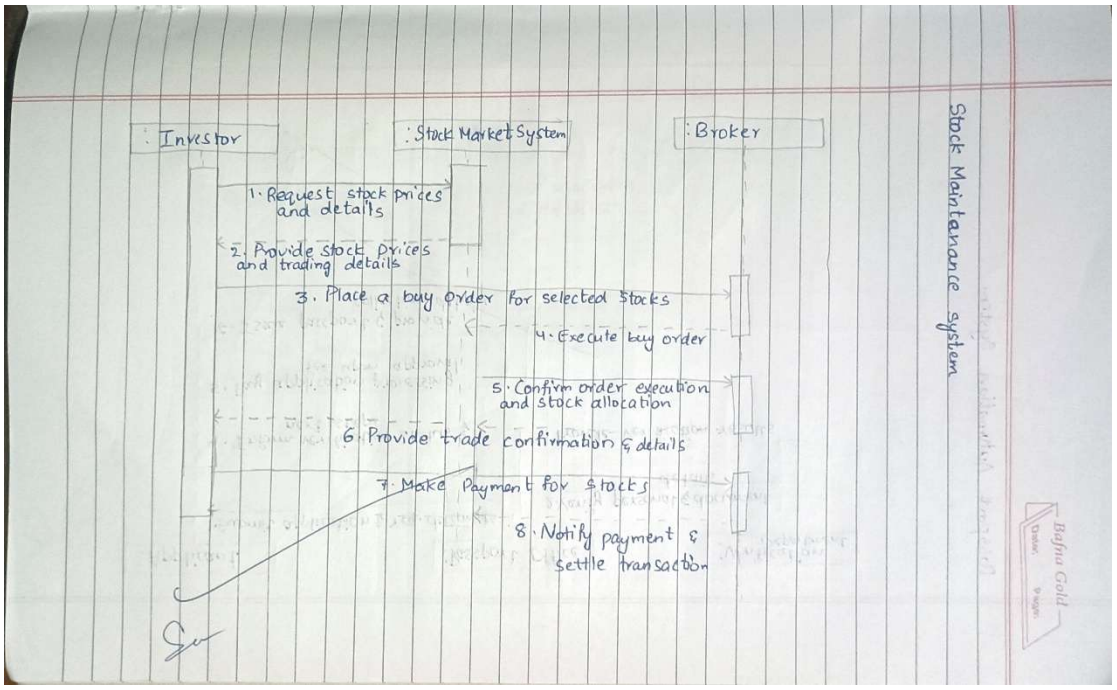


Figure 4.4: Sequence Diagram

This sequence diagram represents the flow of interactions between the Investor, Stock Market System, and Broker during a stock trading transaction. The Investor begins by requesting stock prices and trading details from the Stock Market System. After receiving the information, the Investor places a buy order through the Broker, who executes it on the Investor's behalf. The Stock Market System confirms the order execution and allocates the stocks to the Investor's account, then provides a trade confirmation. The Investor makes payment for the stocks to the Broker, who notifies the Stock Market System, and the transaction is settled successfully.

The diagram highlights the key steps of a stock purchase, including order placement, payment, and settlement. It captures the interactions between the Investor, Stock Market System, and Broker, emphasizing the flow of information and ensuring that the transaction is completed smoothly.


: Investor
: Stock Market System
: Broker
1. Request stock prices and details
2. Provide stock prices and trading details
3. Place a buy order for selected stocks
4. Execute buy order
5. Confirm order execution and stock allocation
6. Provide trade confirmation & details
7. Make Payment for stocks
8. Notify payment & settle transaction
Stock Maintenance system

