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:

SAMRAAT DABOLAY

1BM22CS236

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 SAMRAAT DABOLAY (1BM22CS236) during the 5th Semester Oct24-Jan2025.

Signature of the Faculty Incharge:

Lakshmi Neelima
Assistant Professor

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

Table of Contents

1. Hotel Management System
2. Credit Card Processing
3. Library Management System
4. Stock Maintenance System
5. Passport Automation System

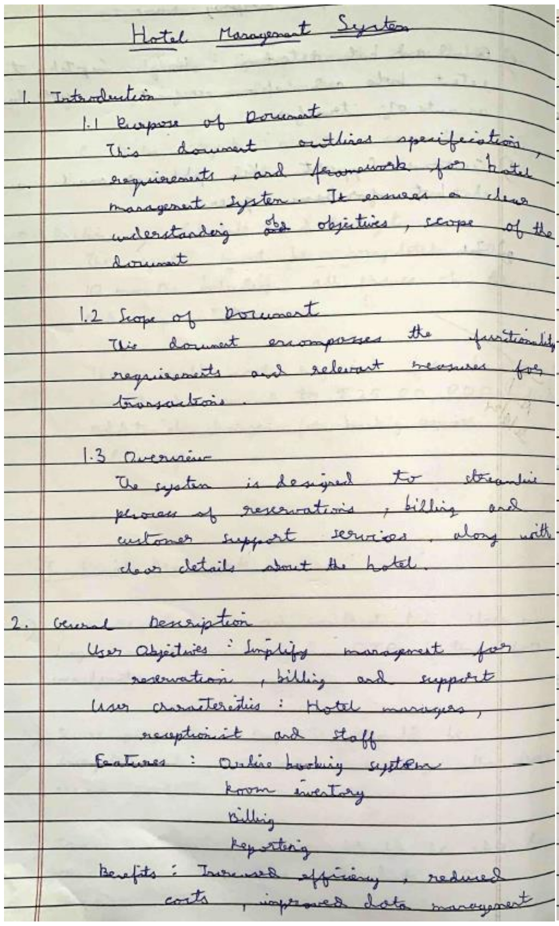
1. Hotel Management System

Problem Statement

Many hotels struggle with inefficient manual processes that lead to booking conflicts, resource allocation issues, billing errors, and poor customer experiences. Current systems are often fragmented, making it difficult to manage reservations, room assignments, and guest services effectively. These challenges result in operational bottlenecks, errors, and dissatisfaction among guests.

The proposed Hotel Management System aims to solve these problems by automating and integrating key functions such as room reservations, guest management, billing, and reporting. This will streamline operations, improve efficiency, and enhance both guest satisfaction and overall hotel performance.

SRS-Software Requirements Specification



Hotel Management System

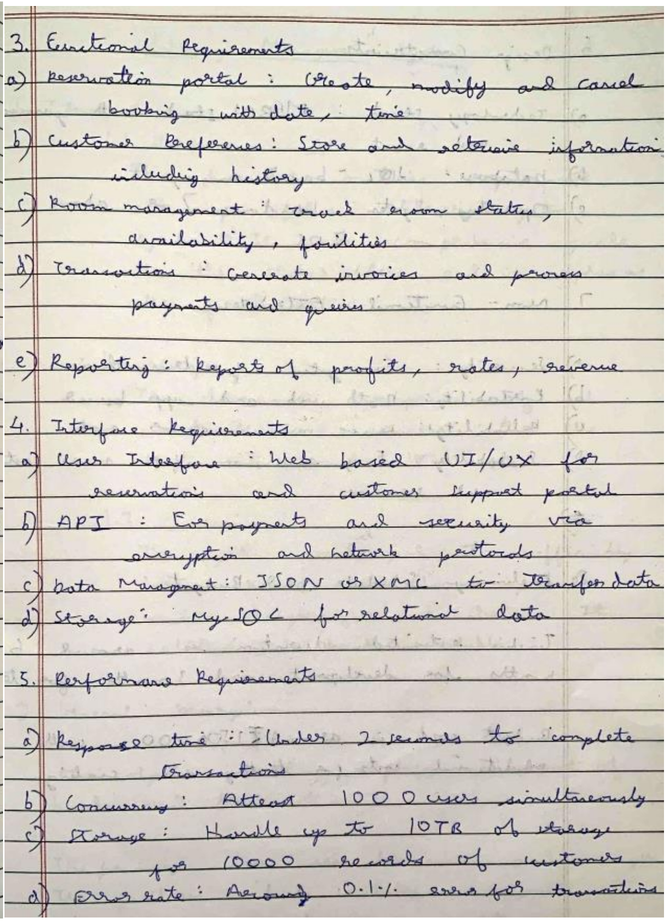
1. Introduction

1.1 Purpose of Document
This document outlines specifications, requirements, and framework for hotel management system. It ensures a clear understanding ~~of~~ objectives, scope of the document

1.2 Scope of Document
This document encompasses the functionality requirements and relevant measures for transactions.

1.3 Overview
The system is designed to streamline process of reservations, billing and customer support services, along with clear details about the hotel.

