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

Lab Record

Software Engineering and Object-Oriented Modeling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

MANASVINI DEEPAK

1BM22CS336

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

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Analysis and Design(22CS6PCSEO) laboratory has been carried out by MANASVINI DEEPAK (1BM22CS336) during the 5th Semester Oct24-Jan2025.

Signature of the Faculty Incharge:

ANUSHA S

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

Table of Contents

1. HOTEL MANAGEMENT SYSTEM

1.1: PROBLEM STATEMENT:

The hotel management industry faces challenges such as inefficiencies in room reservations, payment processing, resource allocation, and record management due to fragmented or manual systems. These issues lead to operational delays, errors like double-bookings, and reduced customer satisfaction. To address these, an **Advanced Hotel Management System** is proposed, integrating booking management, secure payment processing, invoice generation, and resource tracking, including room availability and amenities. This system will streamline operations, enhance customer experiences, improve data accuracy, and support scalability with a modular, user-friendly design.

1.2: SRS DOCUMENT:

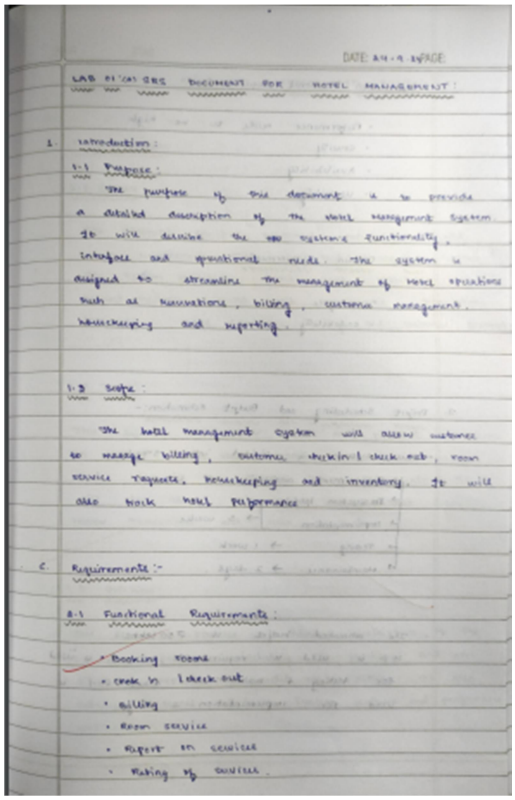


FIG 1.2.1

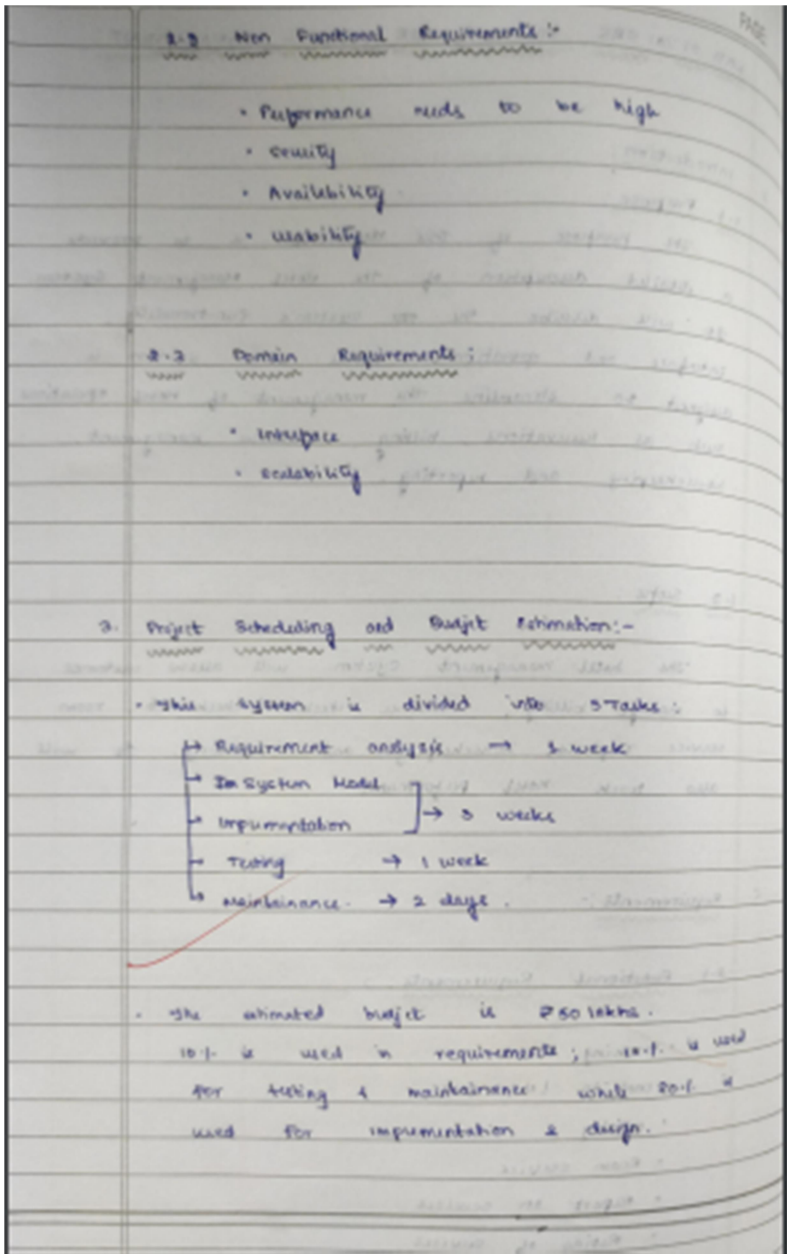

2.2 Non Functional Requirements :-
• Performance needs to be high
• Security
• Availability
• usability
2.3 Domain Requirements :
• Interface
• Scalability
3. Project Scheduling and Budjet Estimation :-
• this system is divided into 5 tasks :
Requirement analysis → 1 week
System Model
Implementation → 3 weeks
Testing → 1 week
maintainance → 2 days.
• the estimated budjet is ₹50 lakhs.
10% is used in requirements ; 10% is used
for testing & maintainance while 20% is
used for implementation & design.

FIG 1.2.2

1.3: CLASS DIAGRAM:

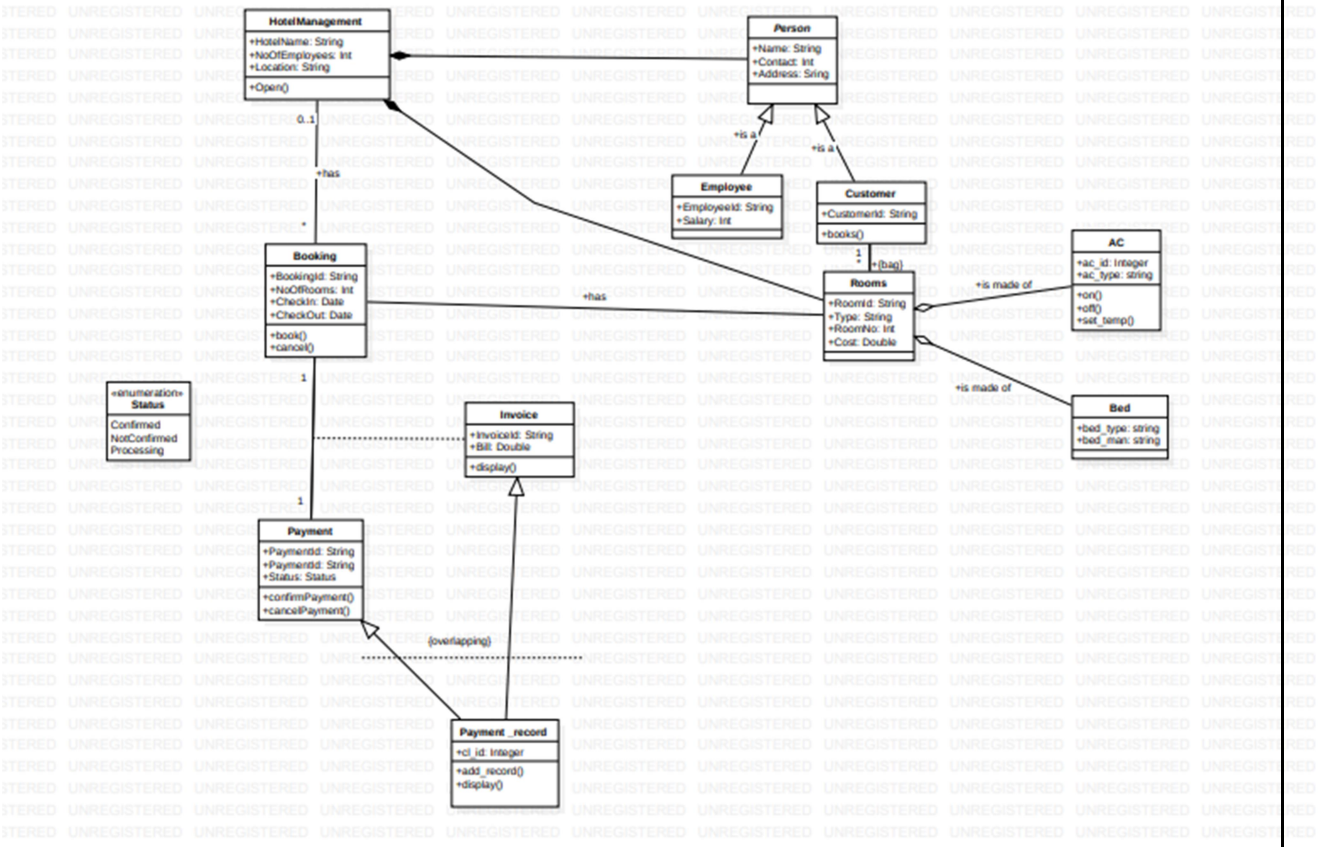


FIG 1.3.1

The advanced hotel management system class diagram outlines a structured framework for managing hotel operations efficiently. At its core, the **Hotel Management** class holds essential details like the hotel name, location, and employee count, with operations to manage the hotel's state. The **Booking** class facilitates reservations, linking with **Person** entities (employees and customers) and **Rooms**, each with attributes such as type, cost, and room number. Payment processing is handled through the **Payment** class, which includes payment statuses and operations to confirm or cancel payments, integrated with the **Invoice** class for billing. Additional components like **AC** and **Bed** classes ensure room amenities are managed effectively, while the **Payment Record** class tracks transaction histories. Clear relationships such as `has`, `books()`, and inheritance (`is a`) between classes ensure modularity, making the system scalable and easy to maintain.

1.4: STATE DIAGRAM:

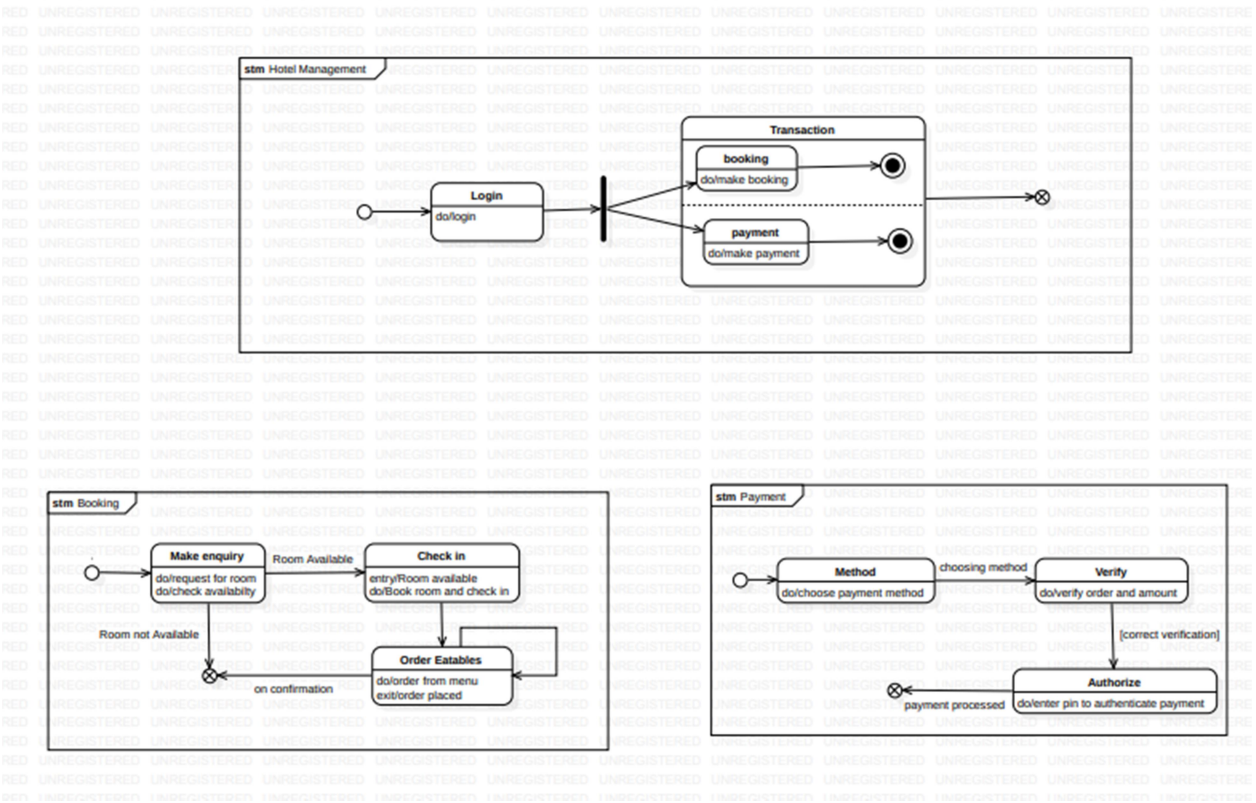


FIG 1.4.1

The state machine diagram for the **Advanced Hotel Management System** outlines the various states and transitions involved in managing hotel operations. Here's an explanation of the key states and transitions:

Key States:

1. **Login**:
    - **Activity**: The user logs into the system.
    - Transition: Upon successful login, the system proceeds to the next relevant action, such as booking or inquiry.
2. **Make Enquiry**:
    - **Activity**: The user requests information about room availability.
    - Transition: Depending on room availability, it moves to the "Room Available" or "Room Not Available" state.
3. **Check-In**:

- **Activity**: If a room is available, the system allows the user to book the room and check in.
- Transition: On successful booking, it progresses to the payment state.

4. **Order Eatables**:
   - **Activity**: The user places orders for food or beverages.
   - Transition: Once the order is placed, the system moves to the payment verification state.
5. **Payment Method**:
   - **Activity**: The user selects a payment method (e.g., credit card, online payment).
   - Transition: The system verifies the chosen payment method before processing.
6. **Verify**:
   - **Activity**: The system verifies the order details and payment amount.
   - Transition: If verification is successful, it moves to the authorization state.
7. **Authorize**:
   - **Activity**: The user authenticates the payment by entering credentials (e.g., PIN or OTP).
   - Transition: On successful authorization, payment is processed.
8. **Transaction (Booking or Payment)**:
   - **Activity**: Final steps to confirm the booking or complete the payment.
   - Transition: Returns to a ready state for the next user interaction or logs the transaction.

Major Transitions:

- From **Login** to operational states such as "Make Enquiry" or "Check-In" depending on user action.
- From **Make Enquiry** to "Room Available" or "Room Not Available" based on availability.
- From **Payment Method** to "Verify" and then to "Authorize" as part of the payment flow.
- From **Order Eatables** to "Payment" for verifying and processing payment for additional orders.

Summary:

The diagram ensures a logical flow of actions, providing a clear sequence for operations like login, room bookings, payments, and service management. Each transition reflects a user action or system validation, ensuring smooth operations and accurate record-keeping.