ACTIVITY DIAGRAM

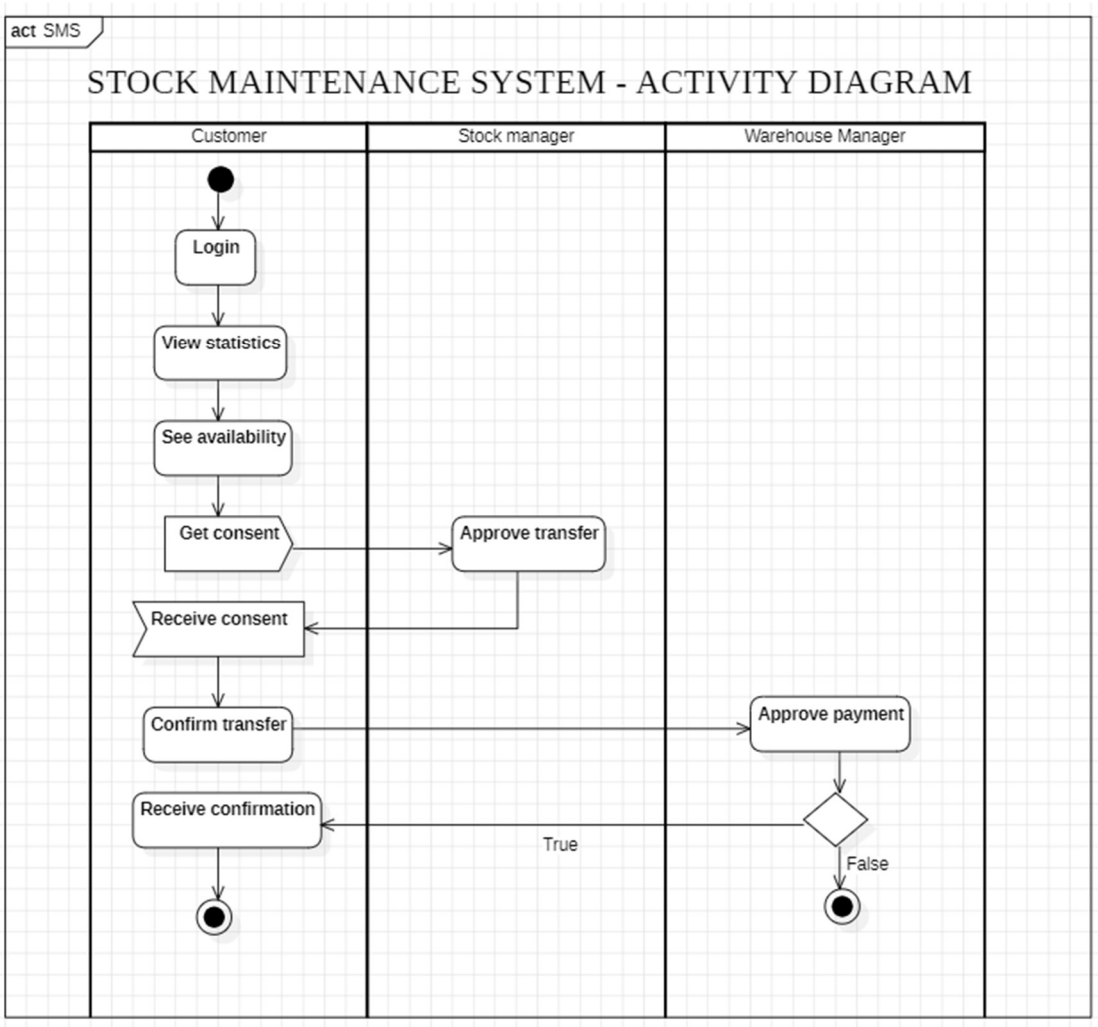
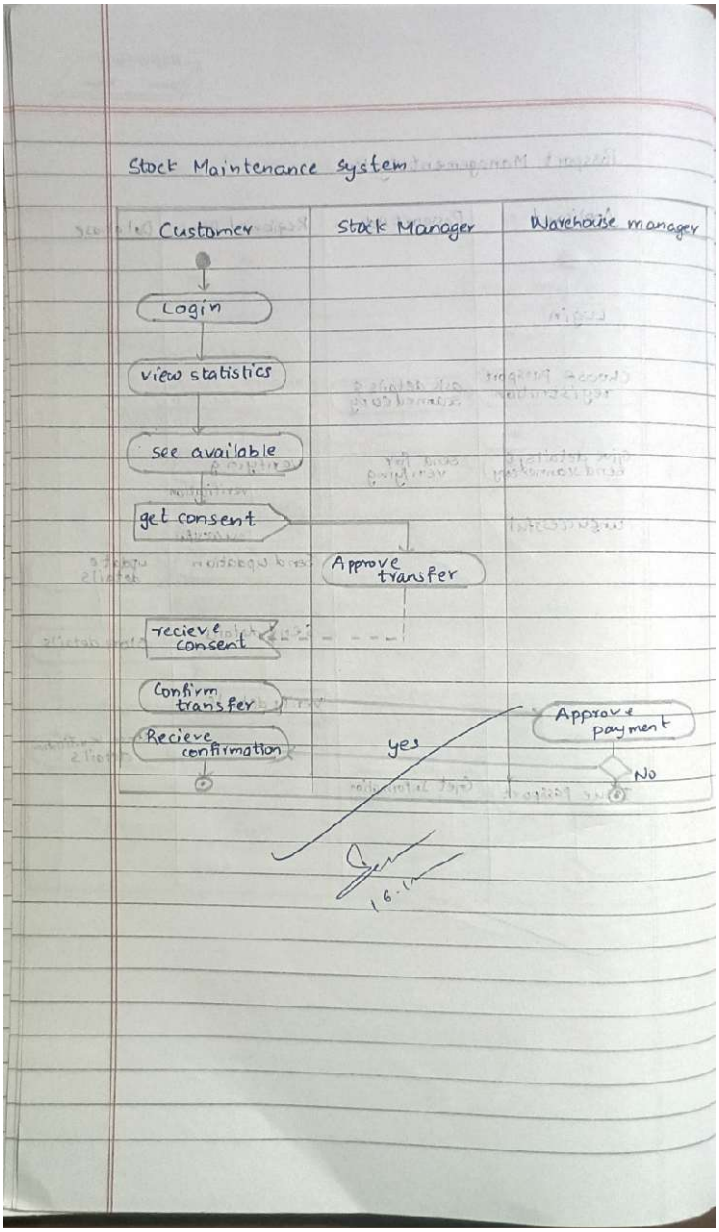


Figure 4.5: Activity Diagram

In this swimlane diagram, the interactions are divided among the customer, stock manager, and warehouse manager swimlanes.

The customer starts by logging into the system and viewing relevant stock statistics. They then check the availability of a specific stock and proceed to obtain consent for a stock transfer. Once consent is obtained from the relevant parties, the customer confirms the transfer and receives confirmation that the stock transfer has been completed.

The stock manager is responsible for approving the stock transfer request initiated by the customer. The warehouse manager then approves the payment for the stock transfer. This diagram illustrates the flow of activities across different roles involved in processing a stock transfer within the system.


Stock Maintenance system
Customer
Stock Manager
Warehouse manager
Login
view statistics
see available
get consent
Approve transfer
recieve consent
Confirm transfer
Recieve confirmation
Approve payment
yes
No

CHAPTER – 5

PASSPORT AUTOMATION SYSTEM

PROBLEM STATEMENT

The Passport Automation System is an efficient software solution designed to automate the processes involved in passport application, verification, and issuance. Traditional manual methods can result in processing delays, data inaccuracies, and increased workloads for government agencies. This system digitizes the application process, integrates with verification databases, and automates workflows to enhance accuracy and transparency. It ensures faster passport processing, improved data security, and a streamlined experience for applicants and authorities alike.

- What is the system for?

The Passport Automation System automates the application, verification, and issuance processes for passports, reducing manual effort and improving efficiency.

- Where is it used?

It is used in government passport offices, embassies, consulates, and online portals where passport-related services are provided.

- Who is it for?

It is for citizens applying for passports, government employees processing applications, and international authorities verifying passport details.

- When is it required?

It is required when individuals apply for a new passport, renew existing passports, or make updates to travel documentation. It is also necessary for managing high application volumes or ensuring compliance with global travel standards.

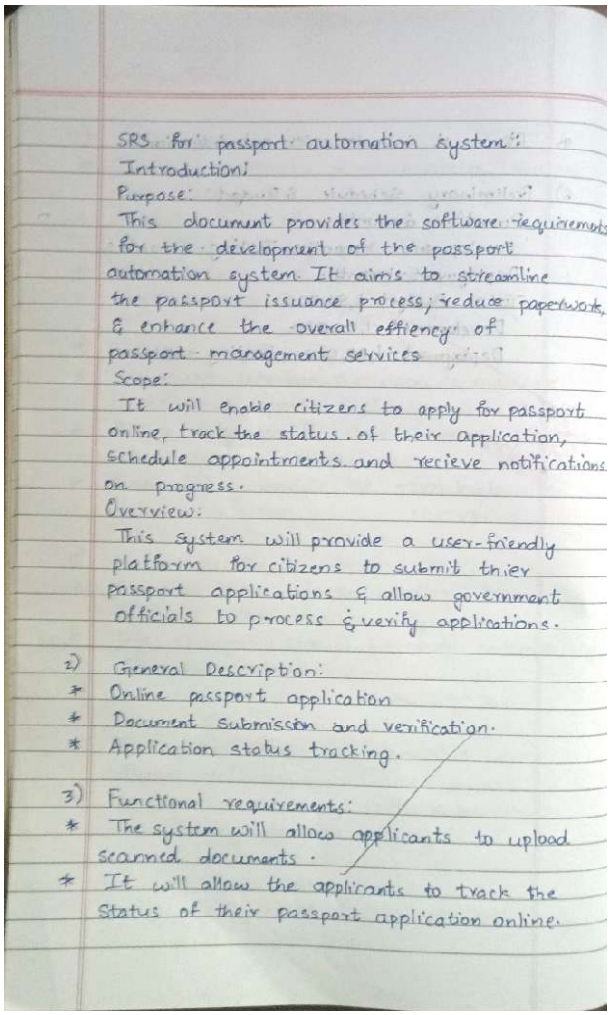
- Why is it needed?