2. General Description
User Objectives : Simplify management for reservation, billing and support
User characteristics : Hotel managers, receptionist and staff
Features : Online booking system
Room inventory
Billing
Reporting
Benefits : Increased efficiency, reduced costs, improved data management



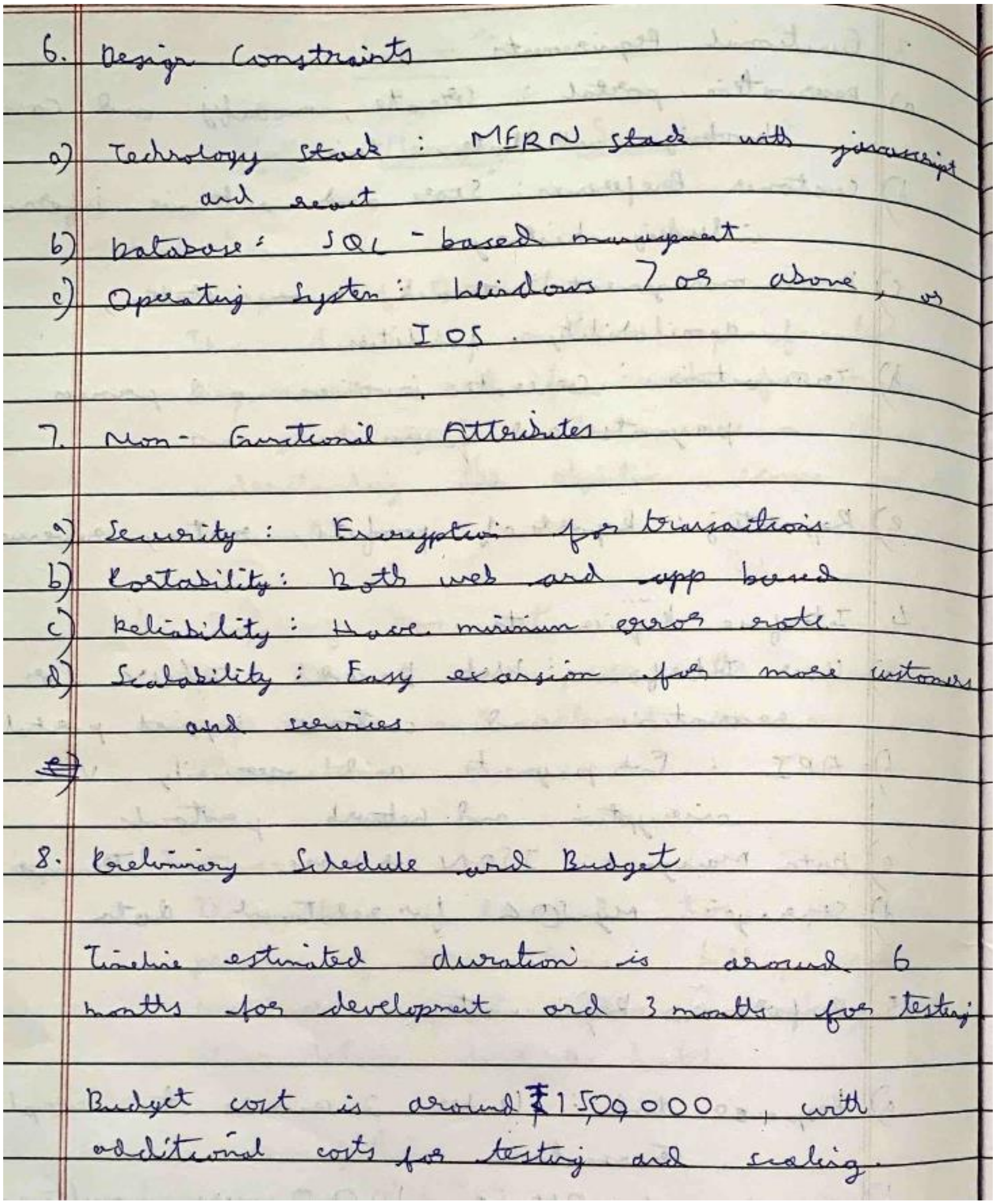
3. Functional Requirements
a) Reservation portal : Create, modify and cancel booking with date, time
b) Customer Preferences : Store and retrieve information including history
c) Room management : track room status, availability, facilities
d) Transactions : Generate invoices and process payments and queries

e) Reporting : Reports of profits, rates, revenue

4. Interface Requirements
a) User Interface : Web based UI/UX for reservations and customer support portal
b) API : For payments and security via encryption and network protocols
c) Data Management : JSON or XML to transfer data
d) Storage : MySQL for relational data