1.5: USE CASE DIAGRAM:

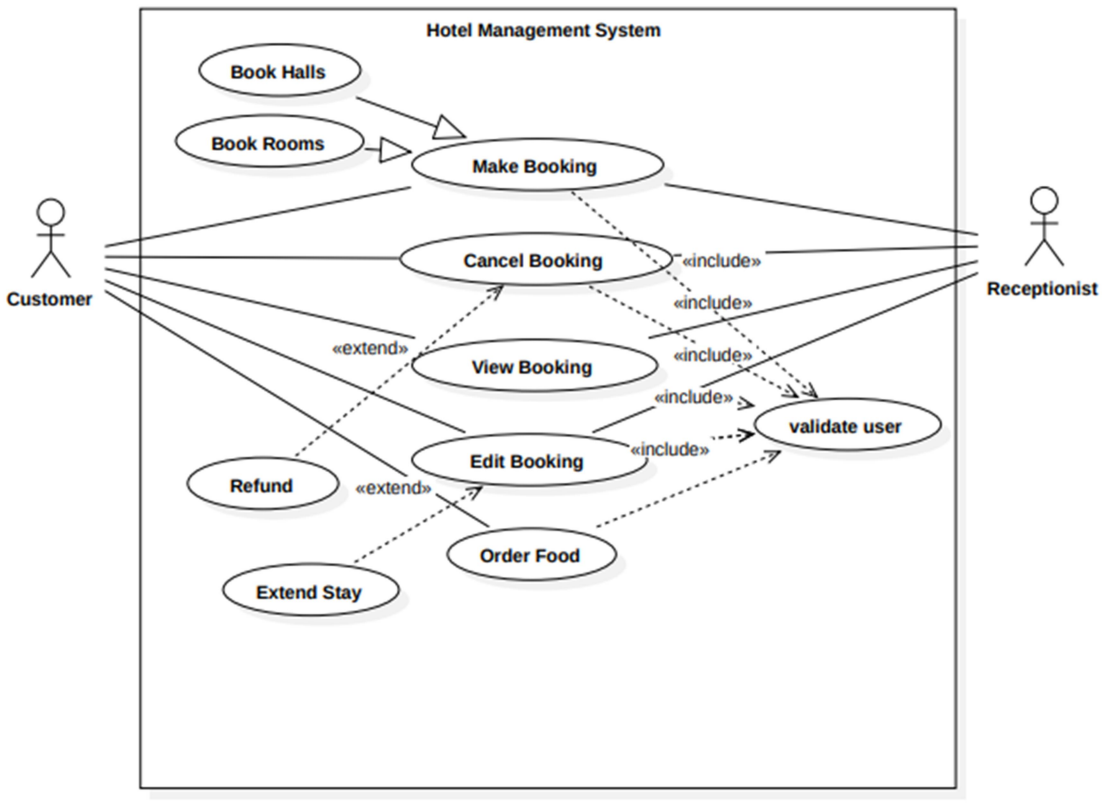


FIG 1.5.1

The **Use Case Diagram** for the Advanced Hotel Management System depicts the primary interactions between two actors: the **Customer** and the **Receptionist**, and their roles in managing bookings and related hotel operations. Key use cases include **Make Booking**,

Cancel Booking, **View Booking**, and **Edit Booking**, which are fundamental to managing room reservations. Additional use cases, such as **Order Food** and **Book Halls**, indicate extended services available to customers. These interactions ensure that the system covers essential hotel functionalities, offering a streamlined experience for users.

The diagram also includes advanced interactions like **Refund** and **Extend Stay**, which are modeled as extensions of standard booking workflows. These features enhance the system's flexibility to handle real-world scenarios, such as customers requesting refunds or extending their stays. An included relationship for **validate user** ensures secure and authorized access to system functionalities, maintaining data integrity and user privacy. Together, these use cases demonstrate a well-rounded system designed to handle both standard and exceptional operations in hotel management.

1.6: SEQUENCE DIAGRAM:

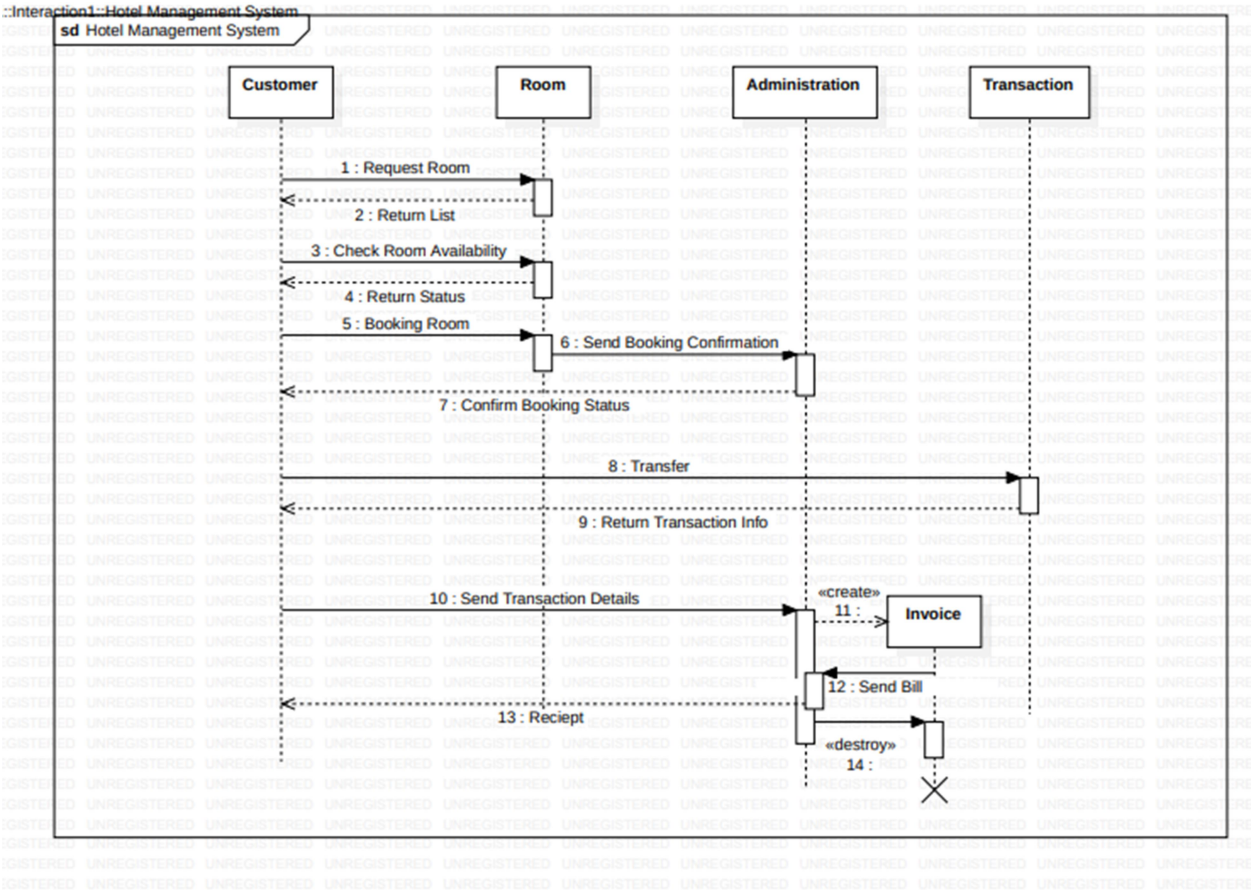
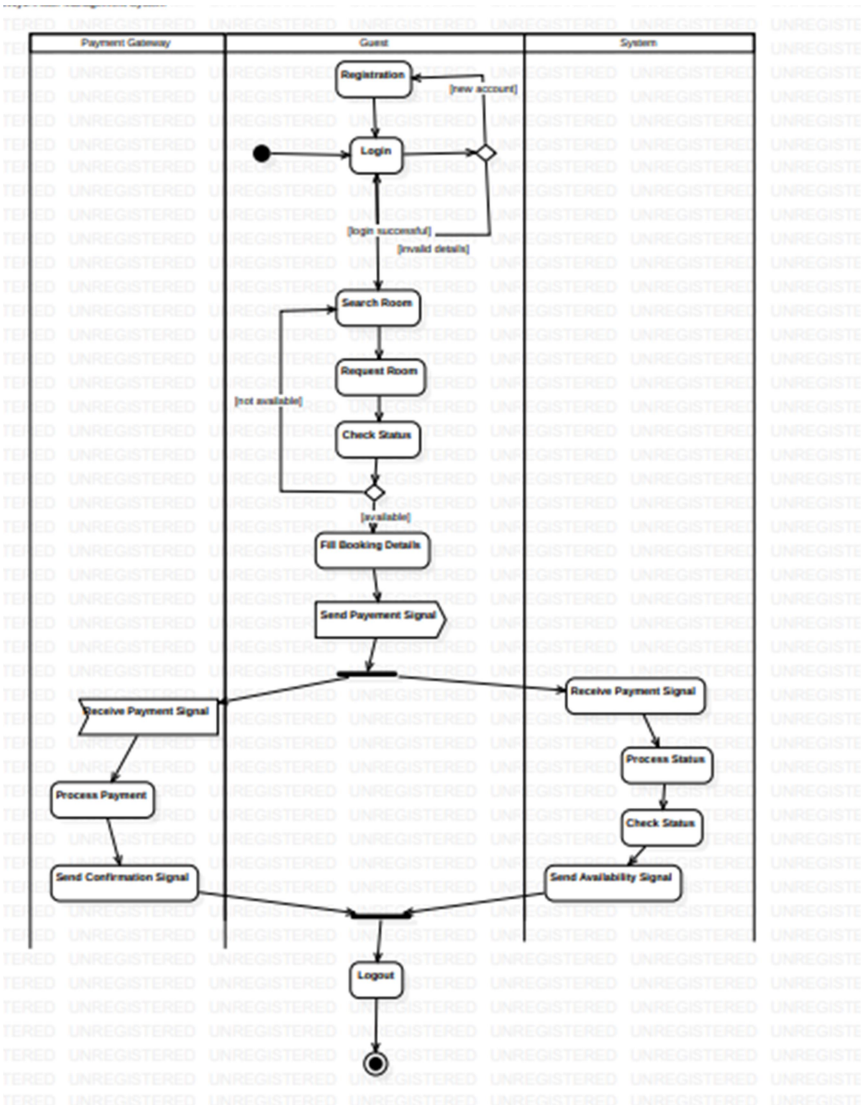


FIG 1.6.1

The sequence diagram illustrates the booking and payment process in a Hotel Management System. It begins with a **Customer** initiating a request to check room availability. The **Room** returns the status to the **Customer**, and if a room is available, a reservation is created by the **Administration**. The **Administration** then confirms the booking and sends transaction details to the **Customer**. The **Customer** then sends these details to generate invoice and receives a receipt.

1.7 : ACTIVITY DIAGRAM:



This activity diagram depicts the activity flow for a hotel room booking process. It involves three participants: the guest, the system, and the payment gateway. The guest

begins by logging in and searching for available rooms. Upon finding a suitable room, they request it, and the system checks its availability. If available, the guest fills in booking details and sends a payment signal. This triggers parallel actions: the payment gateway processes the payment and sends a confirmation signal if successful, while the system simultaneously changes the room's status to booked and sends an availability signal. Finally, upon receiving both signals, the process concludes with the guest logging out. This diagram illustrates the sequential and parallel activities involved in a typical online hotel room booking scenario

2. CREDIT CARD MANAGEMENT SYSTEM

2.1: PROBLEM STATEMENT:

Financial institutions and merchants face challenges in securely and efficiently managing credit card transactions. The process involves verifying customer details, processing payments, fraud detection, and ensuring regulatory compliance. An unreliable or inefficient system can lead to transaction delays, increased risks of fraud, and poor customer experience

2.2: SRS DOCUMENT:

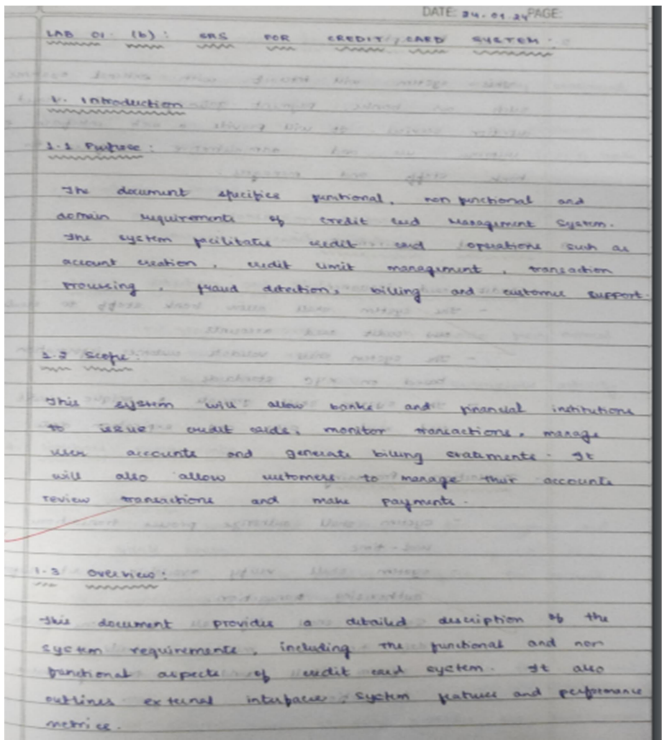


FIG 2.2.1

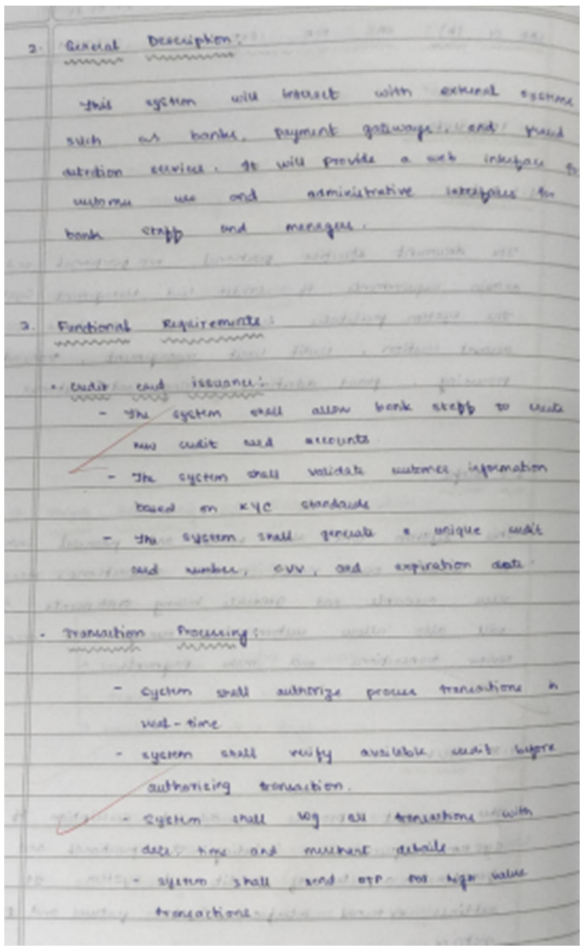


FIG 2.2.2

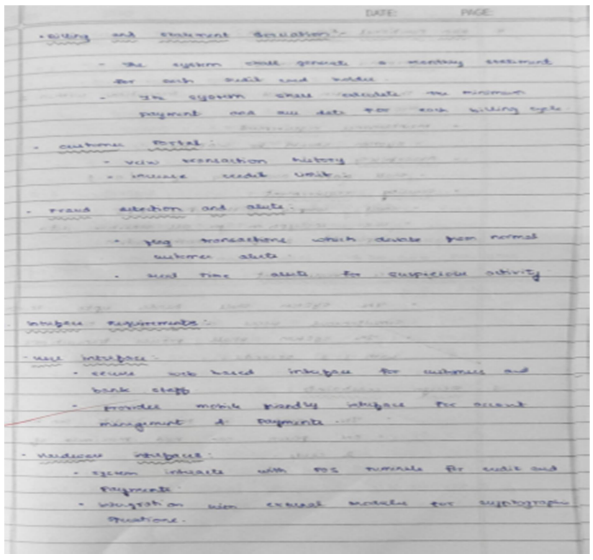
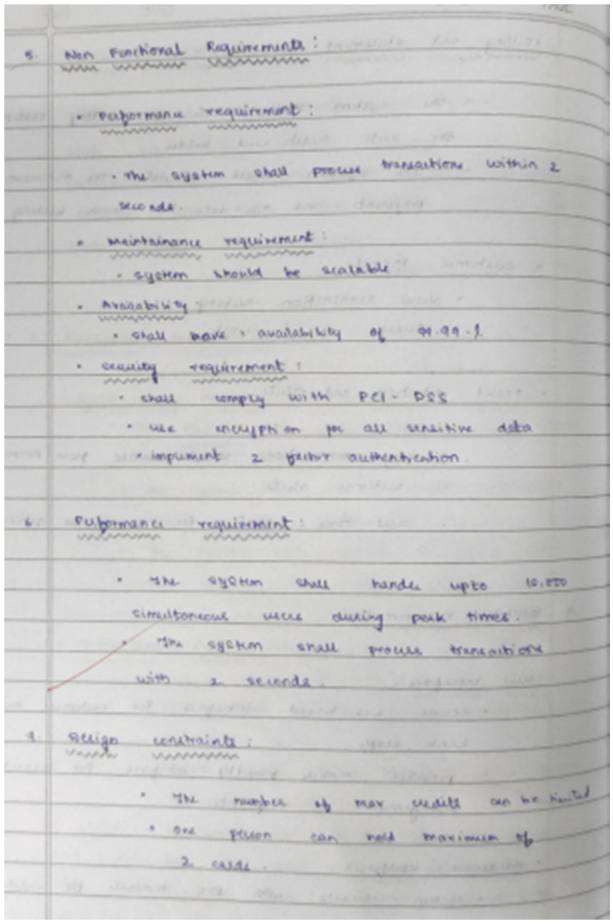