It is needed to reduce processing delays, enhance data accuracy, improve security, and provide transparency to applicants. The system ensures faster service delivery and better coordination among agencies.

- How is it done?

The system uses online portals for application submission, integrated databases for document verification, and automated workflows for processing and tracking applications. Features like biometrics and e-passports enhance security and convenience.

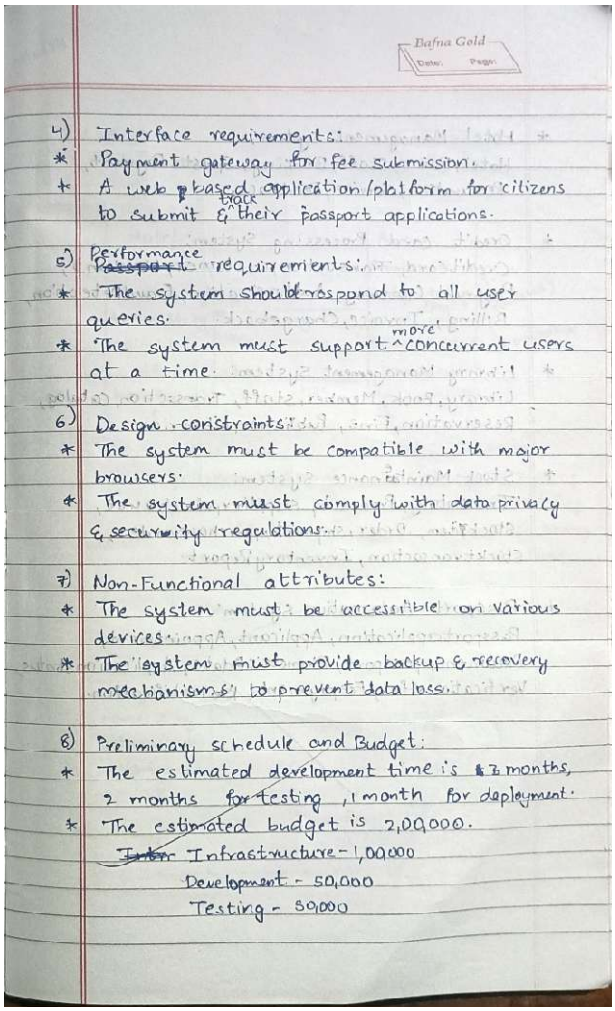
SOFTWARE REQUIREMENTS SPECIFICATION



SRS for passport automation system:
Introduction:
Purpose:
This document provides the software requirements for the development of the passport automation system. It aims to streamline the passport issuance process, reduce paperwork & enhance the overall effiency of passport management services
Scope:
It will enable citizens to apply for passport online, track the status of their application, schedule appointments and recieve notifications on progress.
Overview:
This system will provide a user-friendly platform for citizens to submit thier passport applications & allow government officials to process & verify applications.

2) General Description:
* Online passport application
* Document submission and verification.
* Application status tracking.

3) Functional requirements:
* The system will allow applicants to upload scanned documents.
* It will allow the applicants to track the status of their passport application online.



4) Interface requirements:
* Payment gateway for fee submission.
* A web based application/platform for citizens to submit & track their passport applications.

5) Performance requirements:
* The system should respond to all user queries.
* The system must support more concurrent users at a time.

6) Design constraints:
* The system must be compatible with major browsers.
* The system must comply with data privacy & security regulations.

7) Non-Functional attributes:
* The system must be accessible on various devices.
* The system must provide backup & recovery mechanisms to prevent data loss.

8) Preliminary schedule and Budget:
* The estimated development time is 12 months, 2 months for testing, 1 month for deployment.
* The estimated budget is 2,00,000.
  Infrastructure - 1,00,000
  Development - 50,000
  Testing - 50,000