5. Performance Requirements

a) Response time : Under 2 seconds to complete transactions
b) Concurrency : Atleast 1000 users simultaneously
c) Storage : Handle up to 10TB of storage for 10000 records of customers
d) Error rate : Around 0.1% error for transactions



6. Design Constraints

a) Technology stack : MERN stack with javascript and react
b) Database : SQL - based management
c) Operating System : Windows 7 or above, or IOS.

7. Non- Functional Attributes

a) Security : Encryption for transactions
b) Portability : Both web and app based
c) Reliability : Have minimum error rate
d) Scalability : Easy expansion for more customers and services
~~e)~~

8. Preliminary Schedule and Budget

Timeline estimated duration is around 6 months for development and 3 months for testing

Budget cost is around ₹1500,000, with additional costs for testing and scaling.

Class Diagram

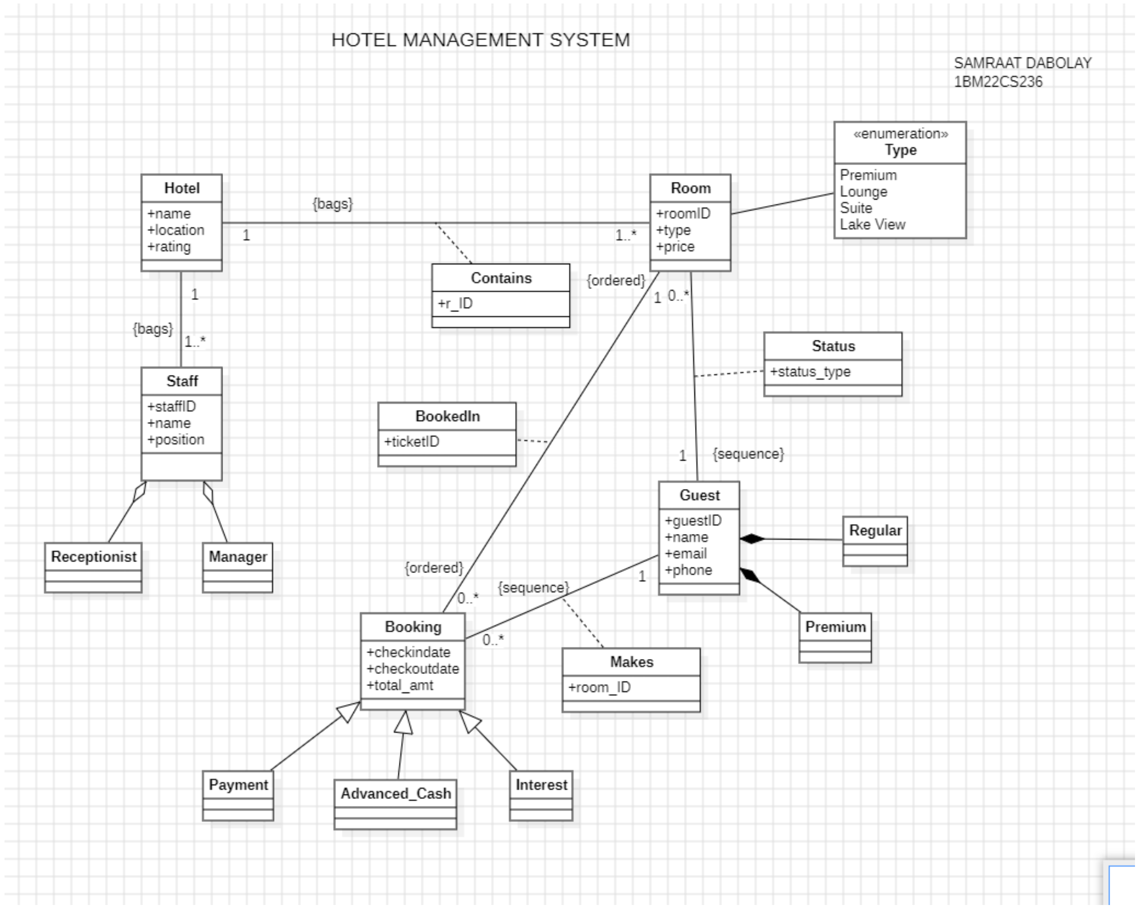


Fig 1.1

The diagram represents 5 main classes: Hotel, Room, Guest, Booking, Staff. Each hotel can have multiple staff. Staff has aggregation consisting of Receptionist and Manager classes. Hotel contains many rooms. Each room has many bookings. A guest can stay in zero or more rooms. Room type has enumeration for premium, lounge, etc. The Room entity