FIG 2.2.3



5. Non Functional Requirements :

- Performance requirement :
  - The system shall process transactions within 2 seconds.
- maintainance requirement :
  - system should be scalable
- Availability :
  - shall have : availability of 99.99 %
- security requirement :
  - shall comply with PCI - DSS
  - use encryption on all sensitive data
  - implement 2 factor authentication

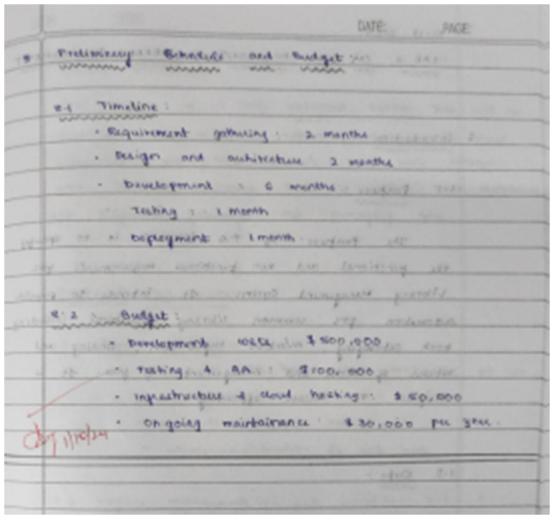
6. Performance requirement :

- The system shall handle upto 10,000 simultaneous users during peak times.
- The system shall process transactions with 2 seconds.

7. Design constraints :

- The number of max credits can be limited
- one person can hold maximum of 2 cards.

FIG 2.2.4



8. Preliminary Schedule and Budget :

8.1 Timeline :
- Requirement gathering : 2 months
- Design and architecture : 2 months
- Development : 6 months
- Testing : 1 month
- Deployment : 1 month

8.2 Budget :
- Development costs : $ 500,000
- Testing & QA : $ 100,000
- Infrastructure & cloud hosting : $ 50,000
- Ongoing maintainance : $ 30,000 per year

FIG 2.2.5

2.3:CLASS DIAGRAM:

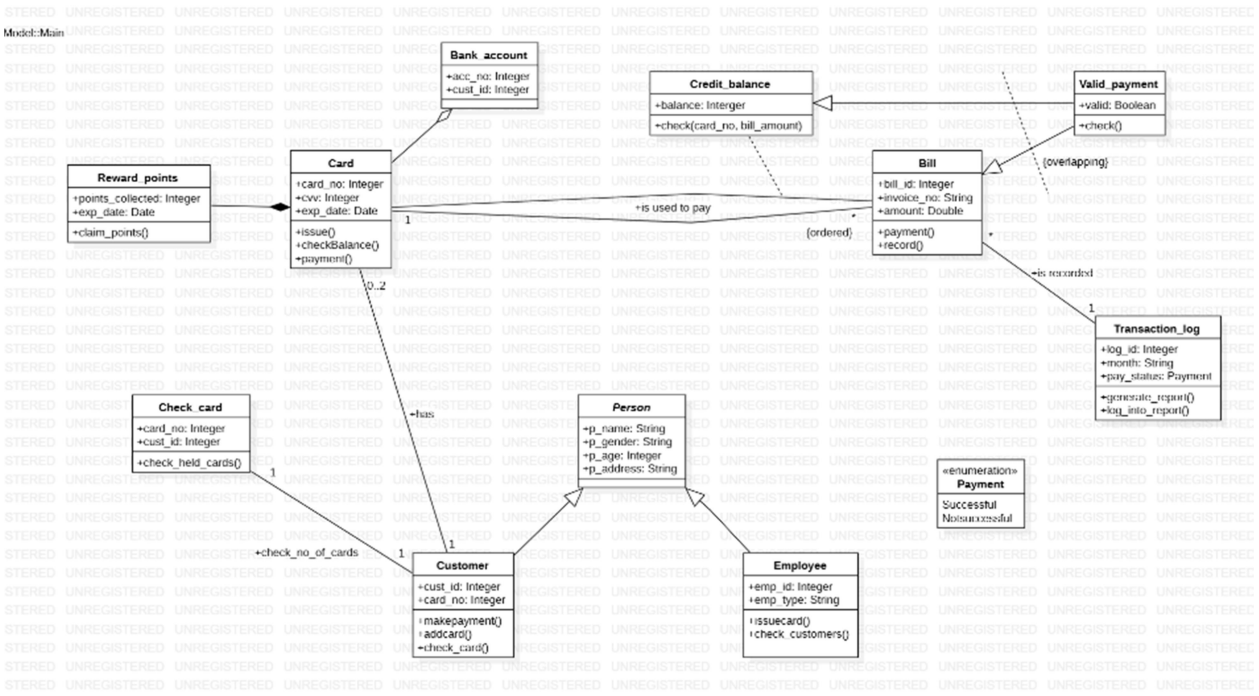


FIG 2.3.1

The class diagram shows the abstract PERSON class which has 2 subclasses Customer and Employee. The class CARD is associated to CUSTOMER and has a composition REWARD POINTS. It also has an aggregation BANK ACCOUNT. PAYMENT is an enumeration. The other classes are BILL,TRANSACTION LOG etc.

2.4: STATE DIAGRAM:

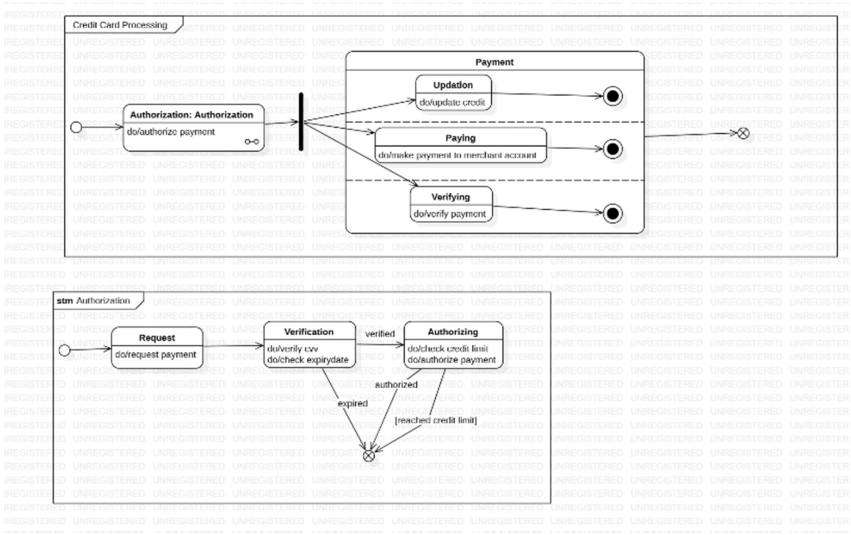


FIG 2.4.1

This state diagram models user interaction with a credit card system. Users begin by logging in, then proceed as either a Customer or Admin. Customers can "Make Payment" (involving card reading, PIN entry, and amount confirmation, leading to either successful payment or transaction denial) or "View Application." Admins can "View Application," which includes searching for applications and subsequently approving or rejecting them, updating the application status. Actions like reading card details or displaying payment invoices are associated with state transitions. The diagram clearly outlines the different paths and possible outcomes for both customer and admin users within the system.

2.5: USECASE DIAGRAM:

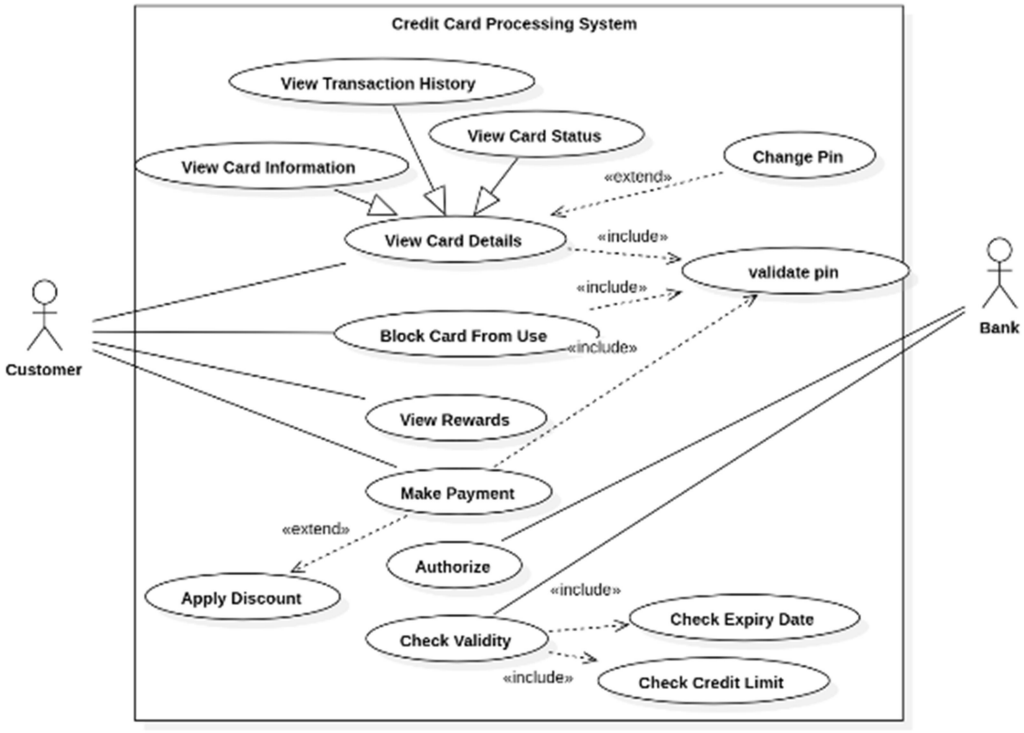


FIG 2.5.1

This use case diagram illustrates interactions within a Credit Card Processing System (CPS). It shows two actors: Customer and Bank. The primary use case is "Credit Card Processing," which involves several included use cases. The Customer can “view card

details”,”block card”, “make payment” ,”view rewards”. The Bank can “check validity of card” and “authorize payments”.

2.6: SEQUENCE DIAGRAM:

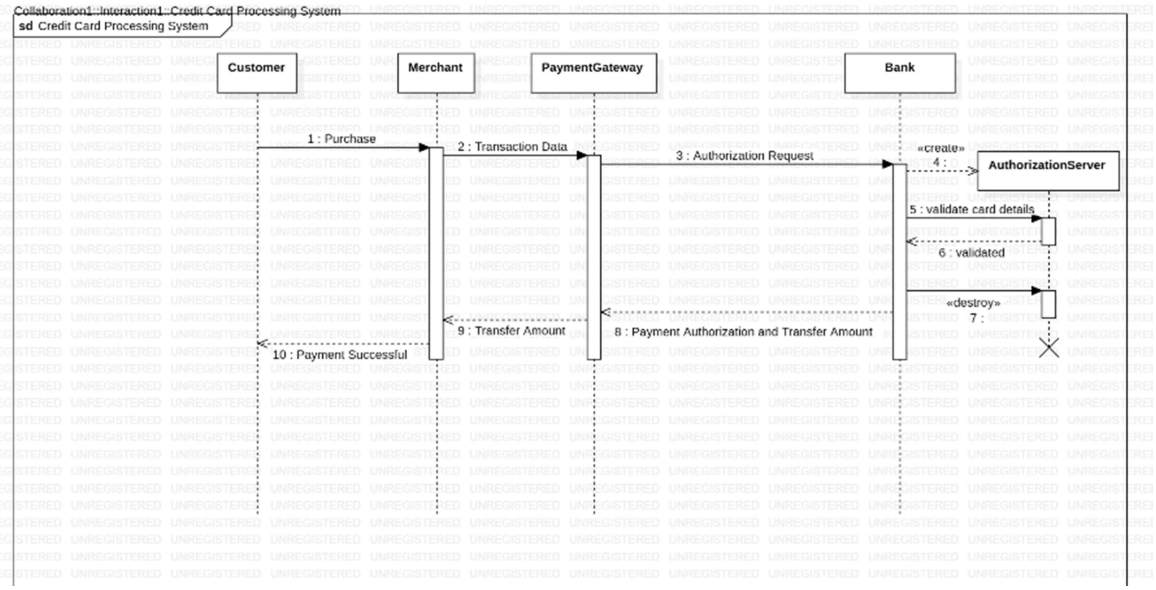


FIG 2.6.1

The scenario here is making a purchase. The CUSTOMER purchases from a Merchant who sends the transaction data to PaymentGateway which then sends authorization request to BANK. The Bank authorises payment and the details are sent back and payment is successful.