UML DIAGRAMS

CLASS DIAGRAM

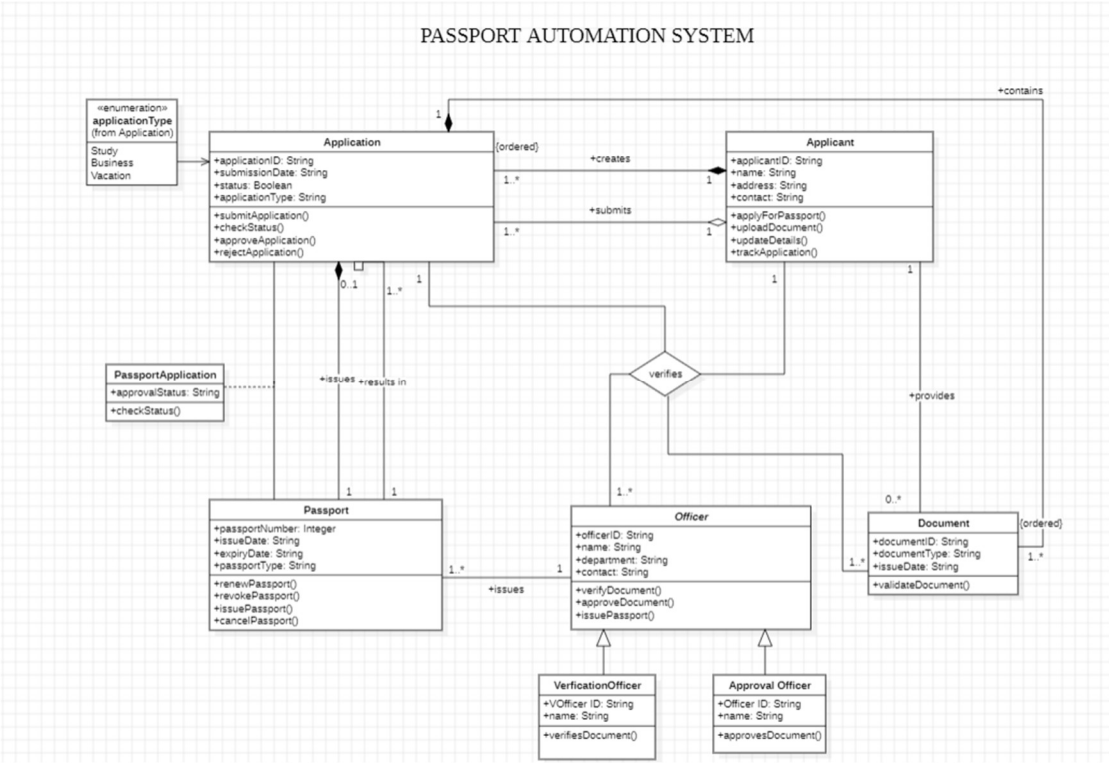
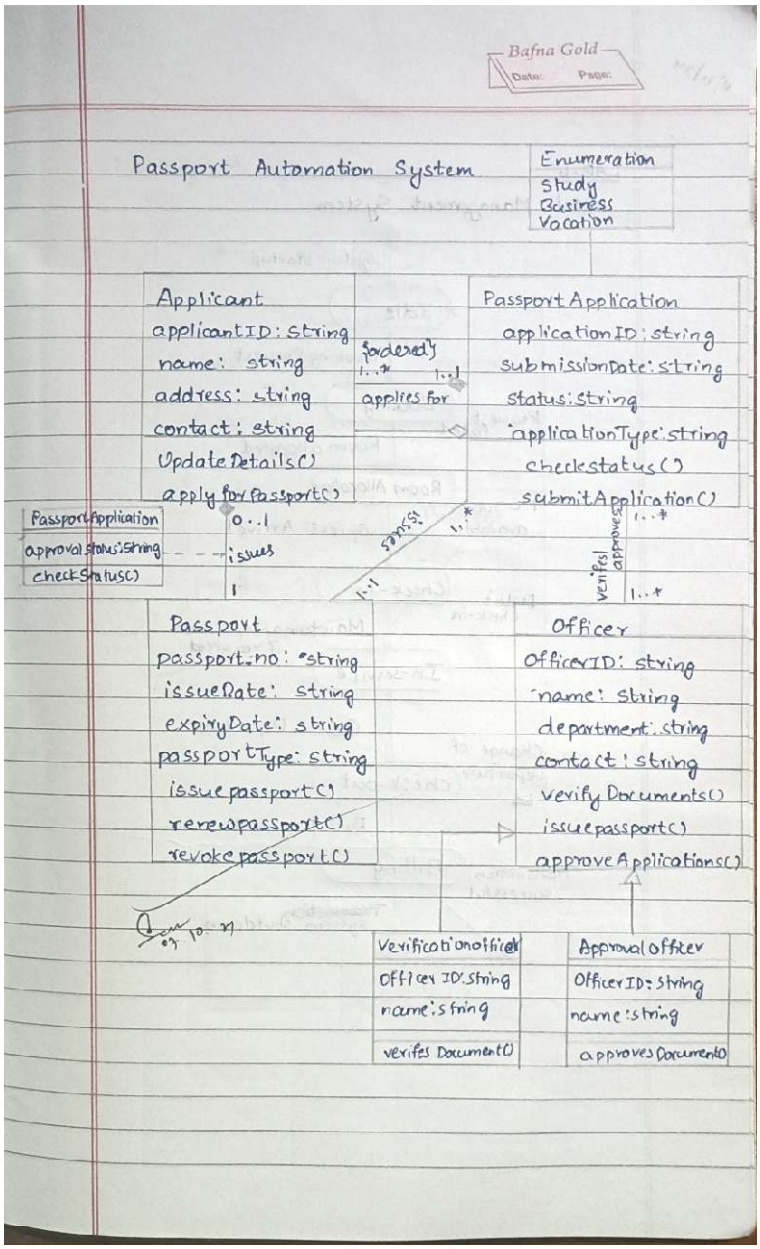


Figure 5.1: Class Diagram

The Passport Automation System Class Diagram outlines key entities and their relationships, streamlining the process of passport issuance and management. The Application class manages passport applications with attributes like application ID, submission date, and status, while the Applicant class represents users applying for passports, with methods to submit applications, upload documents, and track status. The Passport class handles issued passports, including attributes like passport number, issue date, and expiry date, with functions to renew or revoke passports.

The Officer class, including specialized roles like DocumentOfficer and IssuingOfficer, manages tasks such as verifying documents, approving applications, and issuing passports. Applications are associated with the Document class, which validates required files. The system leverages enumerations like ApplicationType (e.g., Work, Study, Business) to categorize applications. Associations define critical workflows, such as applicants submitting multiple applications, officers verifying documents, and issuing or revoking passports. Together, these components ensure a secure, efficient, and transparent process for passport management.


Bafna Gold
Date: Page:
Passport Automation System
Enumeration
Study
Business
Vacation
Applicant
applicantID: String
name: string
address: string
contact: string
UpdateDetails()
apply for passport()
{ordered}
1..*
1..1
applies for
PassportApplication
applicationID: string
submissionDate: string
status: string
applicationType: string
checkstatus()
submitApplication()
PassportApplication
approvalStatus: String
checkStatus()
0..1
issues
1
issued
1..1
1..*
verifies/approves
1..*
Passport
passport_no: string
issueDate: string
expiryDate: string
passportType: string
issuepassport()
renewpassport()
revokepassport()
Officer
OfficerID: string
name: string
department: string
contact: string
verifyDocuments()
issuepassport()
approveApplications()
0..*
VerificationOfficer
OfficerID: string
name: string
verifiesDocument()
ApprovalOfficer
OfficerID: String
name: string
approvesDocument()