encompasses attributes such as roomID, type (potentially an enumeration with values like Premium, Deluxe, Suite), and price. The Staff entity is further categorized into specific roles like Receptionist and Manager, indicating their varying responsibilities within the hotel hierarchy.

The diagram illustrates key relationships within the system. A Hotel entity is associated with multiple Room entities, reflecting the fact that a hotel typically comprises numerous rooms. Staff members have the ability to book Rooms, indicating their involvement in room allocation. Guests are central to the system, as they initiate Bookings. Each Booking entity is linked to a Status entity, signifying the current state of the booking (e.g., confirmed, pending, canceled), and a Payment entity, representing the financial transaction associated with the booking. The Payment entity is further subtyped into Advanced, Cash, and Interest, allowing for a more nuanced categorization of payment methods.

State Diagram

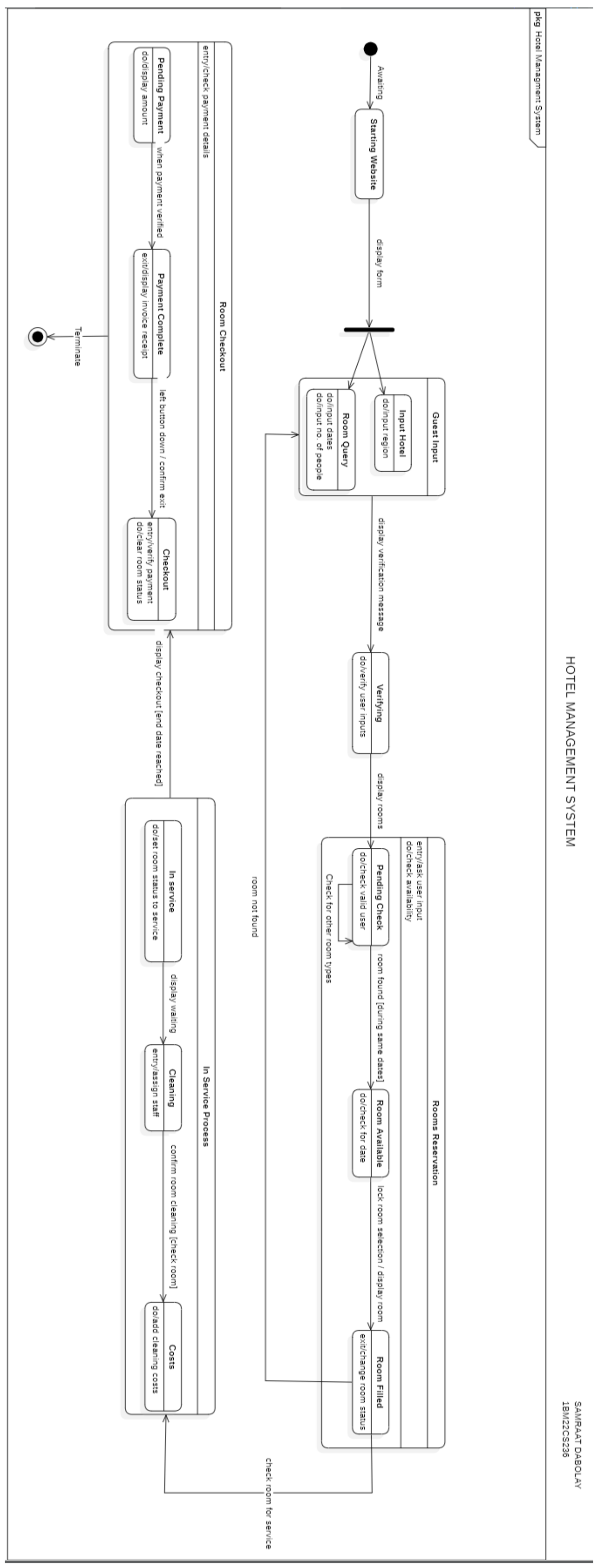


Fig 1.2

The state diagram provides a visual representation of the various states and transitions involved in the hotel management process. It utilizes a combination of basic and advanced features to model the system's behavior.