2.7: ACTIVITY DIAGRAM:

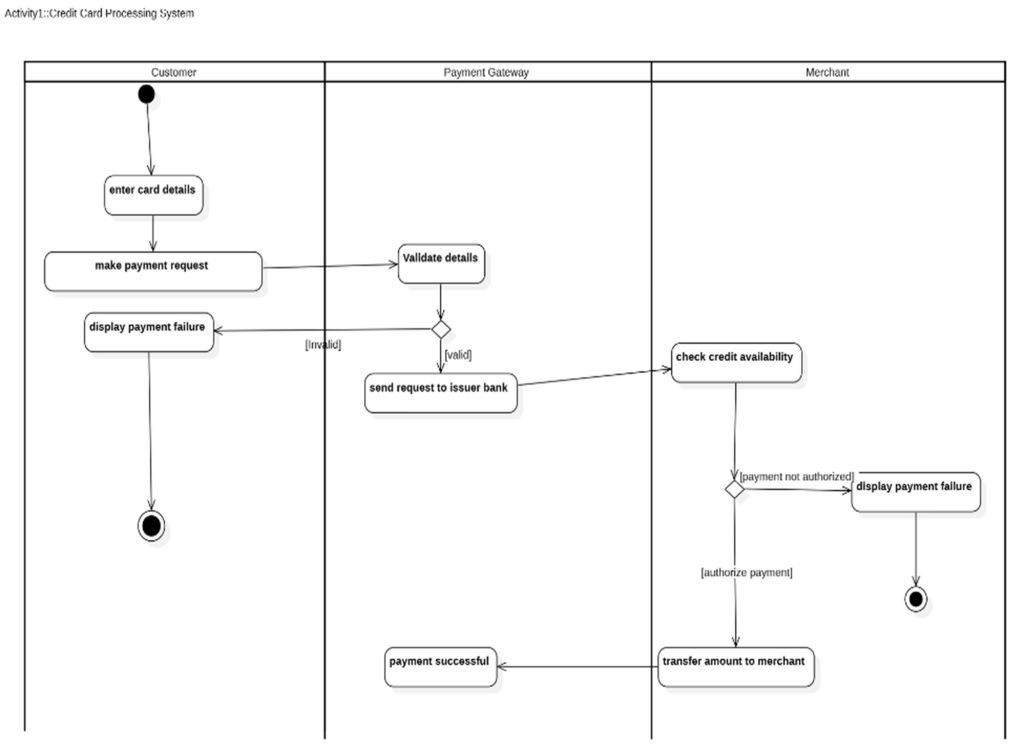


Fig 2.7.1

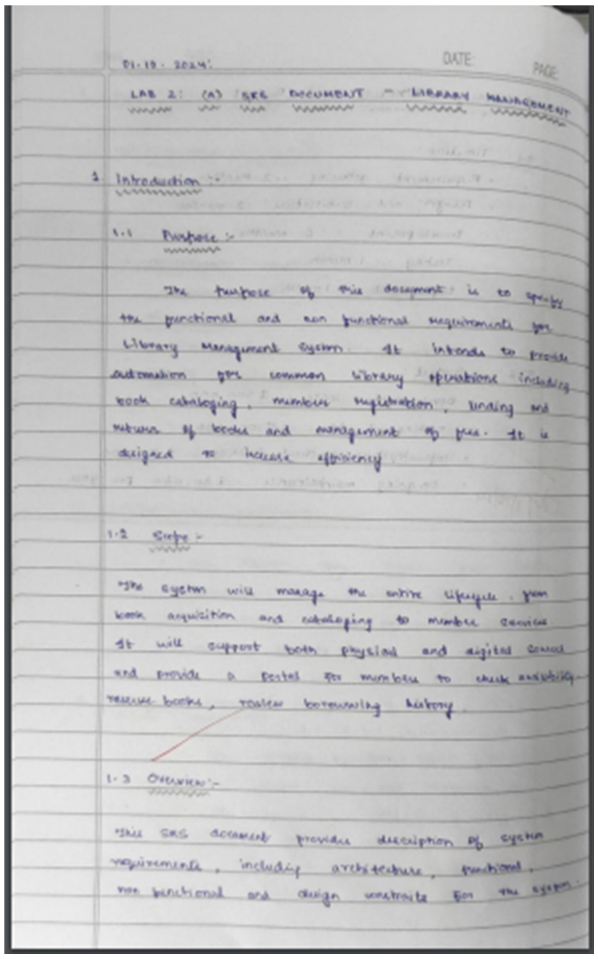
This activity diagram depicts the flow of a credit card transaction involving a Customer, Payment Gateway, and Merchant. The Customer begins by entering card details and making a payment request. The Payment Gateway then validates these details. If invalid, a payment failure is displayed to the Customer, and the process ends. If the details are valid, the Payment Gateway sends a request to the issuer bank, represented here by the Merchant's action of checking credit availability. If the payment is not authorized, a payment failure is displayed to the Merchant, and the process ends. However, if the payment is authorized, the amount is transferred to the Merchant, and a "payment successful" message is sent back through the Payment Gateway, completing the transaction. This diagram clearly shows the sequential steps and decision points in a credit card transaction, highlighting the interactions between the three parties involved.

3. LIBRARY MANGEMENT SYSTEM

3.1:PROBLEM STATEMENT:

Managing the operations of a library manually, such as tracking borrowed books, overdue returns, and inventory updates, is time-consuming and prone to errors. A robust system is needed to streamline these operations, improve efficiency, and enhance the experience for library users.

3.2:SRS DOCUMENT:



01.10.2024 DATE PAGE

LAB 2: (a) SRS DOCUMENT - LIBRARY MANAGEMENT

1. Introduction :-

1.1 Purpose :-

The purpose of this document is to specify the functional and non functional requirements for Library Management System. It intends to provide automation for common library operations including book cataloging, member registration, lending and returns of books and management of fees. It is designed to increase efficiency.

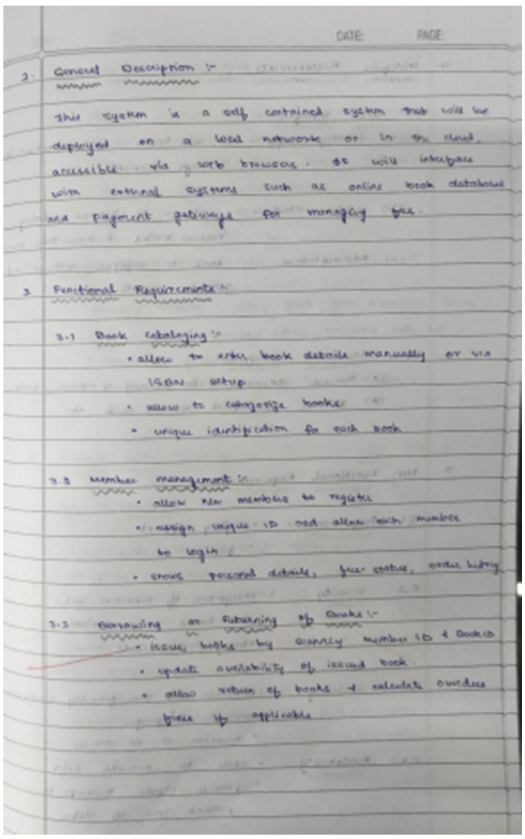
1.2 Scope :-

The system will manage the entire lifecycle, from book acquisition and cataloging to member service. It will support both physical and digital assets and provide a portal for members to check availability, reserve books, track borrowing history.

1.3 Overview :-

This SRS document provides description of system requirements, including architecture, functional, non functional and design constraints for the system.

FIG 3.2.1



2. General Description :-

This system is a self contained system that will be deployed on a local network or in the cloud, accessible via web browsers. It will interface with external systems such as online book databases and payment gateways for managing fees.

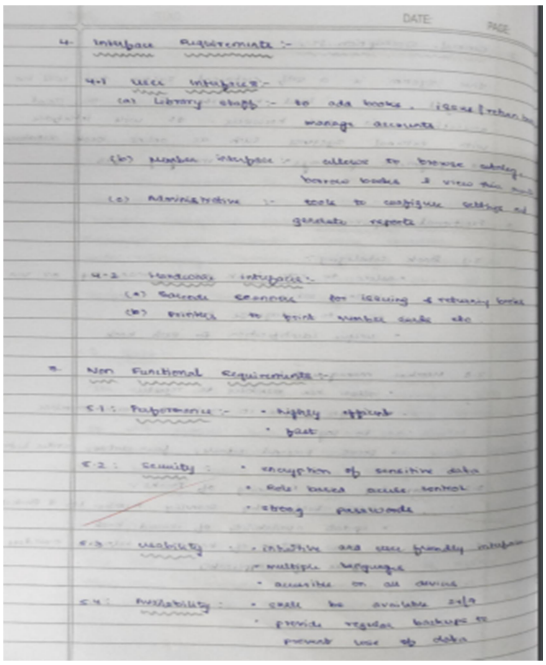
3. Functional Requirements :-

3.1 Book cataloging :-
- Allow to enter book details manually or via ISBN setup
- Allow to categorize books
- Unique identification for each book

3.2 Member management :-
- Allow new members to register
- Assign unique ID and allow each member to login
- Store personal details, fine status, order history

3.3 Borrowing or Returning of Books :-
- Issue books by scanning member ID & BookID
- Update availability of issued book
- Allow return of books & calculate overdues fines if applicable

FIG 3.2.2



4. Interface Requirements :-

4.1 User interface :-
(a) Library staff :- to add books, issue & return books, manage accounts
(b) Member interface :- allow to browse catalog, borrow books & view their history
(c) Administrative :- tools to configure settings & generate reports

4.2 Hardware interfaces :-
(a) Barcode scanners for issuing & returning books
(b) Printers to print member cards etc

5. Non Functional Requirements :-

5.1 Performance :- • highly efficient
• fast

5.2 Security :- • encryption of sensitive data
• Role based access control
• strong passwords

5.3 Usability :- • intuitive and user friendly interface
• multiple languages
• accessible on all devices

5.4 Availability :- • should be available 24/7
• provide regular backups to prevent loss of data

FIG 3.2.3

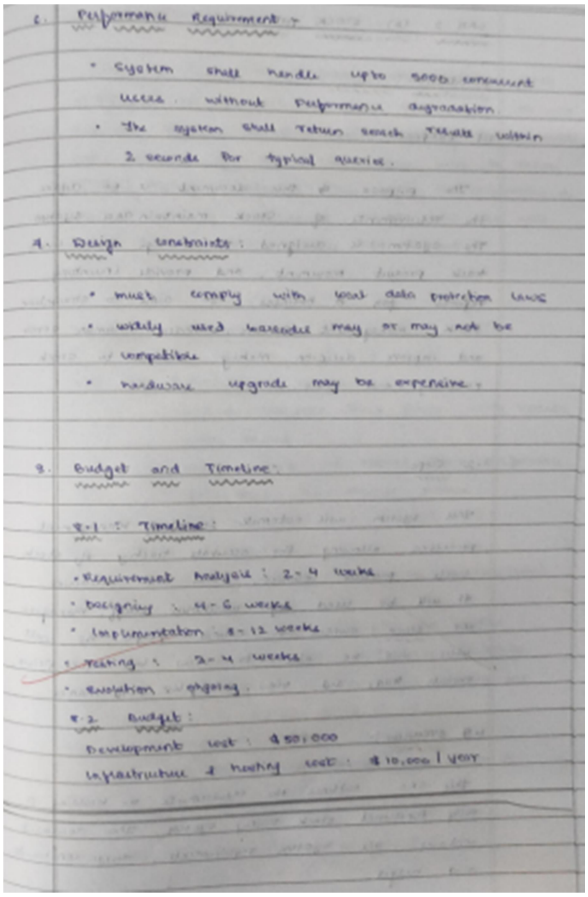

3. Performance Requirement :
• System shall handle upto 5000 concurrent
users without performance degradation.
• The system shall return search results within
2 seconds for typical queries.
4. Design constraints :
• must comply with local data protection laws
• widely used hardware may or may not be
compatible
• hardware upgrade may be expensive.
5. Budget and Timeline :
5.1 : Timeline :
• Requirement Analysis : 2-4 weeks
• Designing : 4-6 weeks
• Implementation : 8-12 weeks
• Testing : 2-4 weeks
• Evolution : ongoing
5.2 Budget :
Development cost : $50,000
Infrastructure & hosting cost : $10,000 / year

FIG 3.2.4

3.3: CLASS DIAGRAM:

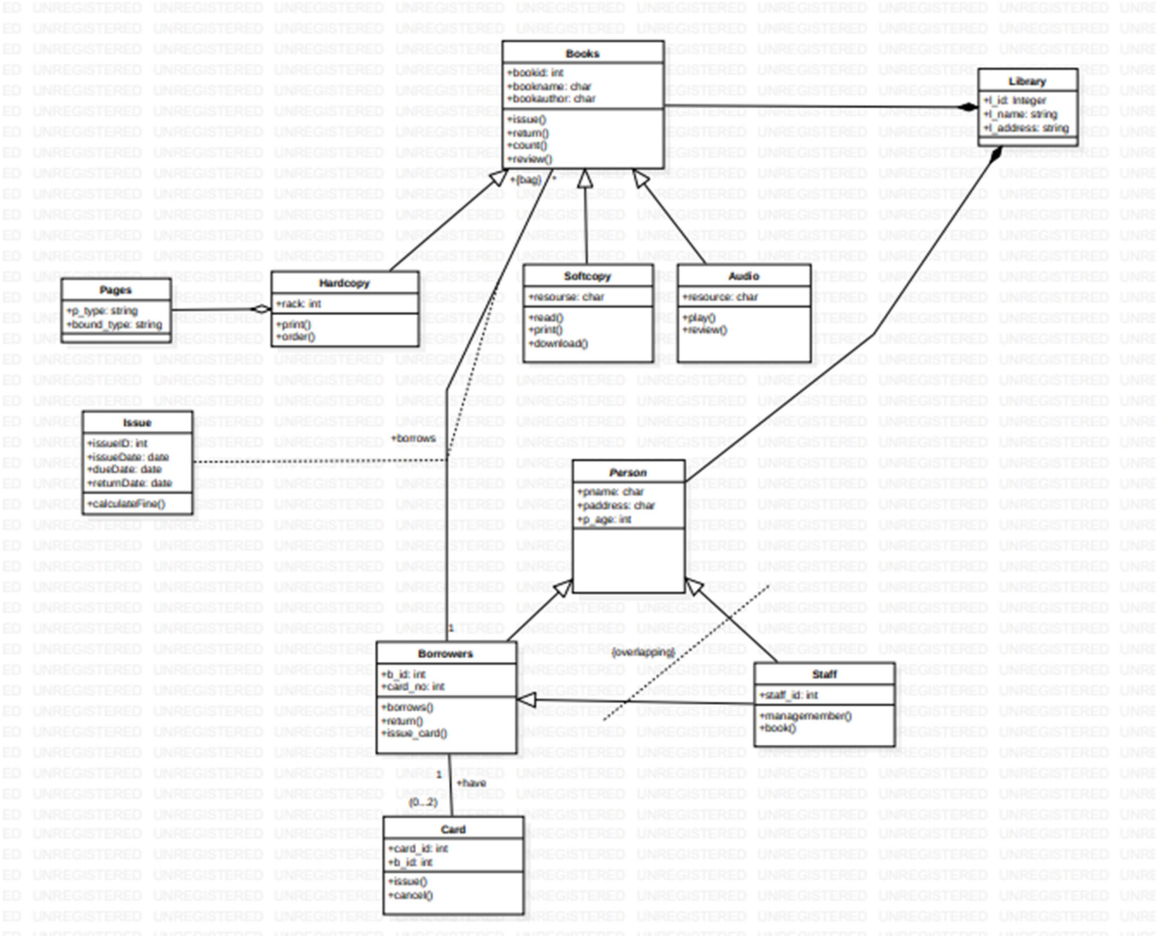


FIG 3.3.1

This UML depicts a class diagram for a Library Management System. It outlines the key entities involved and their relationships. The core entities include Books (with attributes like name, ID and author), Library Card (linking accounts to the library and tracking issued books), and Library (containing books, managing inventory, and handling book lists). Books can be either Hardcopy, Softcopy or Audio.