STATE DIAGRAM

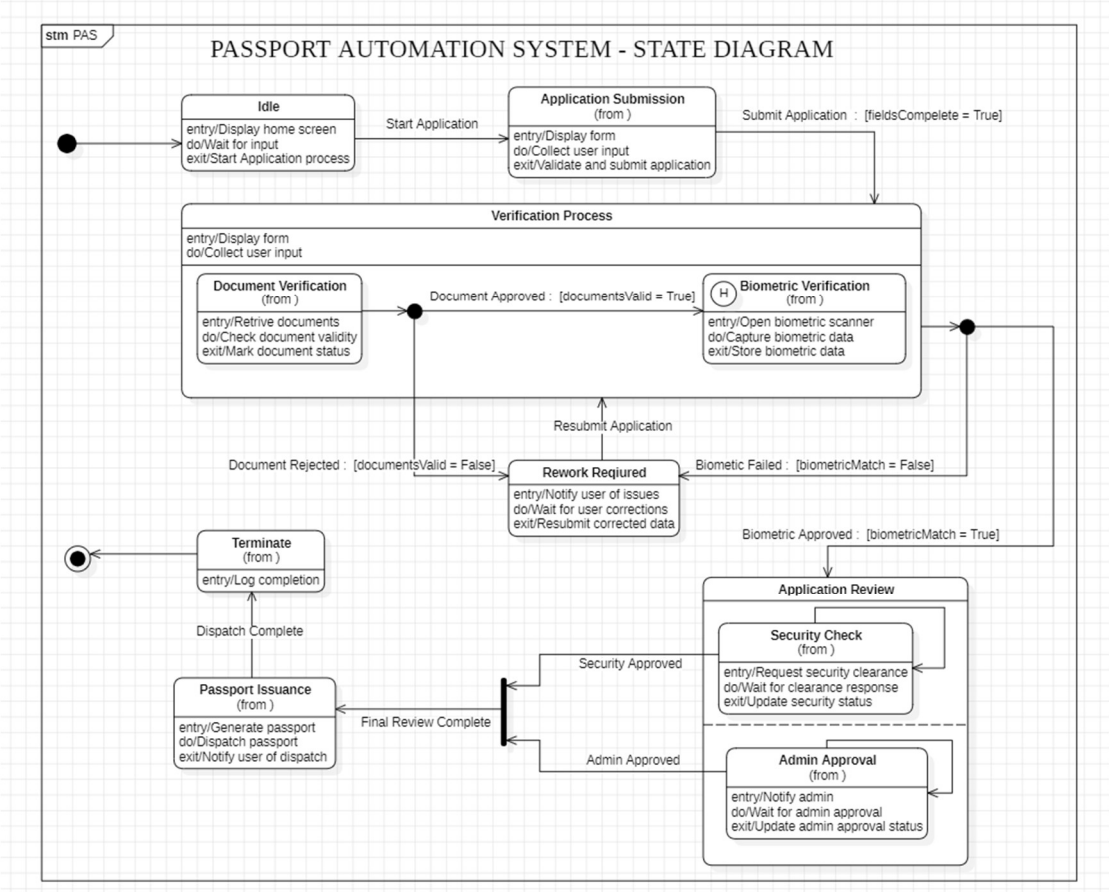


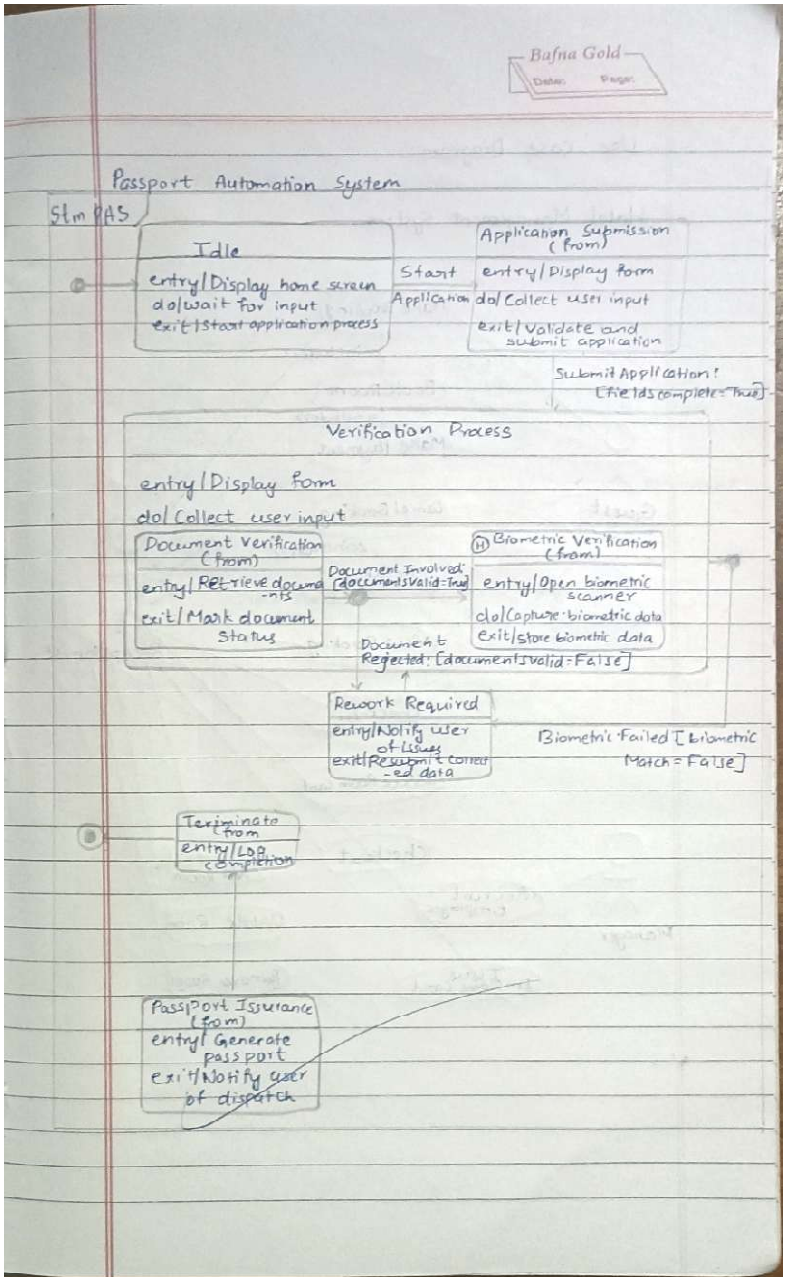
Figure 5.2: State Diagram

The passport application process begins in the Idle state, where the system awaits user interaction. When a user initiates the process by submitting an application, the system transitions to the Application Submission state, where the user inputs data and uploads documents. Submitted applications move to the Verification Process, which includes Document Verification and Biometric Verification. Documents are checked for validity, and biometric data such as fingerprints and facial recognition are captured and stored. If any issues arise, the system transitions to Rework Required, notifying the applicant to make corrections before resubmission.

Once documents and biometrics are approved, the process advances to the Security Check for clearance, followed by Admin Approval, where administrative personnel review the application. Approved applications enter the Final Review Complete state, leading to Passport Issuance, where the passport is generated and

prepared for dispatch. The Dispatch Complete state signifies the successful delivery of the passport to the applicant.

Key transitions include triggers like document approval, biometric verification, security clearance, and admin approval. Activities such as form display, data validation, security status updates, and passport generation ensure smooth operation across states. This diagram encapsulates the end-to-end workflow, emphasizing system efficiency and user engagement in the passport application lifecycle.