The diagram employs states to represent different conditions or phases of the system, such as "Room Search," "Room Query," and "Check-in." Transitions, represented by arrows, indicate the movement between these states, often triggered by events or conditions. The diagram includes a start state and an end state to mark the beginning and conclusion of the process flow.

One notable advanced feature is the use of composite states. The "Room Search" state, for example, is a composite state, encompassing substates like "Search Criteria" and "Search Results." This allows for a more detailed representation of the search process within the overall state diagram. While not explicitly shown, the diagram could potentially incorporate history states to capture the previous state of a composite state before it was exited, which can be valuable for certain system behaviors.

Decision points are integral to the diagram, represented by diamonds. These points signify locations where the system needs to make a choice based on specific conditions. For instance, during the "Room Query" state, the system might need to decide whether to display a list of rooms or provide more detailed information about a selected room. Although not explicitly labeled, actions can be associated with transitions to represent activities that occur during the transition. For example, the "Check-in" transition might be associated with the action of updating the room's occupancy status.

Use-Case Diagram

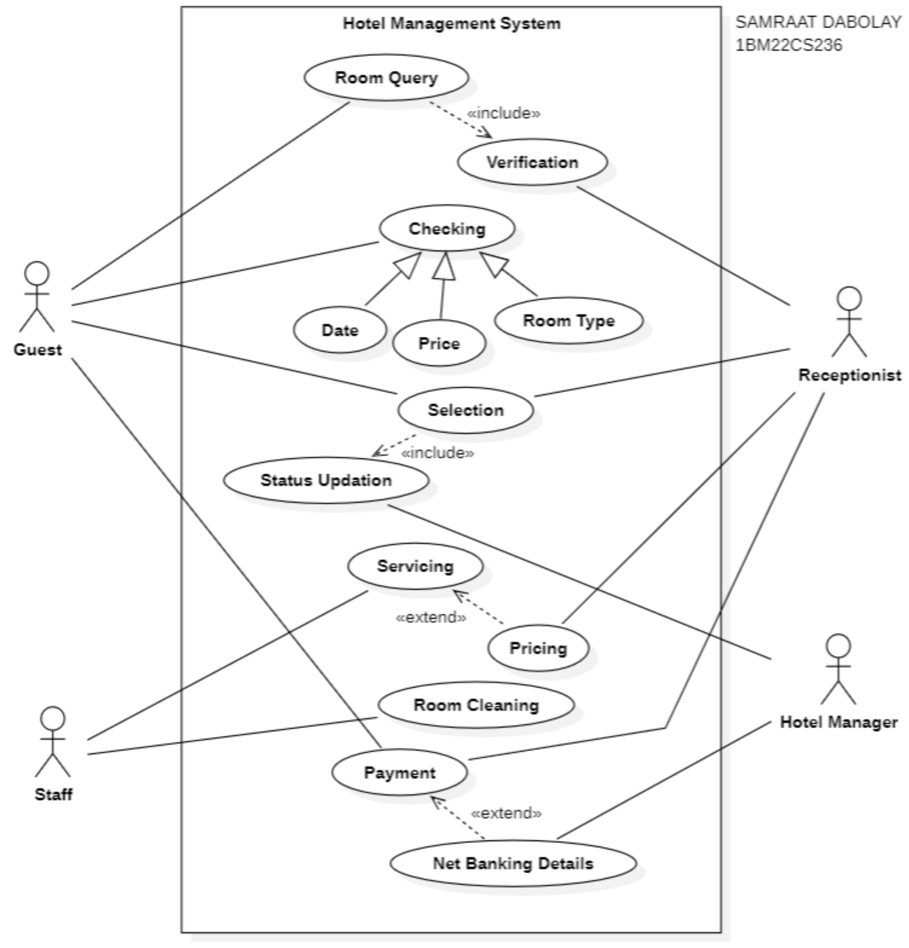


Fig 1.3

The diagram provides a high-level overview of the system's functionality by illustrating the interactions between various actors and use cases. Actors such as Guest, Receptionist, and Hotel Manager are involved in different aspects of the hotel operations. Use cases like "Room Query," "Checking," "Payment," and "Servicing" represent the key activities performed within the system.