3.4: STATE DIAGRAM:

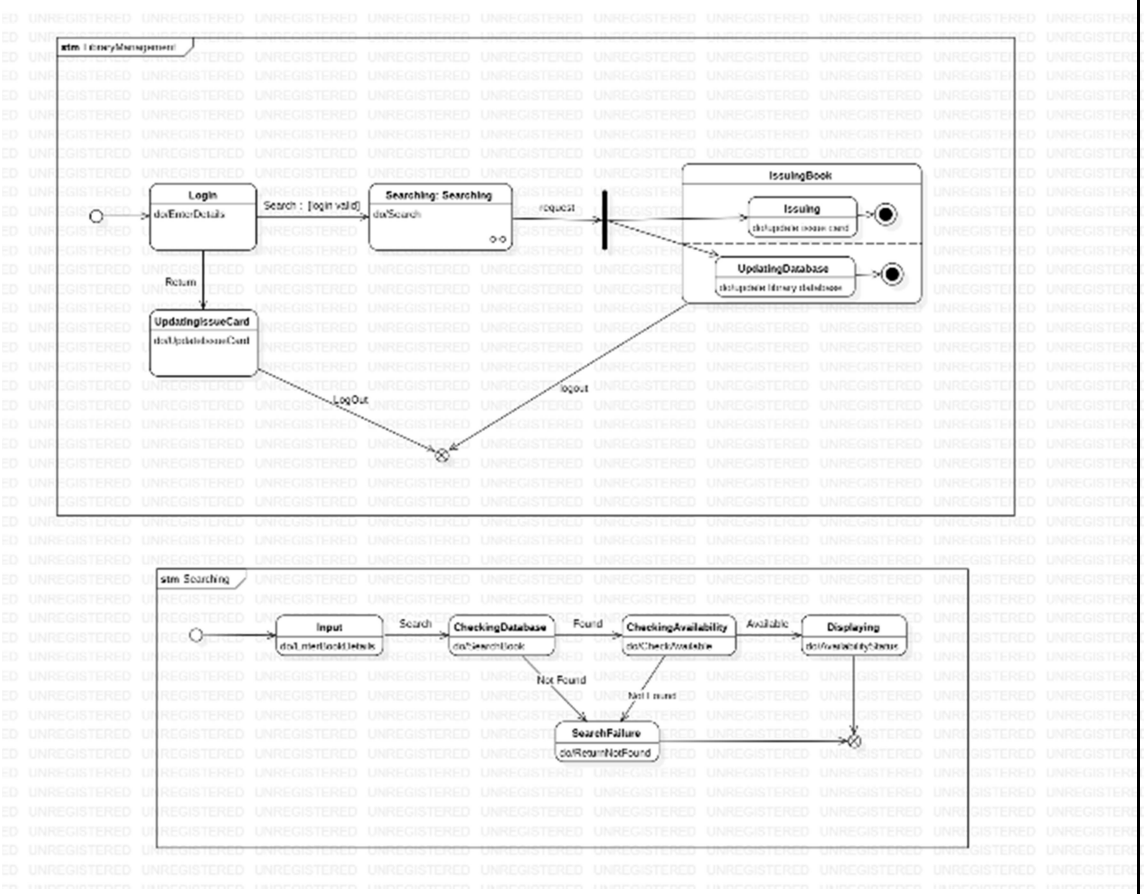


Fig 3.4.1

This state diagram for a Library Management System illustrates the system's dynamic behavior and user interactions. Starting with a login state requiring valid login details, the system branches into two main user roles: Admin and regular User. The Admin Menu allows actions like viewing books, which can lead to removing a specific book (requiring confirmation) or adding a new book via an "Add Book Form". Both removal and addition ultimately return to displaying the updated book list. The User Menu offers options to view the catalog, borrow a book (involving book selection and borrow confirmation), or return a book (with return confirmation). Both borrowing and returning books transition to a "Book Borrowed" or "Book Returned" state, respectively, before returning to the catalog display. Finally, both Admin and User roles can log out, transitioning the system to a logout state and completing the process. This diagram effectively visualizes the various paths users can take within the system and the corresponding state changes.

3.5: USECASE DIAGRAM:

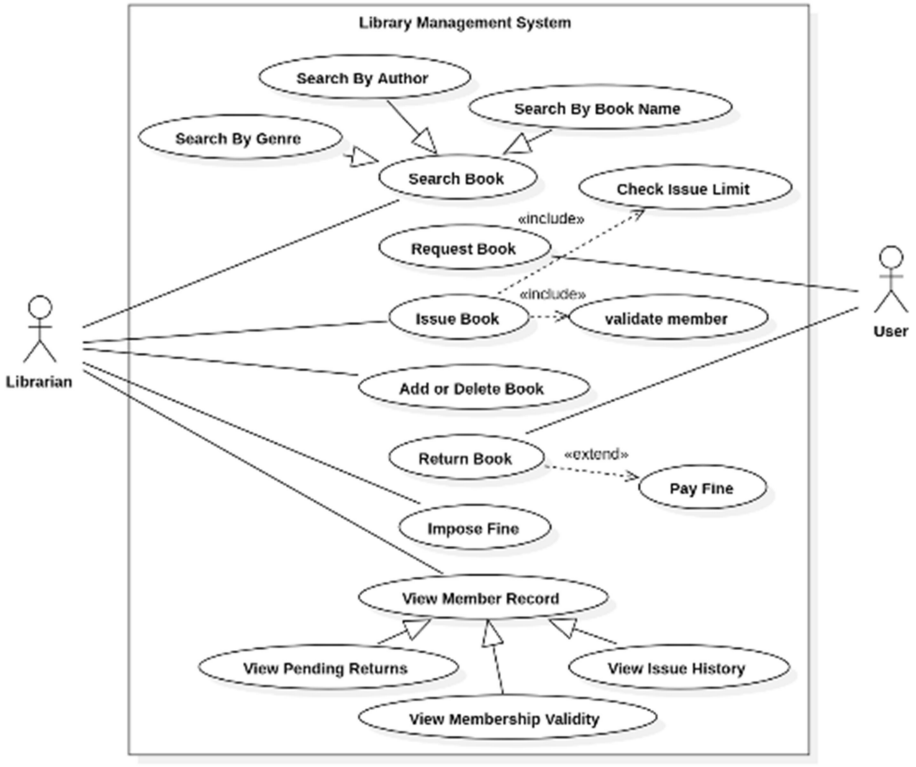


FIG 3.5.1

This UML use case diagram has 2 actors, Librarian and a User. The User can request book and return book. The Librarian can search book, issue book, add or delete book, Impose fine and view member records.

3.6: SEQUENCE DIAGRAM:

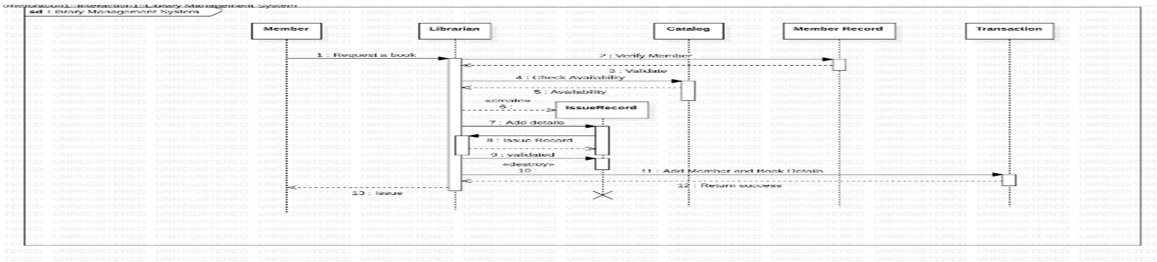


FIG 3.6.1

The UML sequence diagram shows scenario of user issueing a book from the library. The User requests for the book. The Librarian verifies the user, then searches the book if available then the book is issued.

3.7: ACTIVITY DIAGRAM:

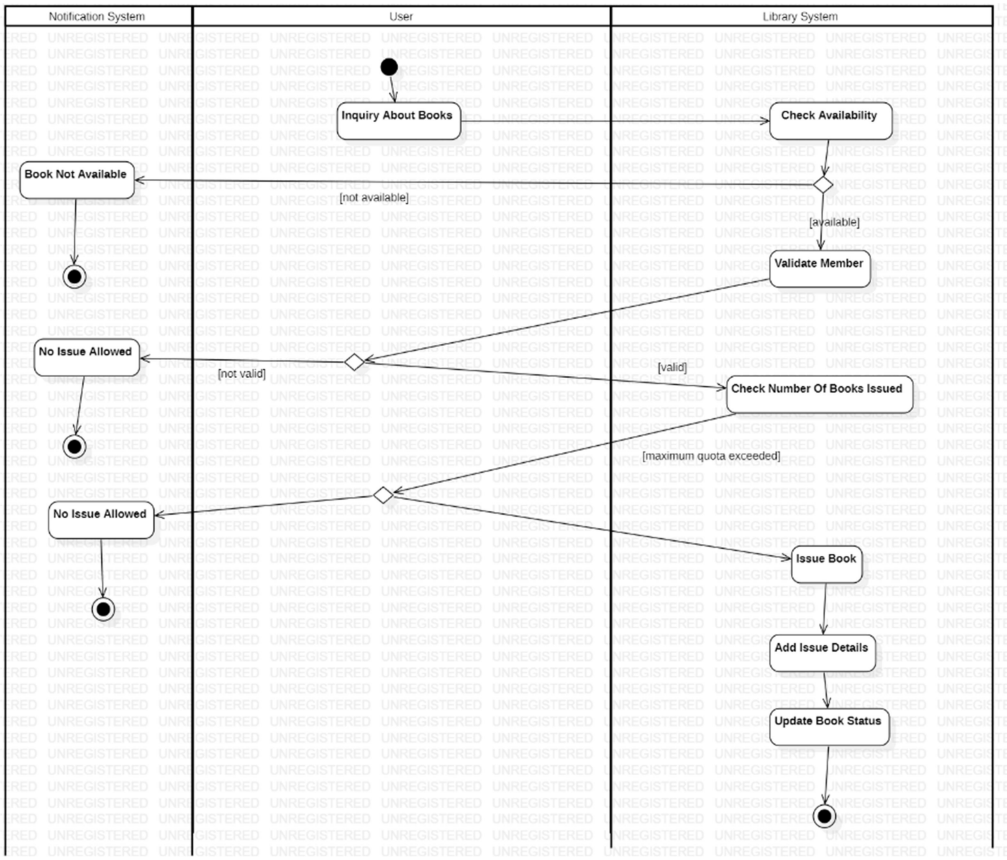


Fig 3.7.1

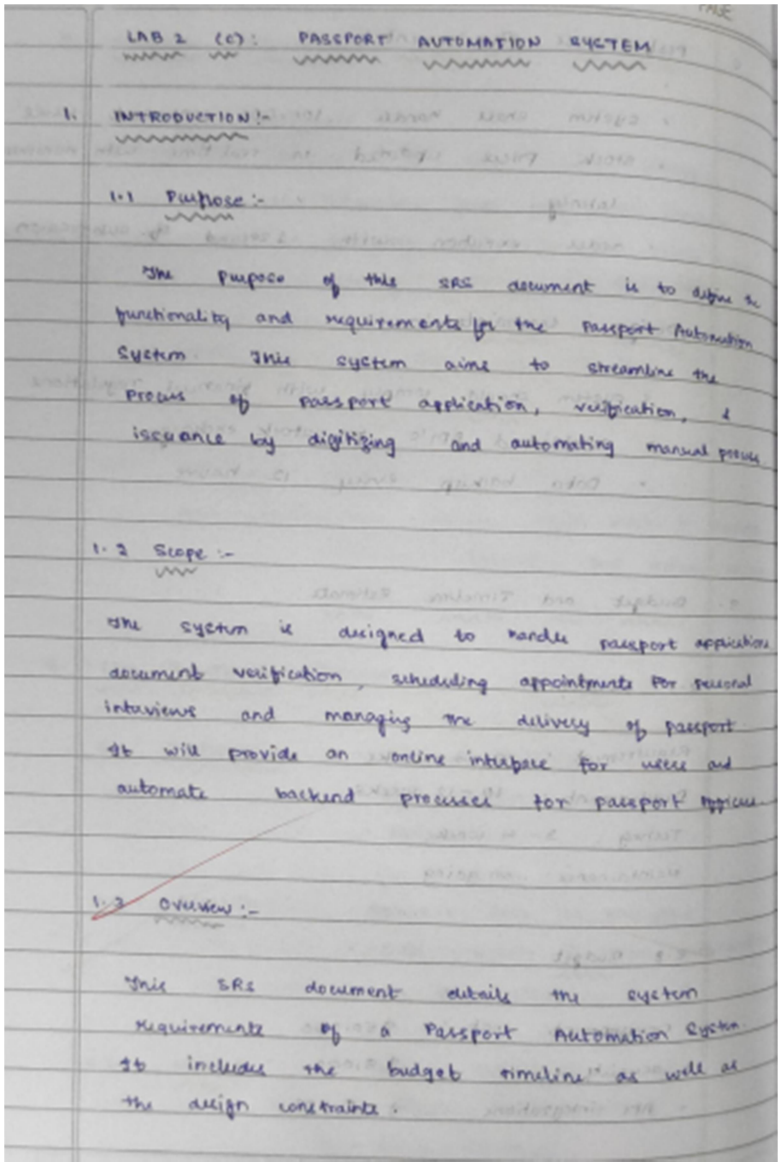
This is an activity diagram for a Library Management System, detailing the process of borrowing a book. It's divided into swimlanes representing the "User," "Library System," and "System Database." The process begins with the "User" requesting to borrow a book. The "Library System" then checks the book's availability in the "System Database." A decision diamond follows: if the book is "not available," a message is displayed to the user. If the book is "available," the "Library System" proceeds to "Issue Book," "Add Issue details," and updates the "book status" in the "System Database." Finally, the "Library System" "Lends the book" to the user, marking the end of the activity. The diagram clearly shows the flow of actions and responsibilities across different parts of the system during a book borrowing transaction, including the handling of book availability.

4. PASSPORT AUTOMATION SYSTEM

4.1: PROBLEM STATEMENT:

Managing passport applications and issuance is a complex process involving multiple stages, including document verification, application tracking, and appointment scheduling. A streamlined system is required to handle these tasks efficiently, minimize errors, and enhance user experience.

4.2: SRS DOCUMENT:



LAB 2 (c): PASSPORT AUTOMATION SYSTEM

1. INTRODUCTION :-

1.1 Purpose :-

The purpose of this SRS document is to define the functionality and requirements for the Passport Automation System. This system aims to streamline the process of passport application, verification, & issuance by digitizing and automating manual process

1.2 Scope :-

The system is designed to handle passport application document verification, scheduling appointments for personal interviews and managing the delivery of passport. It will provide an online interface for users and automate backend processes for passport offices.

1.3 Overview :-

This SRS document details the system requirements of a Passport Automation System. It includes the budget timeline as well as the design constraints.