USE CASE DIAGRAM

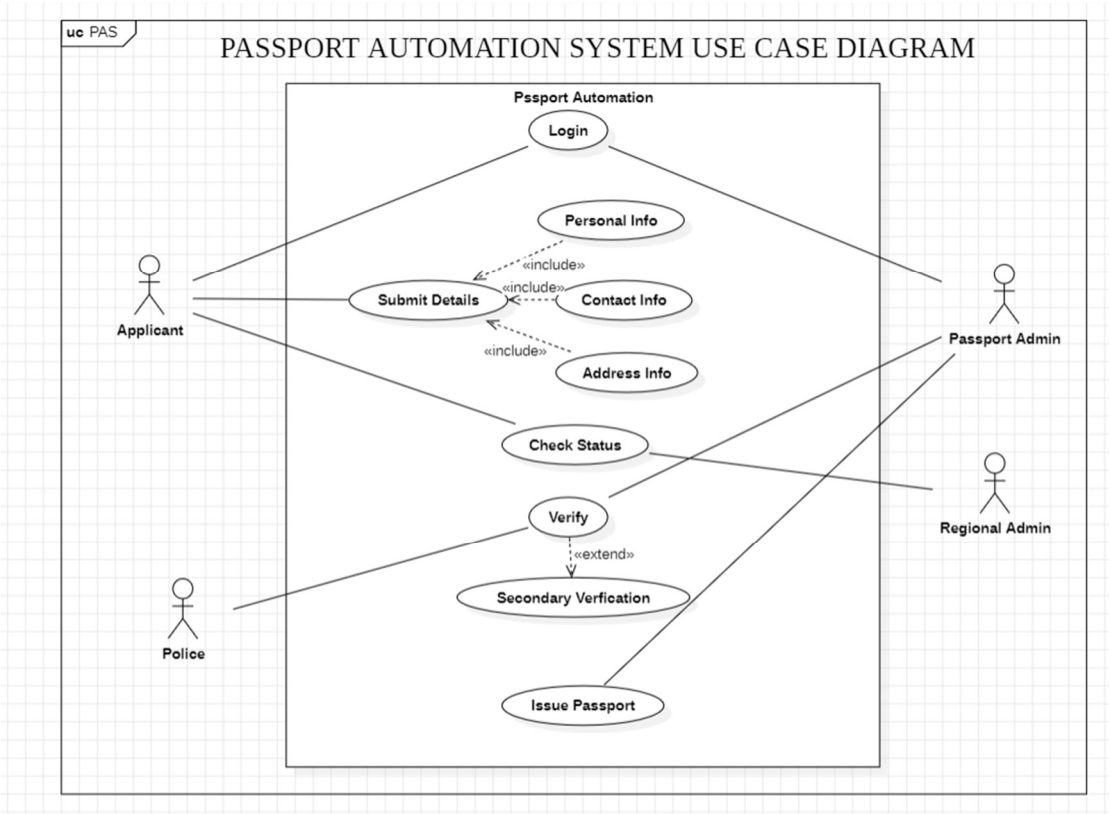
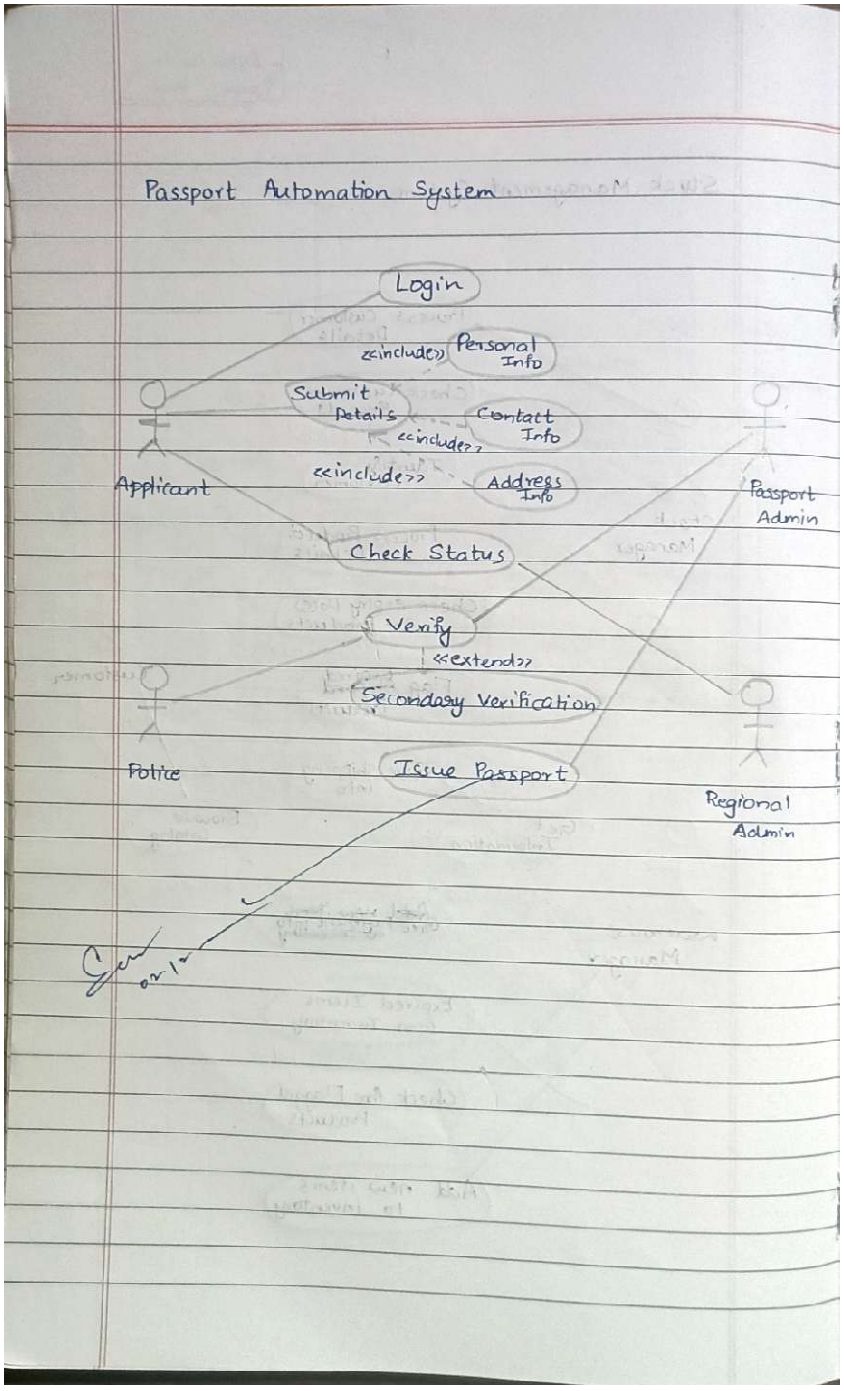


Figure 5.3: Use Case Diagram

The Passport Automation System involves several key actors: the Applicant, Passport Admin, Admin, Regional Admin, and Police. The main use case, Passport Automation, encompasses all the steps in the passport application process, from login to the final issuance of the passport. Applicants start by logging into the system, then proceed to Submit Details, which includes entering personal, contact, and address information. The Passport Admin or system can then allow the applicant to Check Status of their application at any stage.