The diagram demonstrates several advanced relationships between actors and use cases. For instance, the Staff actor is generalized into more specific roles like Receptionist and Hotel Manager, indicating that these roles inherit common properties and behaviors from the general Staff class. Furthermore, the Status Updation use case includes the Selection use case, implying that the Selection use case is a mandatory prerequisite for the Status Updation process. Additionally, the Servicing use case is extended by Pricing and Room Cleaning, suggesting that these are optional or alternative steps that may be performed during the servicing process.

Sequence Diagram

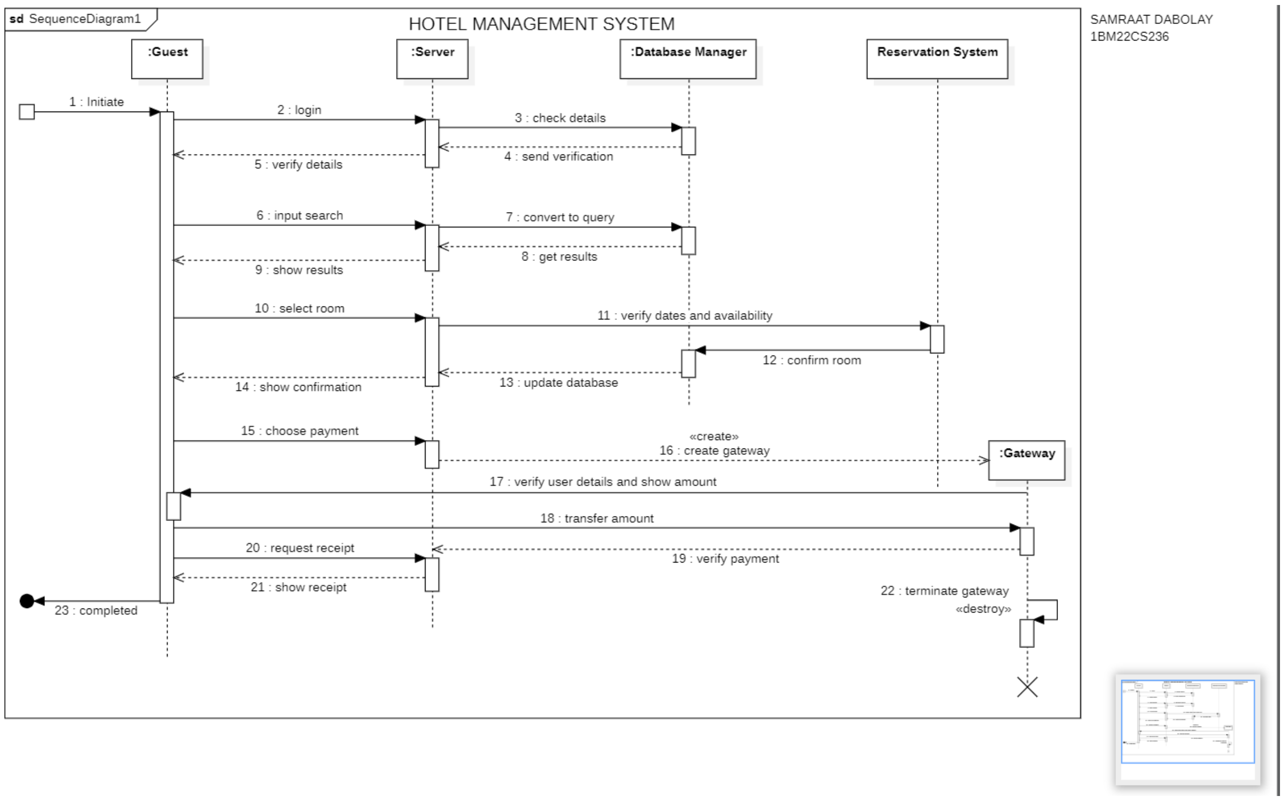


Fig 1.4

The sequence diagram illustrates the step-by-step interaction between various components of the system when a guest makes a room reservation. The process begins with the Guest initiating the reservation by logging into the system. The Server then verifies the Guest's credentials and sends a verification code. Once verified, the Guest enters their search criteria, such as dates and room preferences. The Server converts this information into a database query, which is then processed by the Database Manager to retrieve matching rooms. The Server displays the available rooms to the Guest, who then selects their preferred option.

The Reservation System verifies the availability of the selected room and confirms the reservation if available. The Database Manager updates the room's availability status accordingly. The Server then displays a confirmation message to the Guest and guides them through the payment process. The Server creates a connection to a payment gateway, where the Guest enters their payment details. The gateway verifies the payment information and processes the transaction. Once the payment is successful, the Server displays a receipt to the Guest and terminates the gateway connection. Finally, the reservation process is completed successfully.