FIG 4.2.1

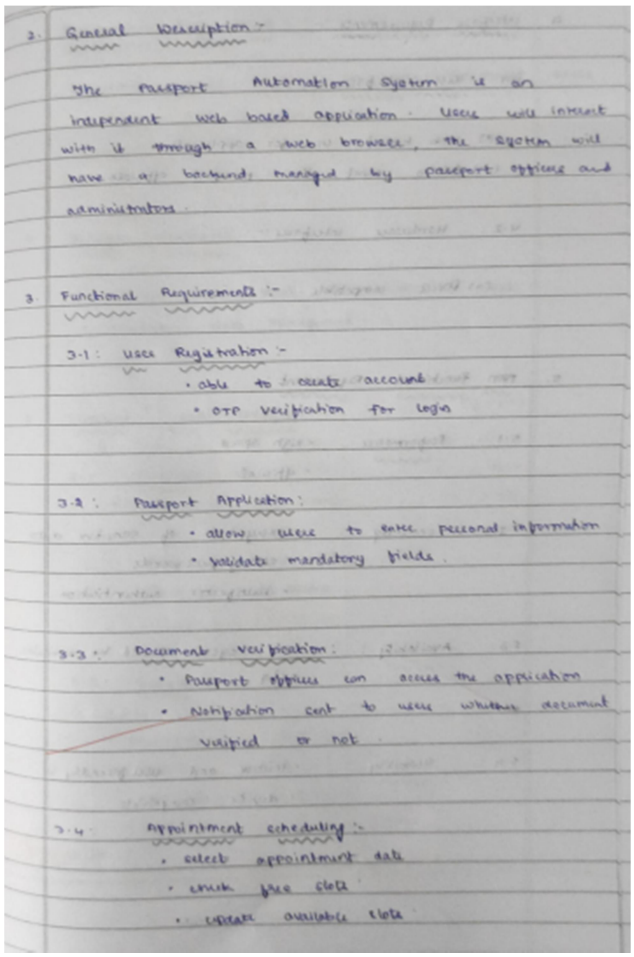


FIG 4.2.2

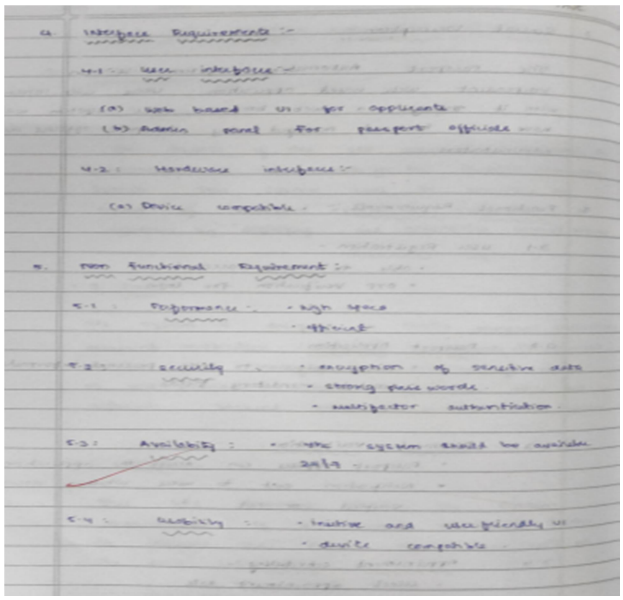


FIG 4.2.3

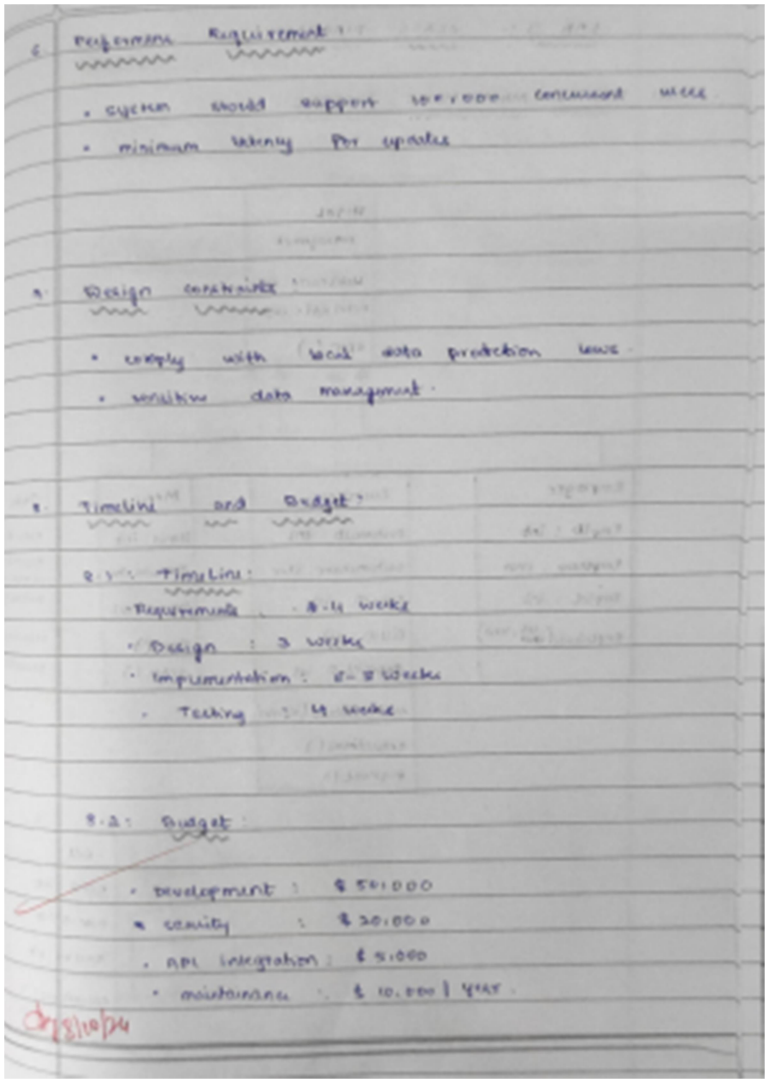

6. Performance Requirement :
• system should support 10k+ concurrent users
• minimum latency for updates
7. Design constraints :
• comply with local data protection laws
• sensitive data management
8. Timeline and Budget :
8.1: Timeline :
• Requirements : 0.5 weeks
• Design : 3 weeks
• Implementation : 6-8 weeks
• Testing : 4 weeks
8.2: Budget :
• Development : $ 50,000
• security : $ 20,000
• API integration : $ 5,000
• maintenance : $ 10,000 / year

FIG 4.2.4

4.3: CLASS DIAGRAM:

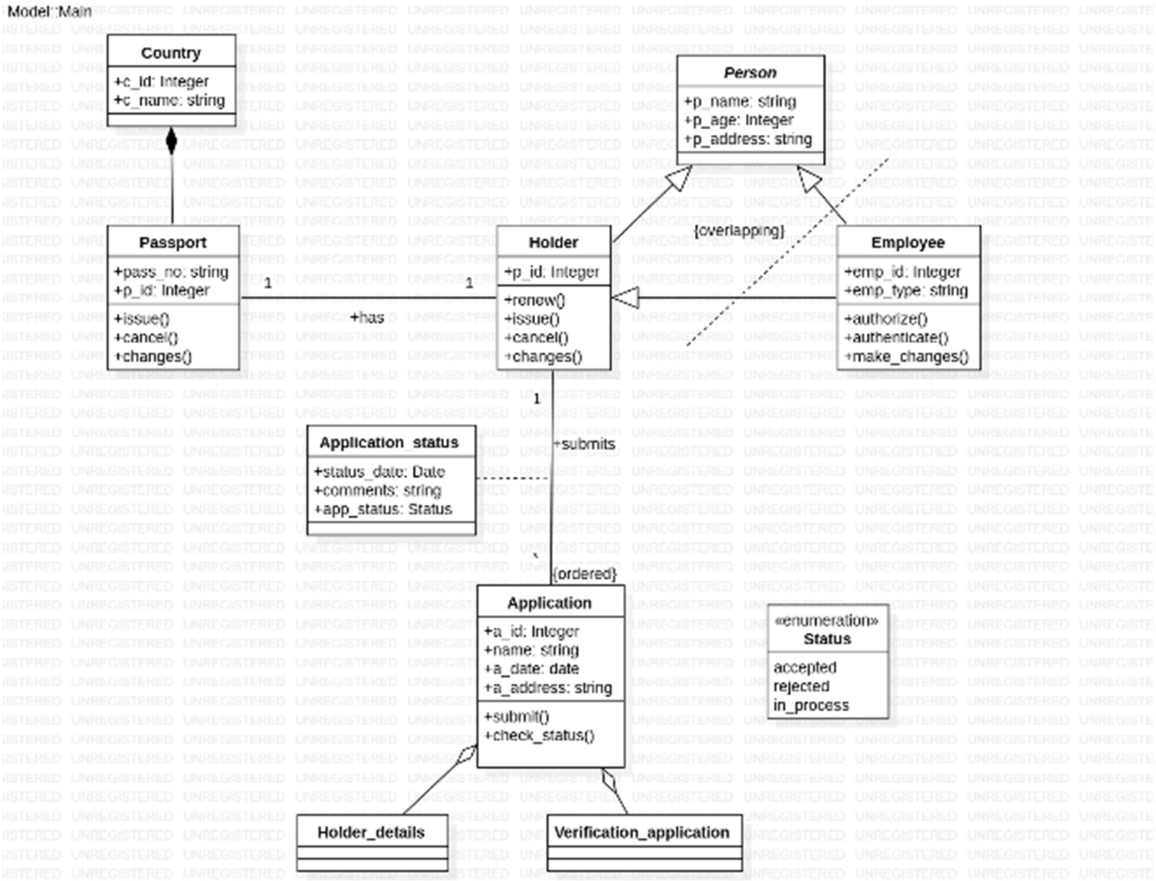


FIG 4.3.1

The Passport Automation System streamlines the end-to-end process of passport application, verification, and issuance by integrating multiple entities and workflows. Applicants can submit various application types, such as initial, renewal, or lost passport applications, each with specific requirements like document verification, fee payments, or police reports for lost passports. Applications are processed through a structured workflow where officers handle verification to ensure compliance. Approved applications result in passport issuance, with details such as passport number, issue, and expiry dates recorded. The system also tracks fee payments and facilitates passport delivery through the embassy. This automation ensures efficiency, transparency, and minimal delays in the passport issuance process.

4.4: STATE DIAGRAM:

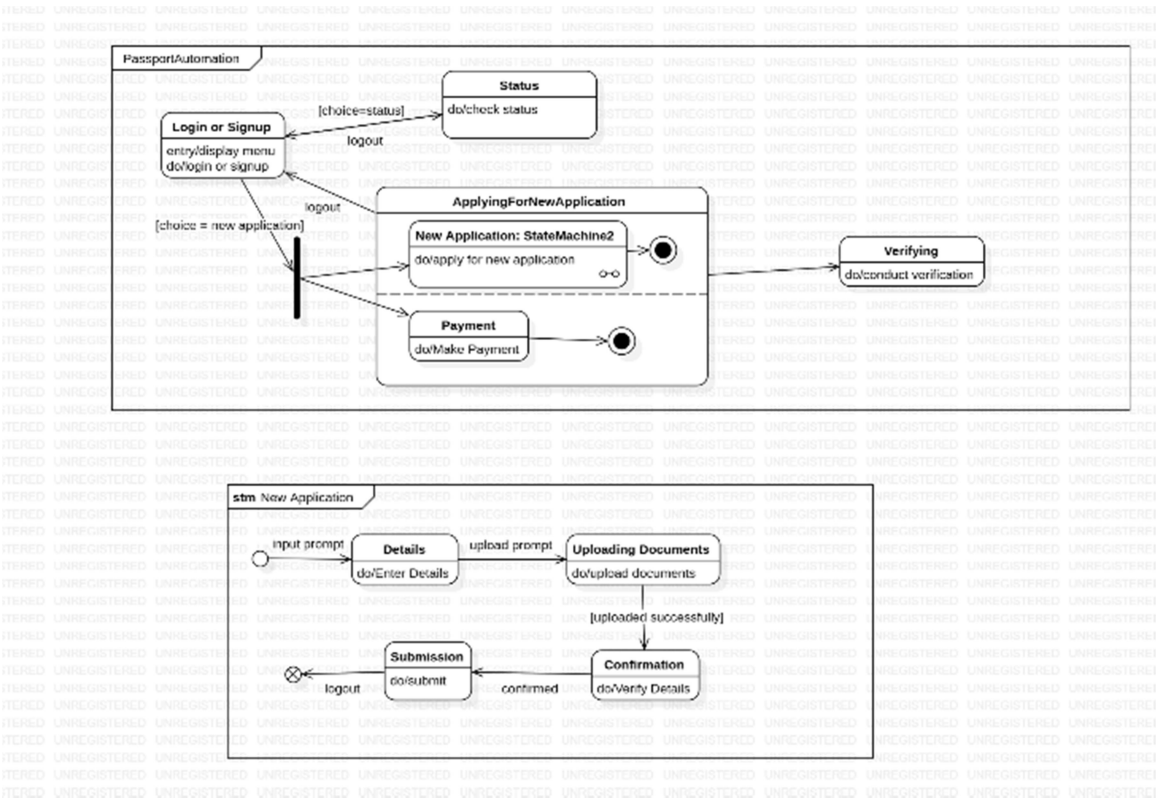


FIG 4.4.1

The diagram depicts the state model for a **Passport Management System** and its integrated **Payment Processing** system. The process begins with the initialization of the system, where users start the passport application process by submitting an application form. Following submission, the system transitions to payment processing, where funds are verified. Upon successful payment, the application proceeds to document submission and verification. If documents are verified successfully, the process moves forward to passport processing, printing, and preparing the passport for shipping. The final stage involves shipping the passport to the user and confirming delivery. In the Payment Processing subsystem, the flow handles various outcomes of payment transactions, including successful payment, failure, or cancellation, with corresponding system updates and user notifications. This state model ensures a structured, step-by-step approach to managing the passport application and delivery lifecycle.

4.5: USE CASE DIAGRAM:

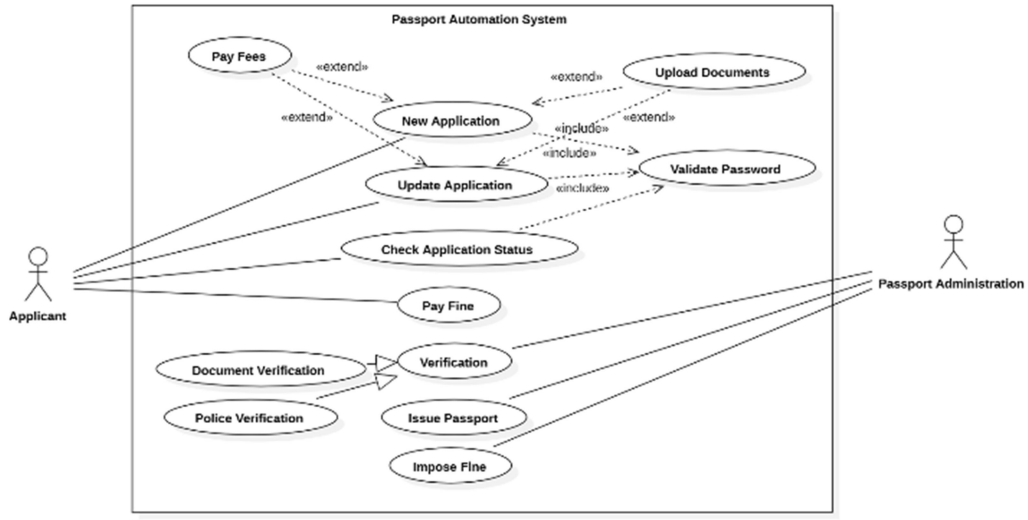


FIG 4.5.1

This use case diagram for a Passport Automation System. It illustrates the interactions between two actors: the Applicant and the Passport Management System. The Applicant can submit a new application, update existing application and check application status. He can also pay fine or fees as necessary. All the above usecases include validating user and password. The Passport Administrator can verify the application and applicant, issue passport and also impose fine.