Verification is a crucial part of the process. The Verify use case, which can be extended to include Secondary Verification (involving police or other agencies), ensures all applicant details are accurate before proceeding. Once verified, the system proceeds to Issue Passport, the final step in the process. Relationships like <<include>> (e.g., submitting personal, contact, and address information) and <<extend>> (e.g., extending verification with additional checks) define the flow and dependencies between these use cases. This diagram helps visualize the interactions and responsibilities of each actor in the passport application process.


Passport Automation System
Login
<<include>>
Personal Info
Submit Details
Contact Info
<<include>>
<<include>>
Address Info
Applicant
Passport Admin
Check Status
Verify
<<extend>>
Secondary Verification
Police
Issue Passport
Regional Admin

SEQUENCE DIAGRAM

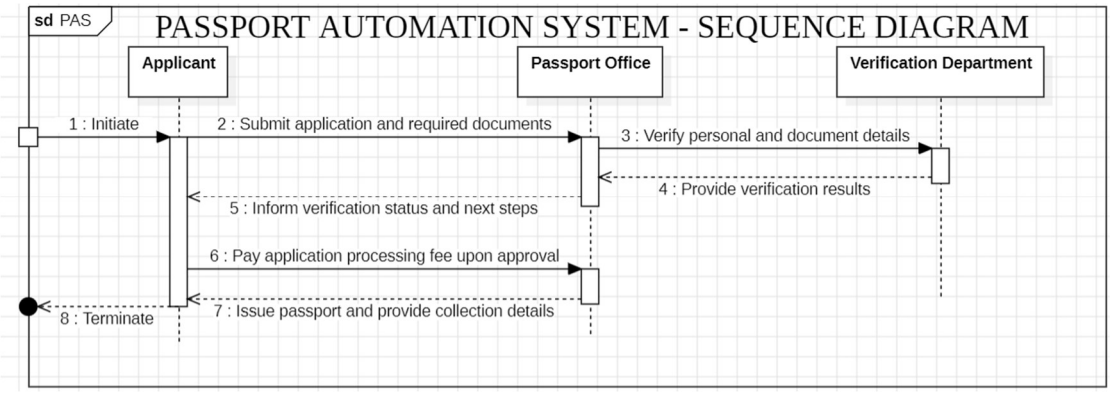
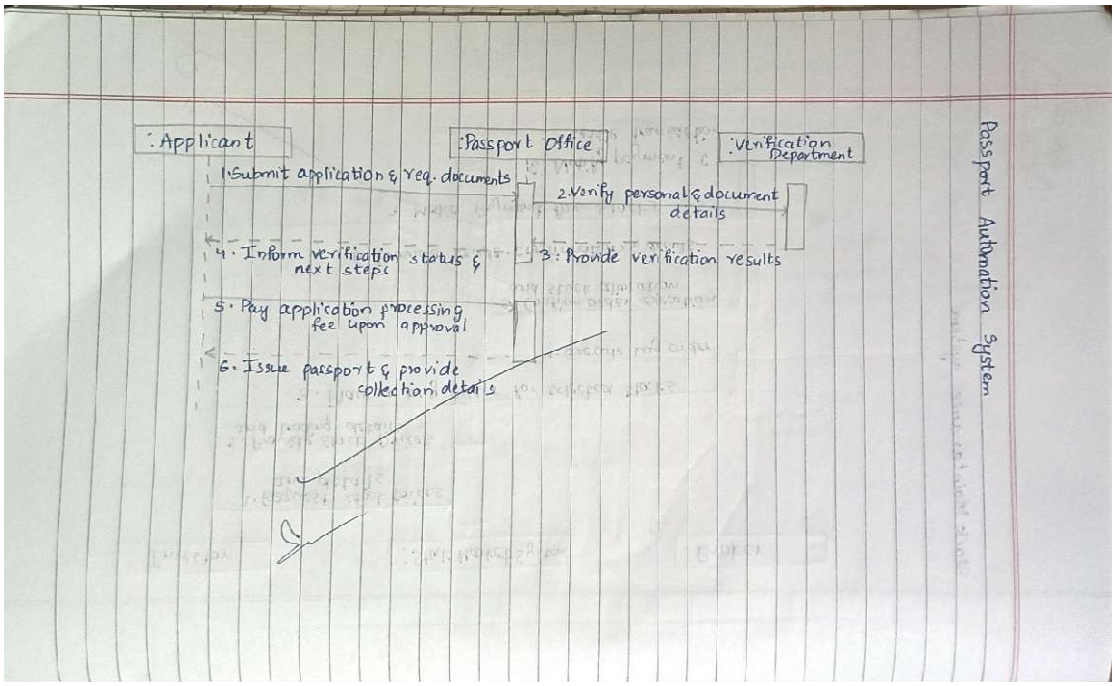


Figure 5.4: Sequence Diagram

This sequence diagram illustrates the interactions between the Applicant, Passport Office, and Verification Department during the passport application process. The Applicant begins by submitting the application and required documents to the Passport Office. The Passport Office sends these details to the Verification Department for verification. Once the verification is completed, the Passport Office informs the Applicant about the verification status and the next steps, such as payment or appointment scheduling.

If the application is approved, the Applicant proceeds to pay the processing fee. Upon successful payment and any necessary background checks, the Passport Office issues the passport and provides collection details to the Applicant. The process is completed once the passport is issued and collected. The diagram emphasizes the flow of information between the entities and the critical steps involved in obtaining a passport.