Activity Diagram

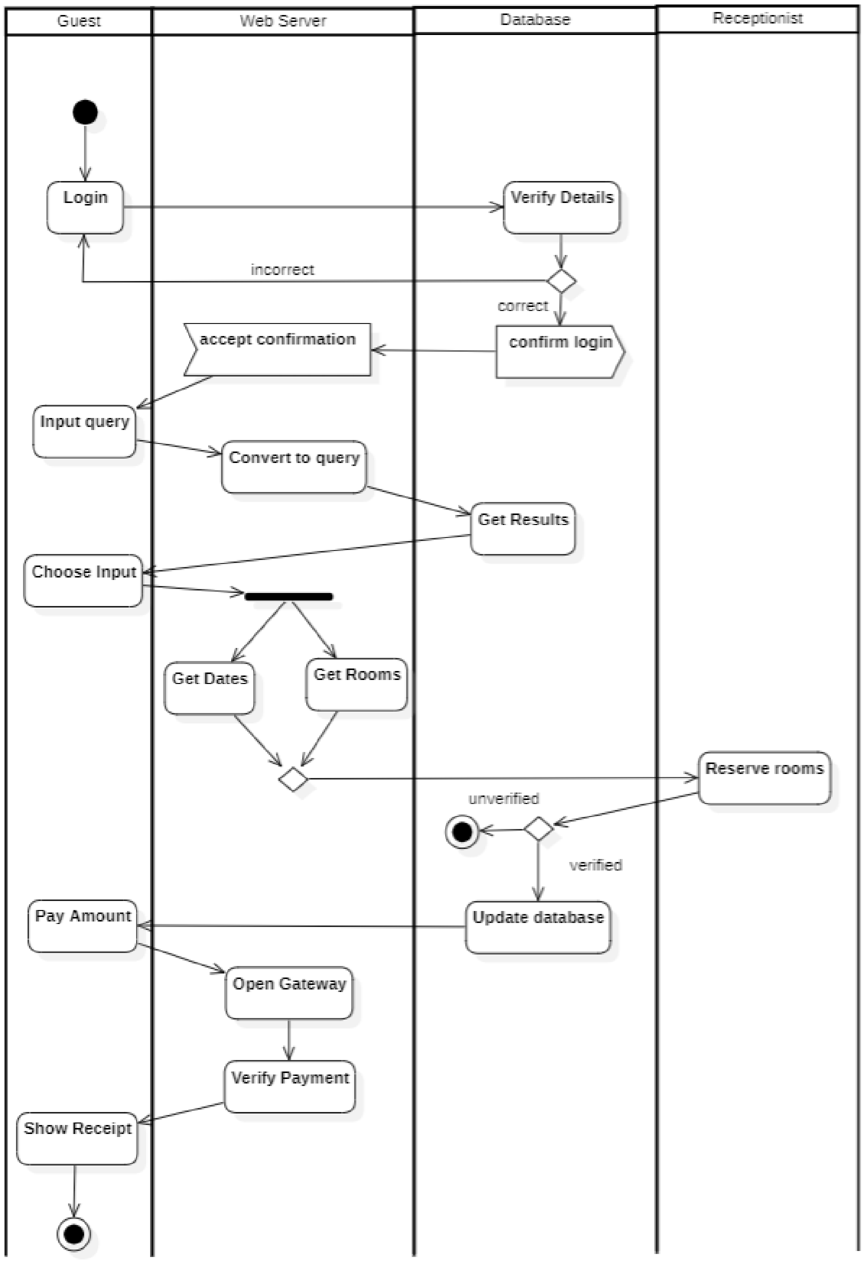


Fig 1.5

This Activity Diagram illustrates the workflow involved in the hotel room reservation process. It depicts the sequence of activities performed by various actors such as the Guest, Web Server, Database, and Receptionist.

The process begins with the Guest initiating the login. The Web Server then verifies the Guest's credentials with the Database. Upon successful login, the Guest enters their search criteria, such as desired dates and room type. The Web Server translates this input into a query for the Database, which retrieves the available rooms. The Web Server presents the search results to the Guest, allowing them to select their preferred room(s).

The Guest's selection is then sent to the Receptionist for verification. The Receptionist checks the availability and confirms the reservation, subsequently updating the database. The Guest is then prompted to pay for the reservation. The Web Server facilitates the payment process by opening a payment gateway. Once the payment is successful, the Web Server displays a receipt to the Guest, confirming the successful reservation.

The Activity Diagram effectively visualizes the flow of activities and decision points within the reservation process, providing a clear overview of the system's workflow.