4.6: SEQUENCE DIAGRAM:

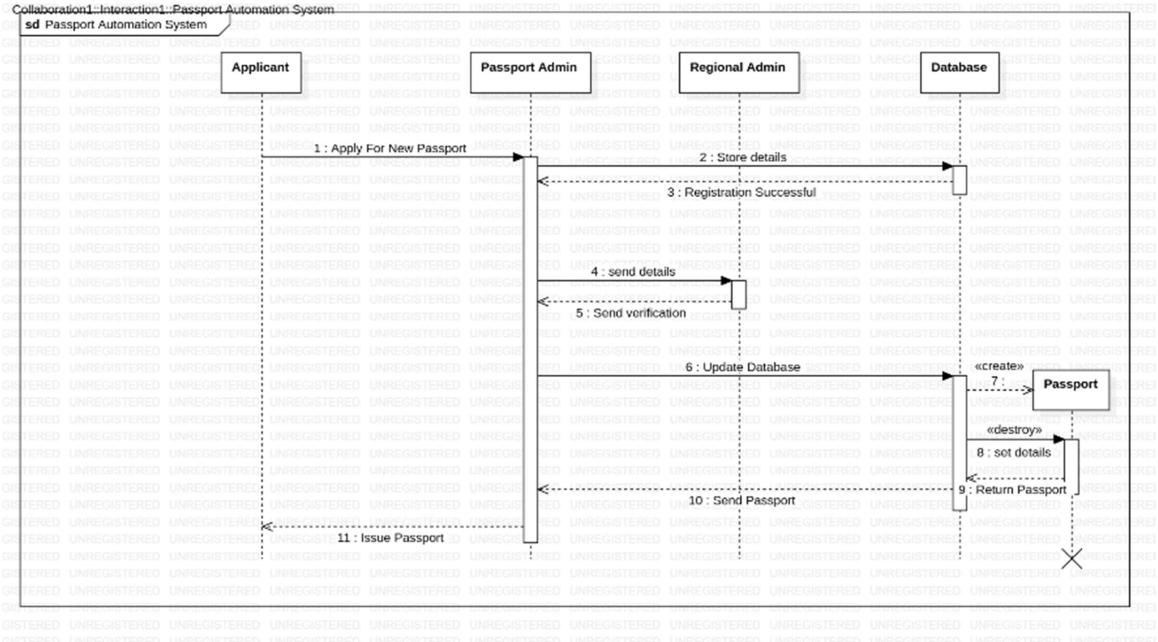


FIG 4.6.1

This showcases the scenario of passport application. The APPLICANT applies for new passport. The PASSPORT ADMIN then verifies the details, stores them an registers the applicant. Then the Passport is issued to the Applicant after police verification.

4.7: ACTIVITY DIAGRAM:

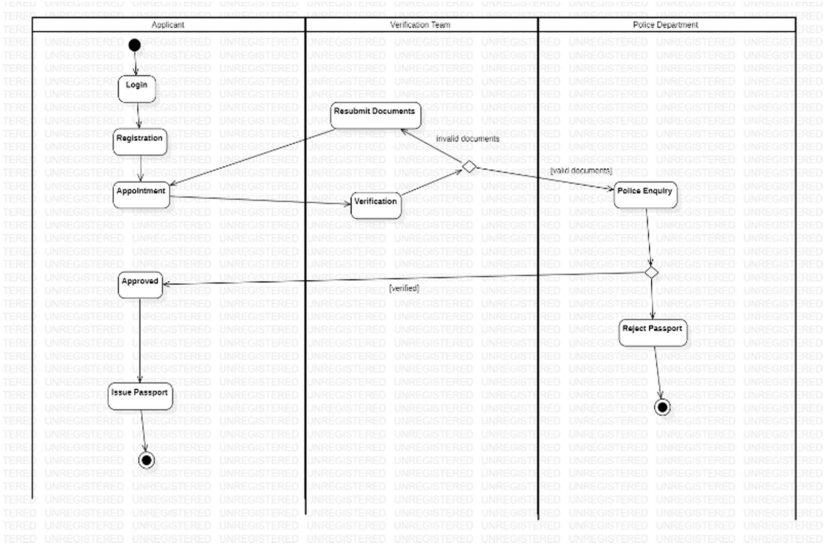


Fig 5.7.1

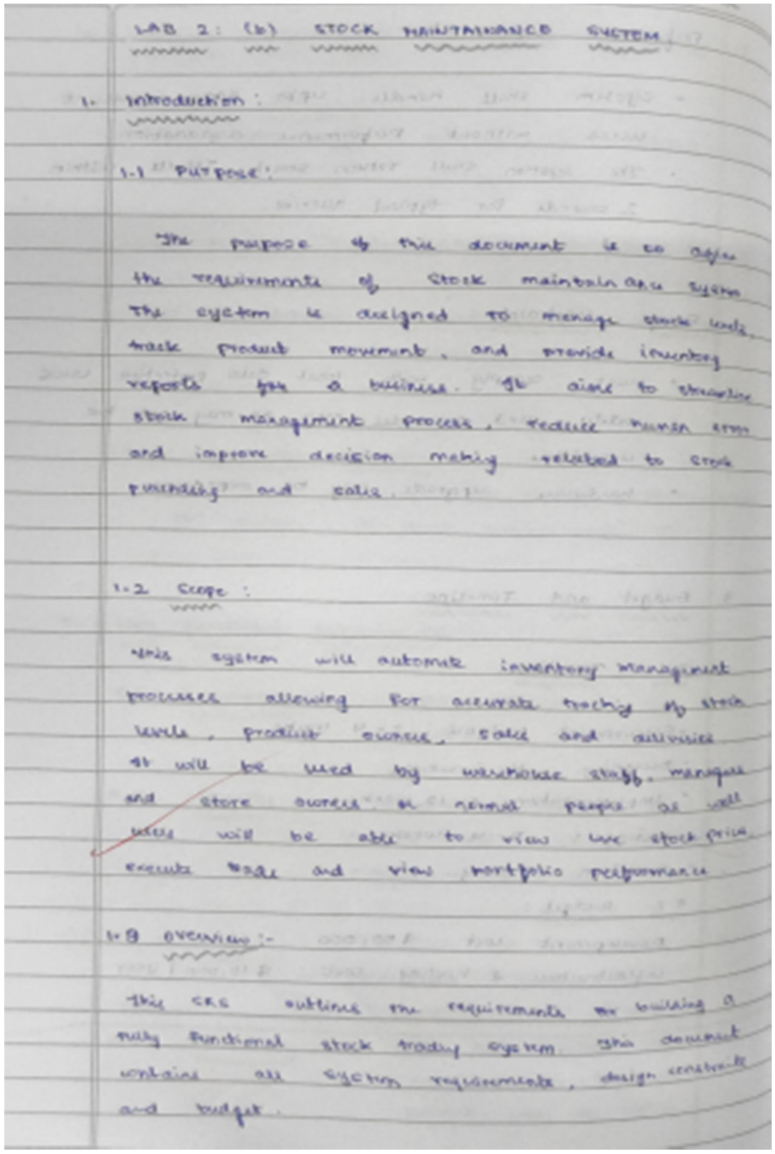
The activity diagram represents the workflow of a Passport Automation System, detailing the process across four primary components: Passport Automation System, Verification, Database, and Payment Gateway. It begins with a user decision between logging in (if an account exists) or signing up for a new account. Once authenticated through credential validation in the Database, the user submits an application. The Verification process involves two rounds of checks, including interaction with an authorities network, to ensure the details are accurate. After both rounds are completed, the application status moves to "Verification Completed." Simultaneously, the Database stores the data and confirms its integrity. In the Payment Gateway, a transaction is generated to complete the process, and upon successful payment, the application is officially submitted. This diagram clearly outlines the sequential and parallel actions, ensuring a comprehensive understanding of the system's operations.

5. STOCK MANAGEMENT SYSTEM

5.1: PROBLEM STATEMENT:

The problem is to create a stock market management system that monitors and analyzes real-time stock prices, manages trading orders, and provides data-driven insights for investors. The system should ensure accurate transaction execution, timely market updates, portfolio management, and risk analysis to assist in making informed investment decisions.

5.2: SRS DOCUMENT:



LAB 2: (b) STOCK MAINTAINANCE SYSTEM

1. Introduction :

1.1 Purpose :

The purpose of this document is to define the requirements of stock maintainance system. The system is designed to manage stock levels, track product movement, and provide inventory reports for a business. It aims to streamline stock management process, reduce human error and improve decision making related to stock purchasing and sales.

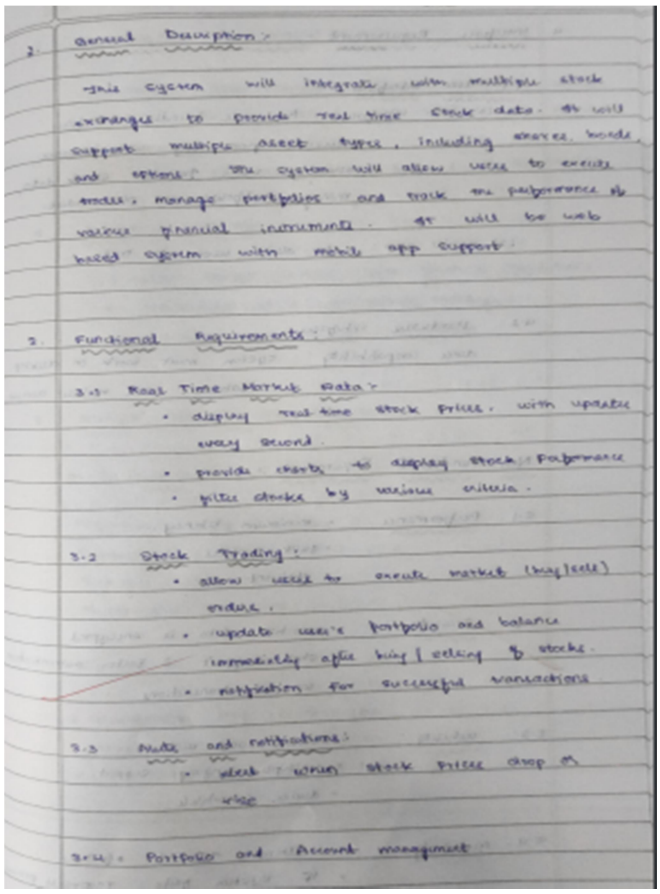
1.2 Scope :

This system will automate inventory management processes allowing for accurate tracking of stock levels, product sources, sales and activities. It will be used by warehouse staff, managers and store owners as normal people as well users will be able to view live stock price execute trade and view portfolio performance.

1.3 Overview :-

This SRS outlines the requirements for building a fully functional stock trading system. This document contains all system requirements, design constraints and budget.

FIG 5.2.1



2. General Description :

This system will integrate with multiple stock exchanges to provide real time stock data. It will support multiple asset types, including shares, bonds, and options. The system will allow users to create trades, manage portfolios and track the performance of various financial instruments. It will be web based system with mobile app support.

3. Functional Requirements :

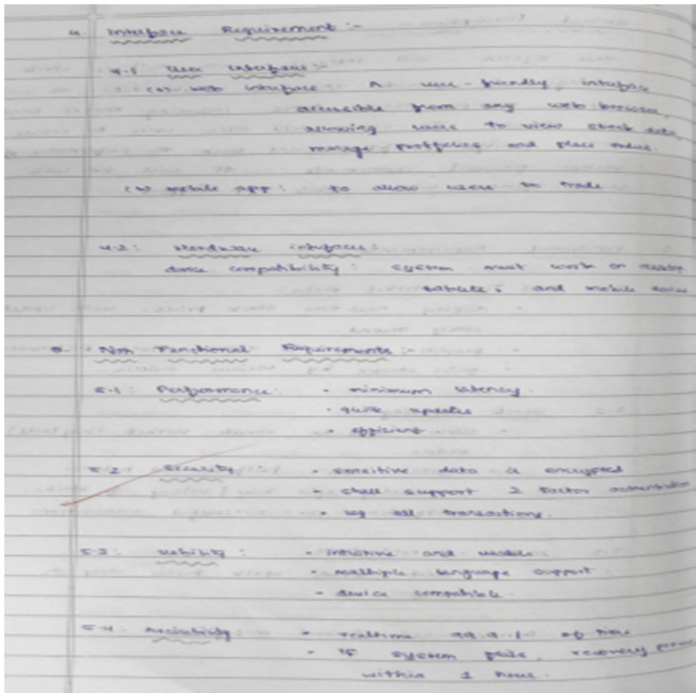
3.1 Real Time Market Data :
- display real time stock prices, with update every second.
- provide charts to display stock performance
- filter stocks by various criteria.

3.2 Stock Trading :
- allow users to execute market (buy/sell) orders.
- update user's portfolio and balance immediately after buying / selling of stocks.
- notification for successful transactions

3.3 Alerts and notifications :
- alert when stock prices drop or rise

3.4 Portfolio and Account management

FIG 5.2.2



4. Interface Requirement :-

4.1 User interface :
(a) web interface : A user-friendly interface accessible from any web browser, allowing users to view stock data, manage portfolios and place orders.
(b) mobile app : to allow users to trade

4.2 Hardware interfaces :
device compatibility : system must work on various tablets, and mobile devices

5. Non Functional Requirements :-

5.1 Performance - minimum latency
- quick speeds
- efficient

5.2 Security - sensitive data is encrypted
- shall support 2 factor authentication
- log all transactions

5.3 Usability - intuitive and modern
- multiple language support
- device compatible

5.4 Availability - reliable 99.9% of time
- if system fails, recovery from within 1 hour

FIG 5.2.3

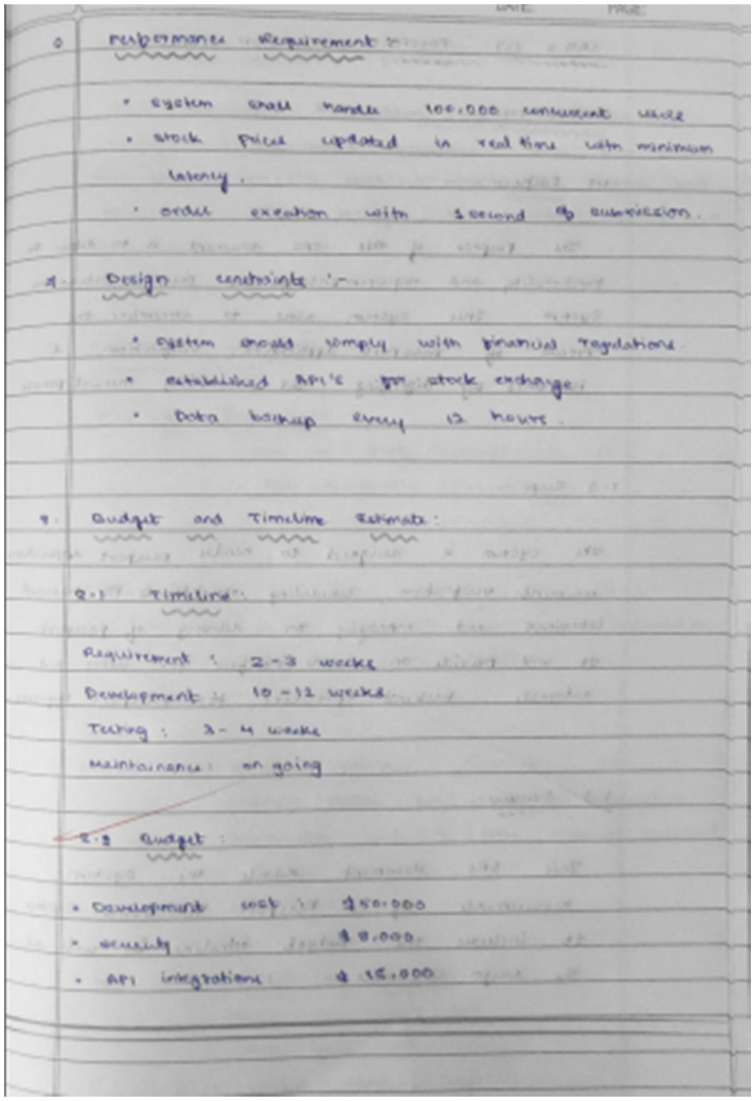

Performance Requirement:
• System shall handle 100,000 concurrent users
• Stock prices updated in real time with minimum latency.
• order execution with 1 second of submission.
Design constraints:-
• System should comply with financial regulations
• established API's for stock exchange.
• Data backup every 12 hours.
Budget and Timeline Estimate:
Timeline:
Requirement: 2-3 weeks
Development: 10-12 weeks
Testing: 3-4 weeks
Maintainance: on going
Budget:
• Development cost: $50,000
• security $8,000
• API integrations: $15,000

FIG 5.2.4

5.3: CLASS DIAGRAM:

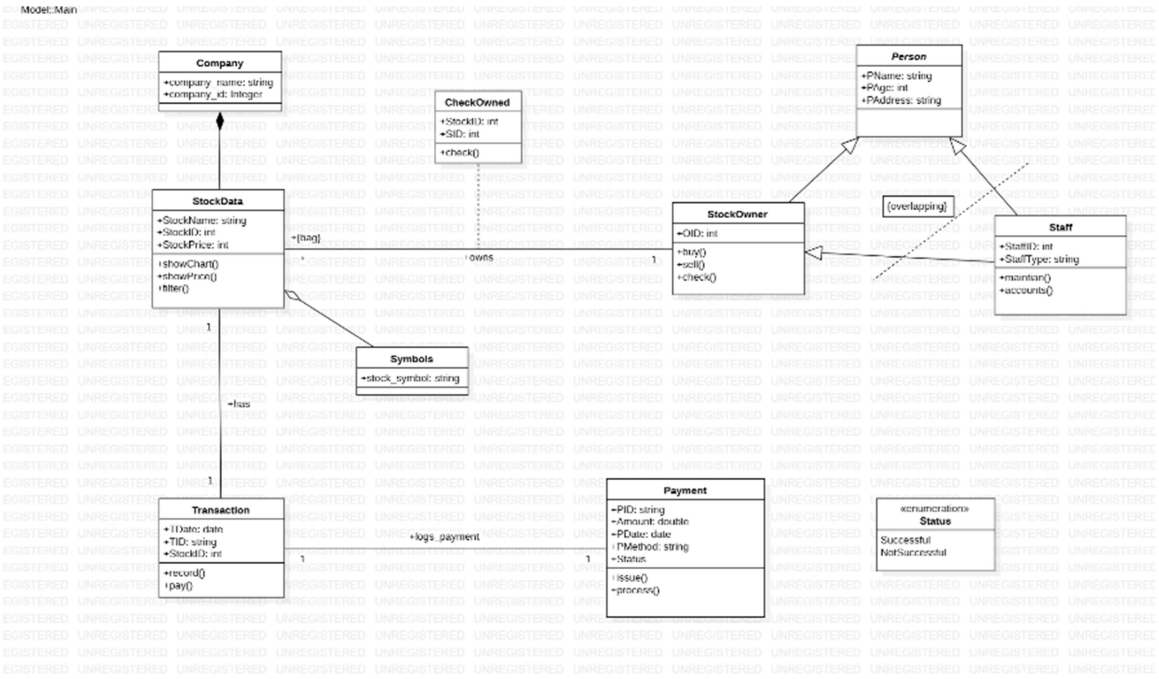


FIG 5.3.1

This UML class diagram shows an abstract class Person having two subclasses- StockOwner and Staff. The stockowners own various stocks shown by the class StockData which are associated through the association class CheckOwned. Each stock is owned by a Company which is related as a composition. Every Stock exchange is recorded through a Transaction log. Status is an enumeration which holds two values successful and not successful.

5.4: STATE DIAGRAM:

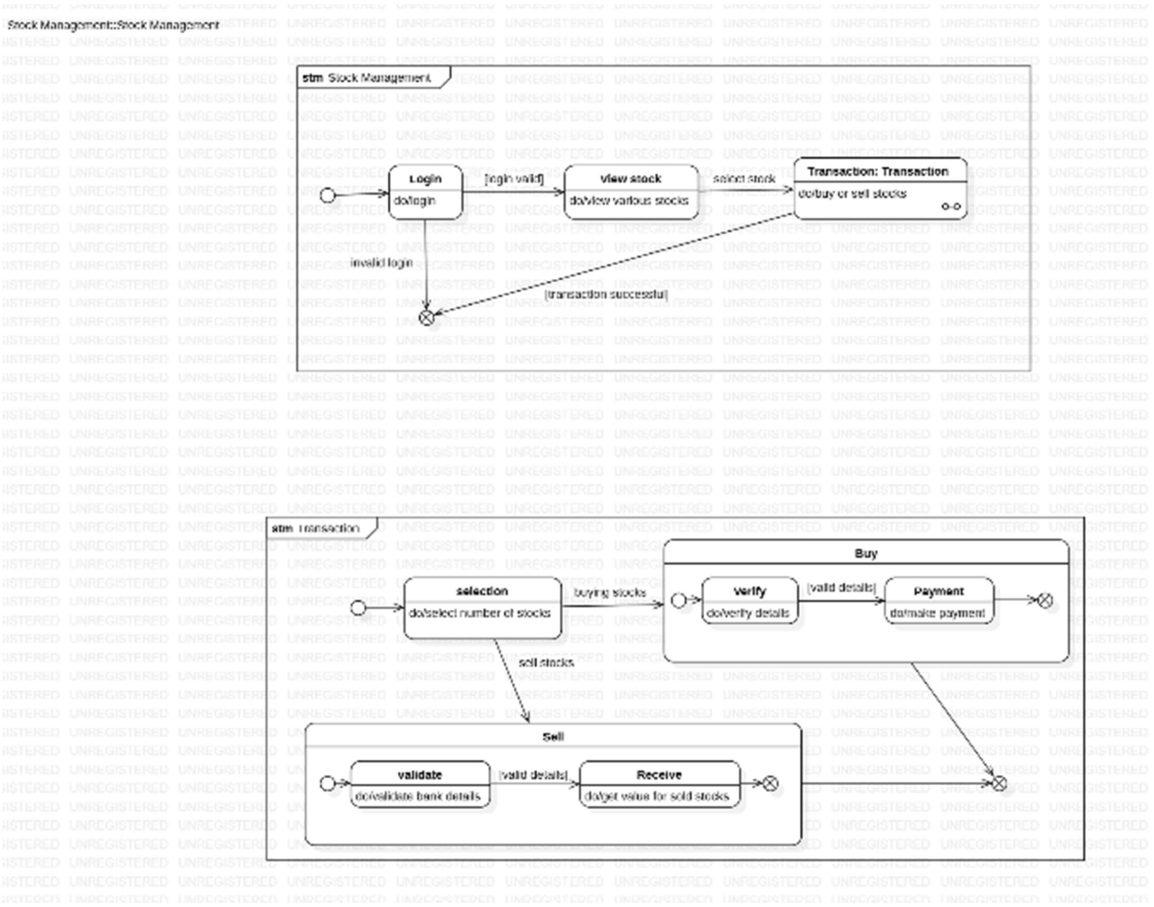


FIG 5.4.1

The provided diagram outlines a state machine model for a stock management system, detailing various interactions and transitions within its components. It begins with a login process, where users are authenticated. Upon successful login, users can either view available stock or initiate a transaction. Transactions include buying and selling stocks, with a selection phase that allows the user to specify the type and quantity of stock.

For buying, the process involves verifying purchase details and making payments if the information is valid. For selling, users must validate their bank details to proceed with receiving the sale value. Each state transitions to the next only if the required conditions are met, ensuring smooth and secure stock management operations.

5.5: USECASE DIAGRAM:

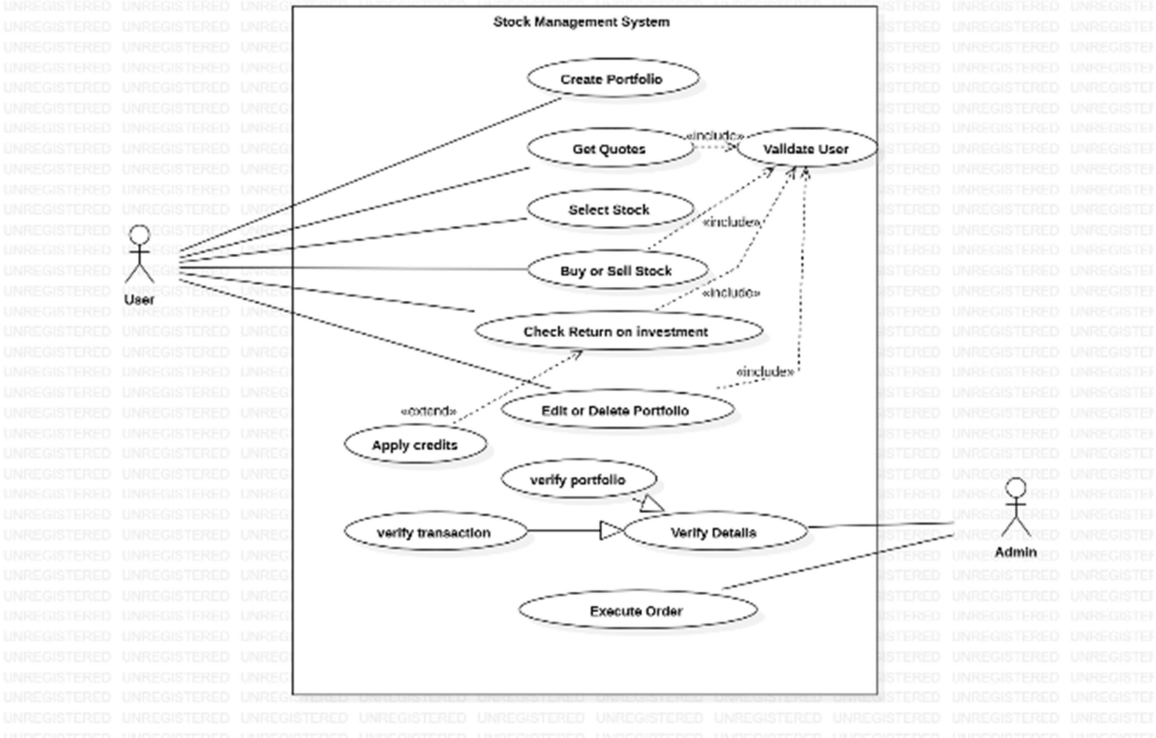


FIG 5.5.1

The provided use case diagram for a Stock Management System illustrates the interactions between two main actors: **User** and **Admin**.

The **User** can create a portfolio, get stock quotes, check the return on investment, verify details, and execute stock orders. The processes of getting quotes, checking ROI, and executing orders are included in the primary activities. Extensions like applying credits may enhance the execution of orders. Meanwhile, the **Admin** is responsible for stock-related operations such as selecting stocks, buying or selling them, editing or deleting portfolios, and validating user credentials. Both actors perform specific actions to ensure the system operates effectively while maintaining the integrity of user transactions and portfolio management.

5.6: SEQUENCE DIAGRAM:

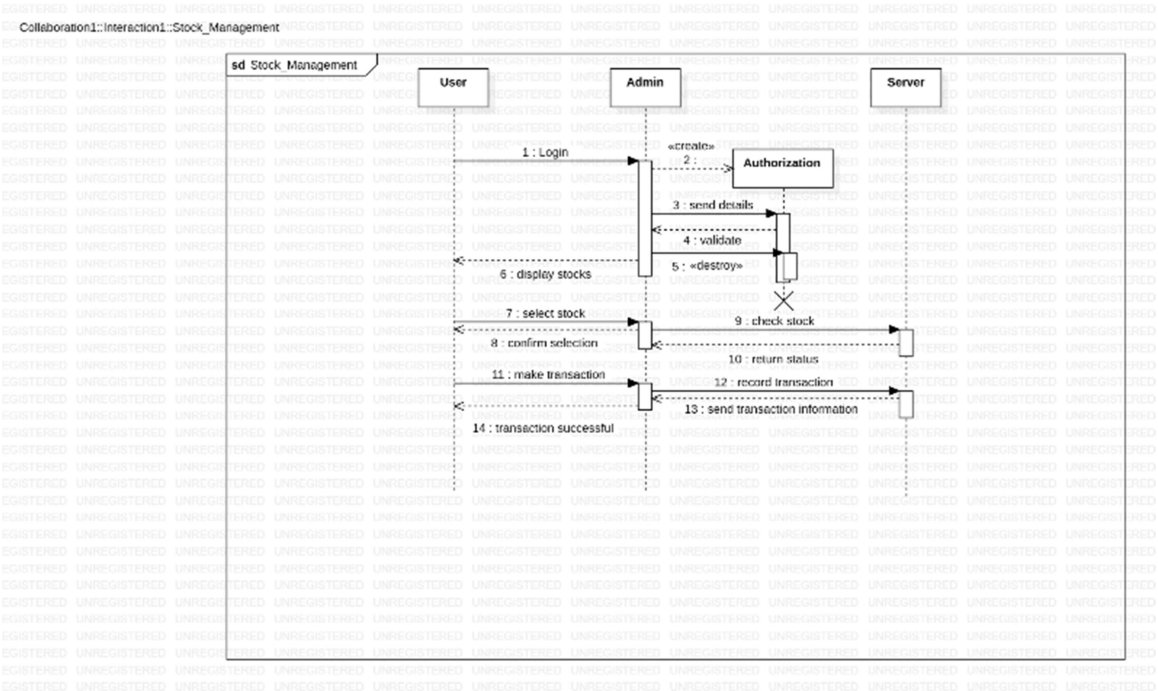


FIG 5.6.1

The document describes an advanced sequence diagram for a stock management system, detailing interactions between components like the user, admin server, and the stock database. It seems to outline processes such as user login, stock validation, stock selection, transaction recording, and confirmation of successful transactions. The sequence involves both the flow of user commands and system responses, showcasing the interplay between authentication, inventory checks, and financial transactions.

The system flow starts with user login authorization and transitions into the user selecting and confirming stocks, where the system validates availability. If the stocks are sufficient, the system processes the transaction, records it, and provides feedback to the user. This structure ensures a robust interaction model that supports real-time stock status updates and secure transaction handling

5.7: ACTIVITY DIAGRAM:

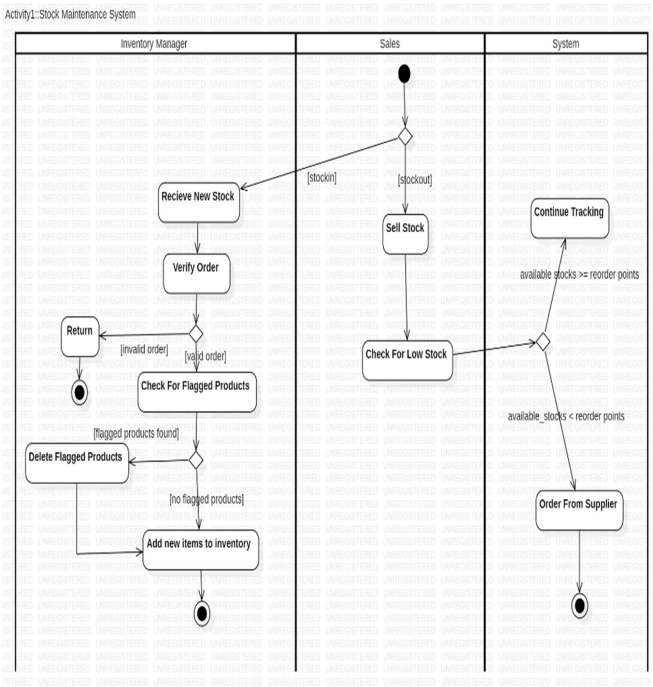


FIG 5.7.1

In the context of the stock market, the activity diagram represents the lifecycle of managing a portfolio and trading stocks. The **Portfolio Manager** begins by evaluating new stocks for investment, ensuring that potential stocks meet the required criteria through the **Validate Stock Purchase Order** step. If stocks are deemed unsuitable, they are rejected, while promising stocks are added to the portfolio. Simultaneously, the **Check for Underperforming Investments** process helps the manager identify and sell underperforming stocks to optimize the portfolio. These actions ensure the portfolio remains healthy and aligned with investment goals. On the other hand, the **Automated Monitoring System** continuously tracks stock performance and portfolio liquidity. It flags low-performing investments and identifies gaps in the portfolio, such as low liquidity or sector imbalances. When stocks fall below a predefined threshold the system generates buy recommendations, ensuring the portfolio remains diversified and profitable. The **Trading Desk** executes trades, managing both the sale of flagged stocks and the purchase of new ones to maintain optimal performance. This integrated workflow ensures seamless portfolio management and effective decision-making in the dynamic stock market environment.