:Applicant
:Passport Office
:Verification Department
1:Submit application & req. documents
2.Verify personal & document details
3: Provide verification results
4. Inform verification status & next steps
5. Pay application processing fee upon approval
6. Issue passport & provide collection details
Passport Automation System

ACTIVITY DIAGRAM

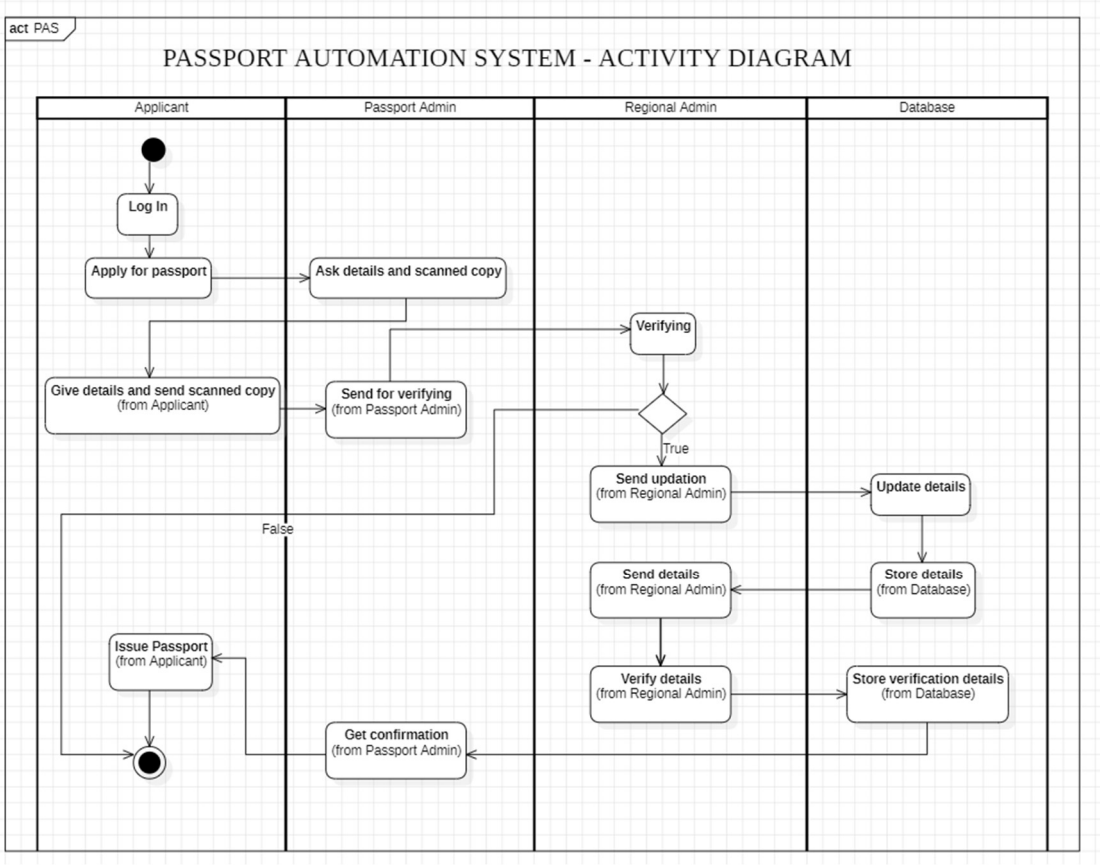


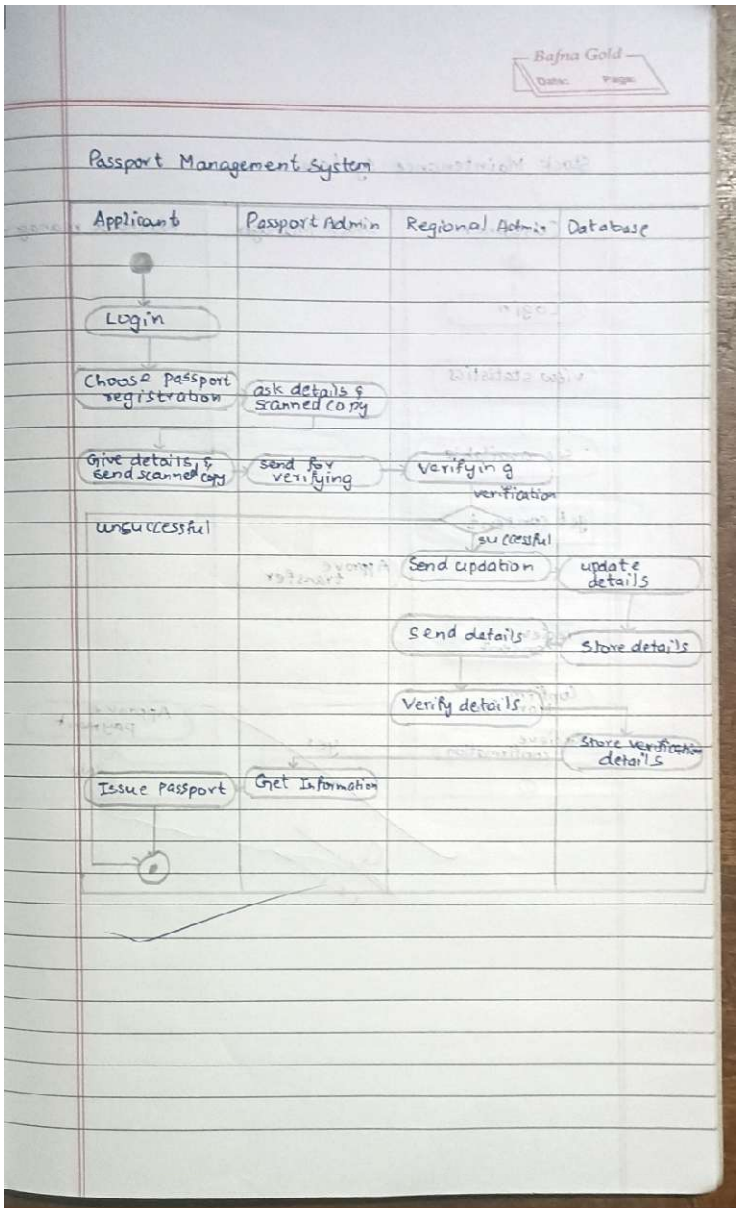
Figure 5.5: Activity Diagram

In this swimlane diagram, the flow of activities is divided among the Applicant, Passport Admin, Regional Admin, and Database swimlanes.

The Applicant starts the process by logging into the system and applying for a passport. The applicant then provides personal details and uploads scanned copies of the required documents, and finally, receives the issued passport once the process is completed.

The Passport Admin requests the necessary details and scanned documents from the applicant, then forwards the application to the Regional Admin for verification. After receiving the confirmation from the regional admin regarding the verification status, the passport admin proceeds with the next steps.

The Regional Admin is responsible for verifying the applicant's information and documents. Once verified, the regional admin sends the update back to the passport admin and updates the database with the verified details. The Database stores both the applicant's details and the verification results provided by the regional admin, ensuring all information is accurately recorded for further processing.


Bafna Gold
Passport Management System
Applicant
Passport Admin
Regional Admin
Database
Login
Choose passport registration
ask details & scanned copy
Give details & send scanned copy
send for verifying
Verifying
verification
unsuccessful
successful
Send updation
update details
Send details
Store details
Verify details
Store verification details
Issue passport
Get Information