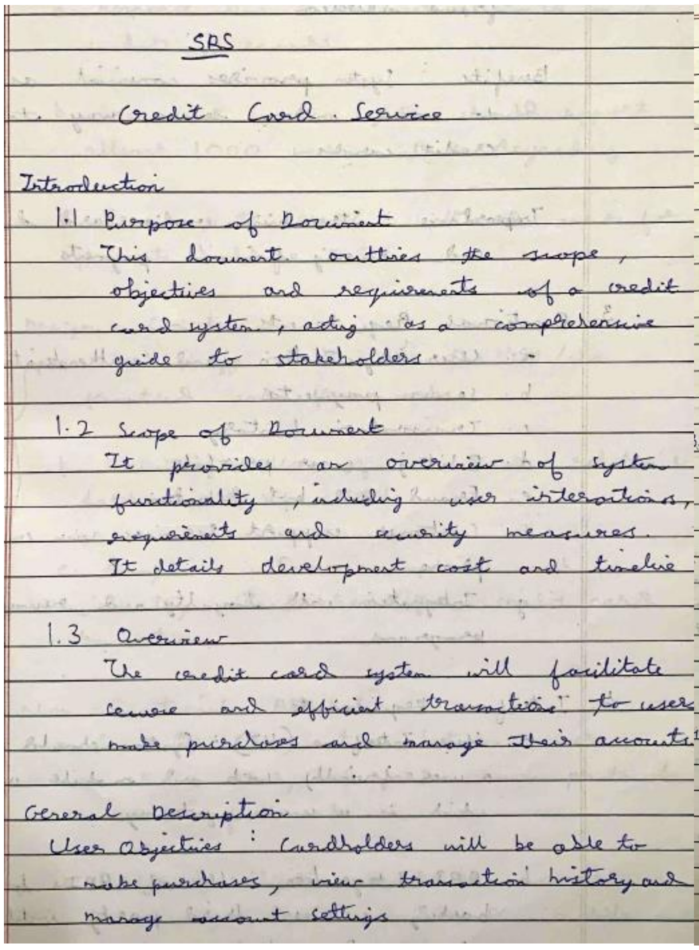
2. Credit Card Processing System

Problem Statement

Online businesses face challenges with secure, efficient, and reliable credit card processing. Current systems may suffer from issues like slow transaction times, high error rates, security vulnerabilities, and a lack of integration with various payment methods. These problems can lead to payment failures, security breaches, and a poor customer experience.

The proposed Credit Card Processing Online System aims to address these issues by offering a secure, fast, and reliable payment solution. The system will ensure seamless transactions, enhance security features, reduce errors, and provide integration with multiple payment gateways, ultimately improving the user experience and operational efficiency.

SRS-Software Requirements Specification



SRS

Credit Card Service

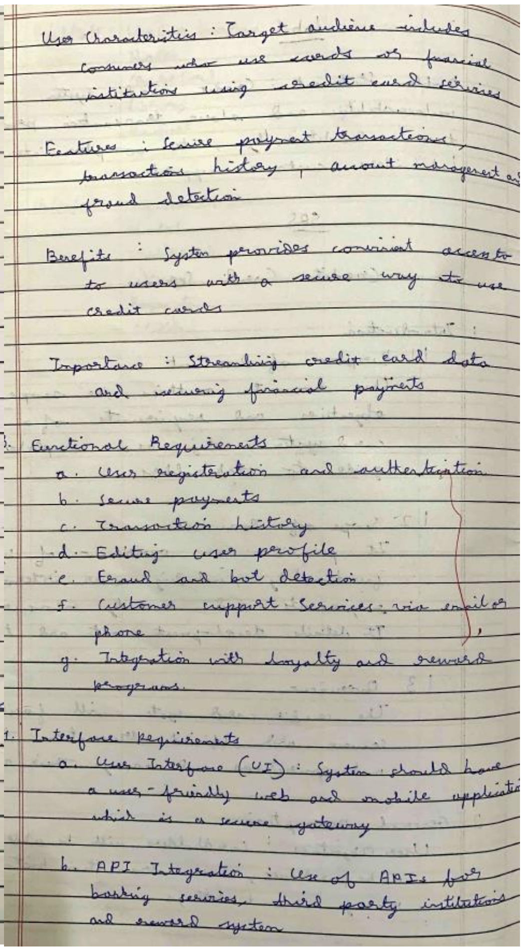
Introduction

1.1 Purpose of Document
This document outlines the scope, objectives and requirements of a credit card system, acting as a comprehensive guide to stakeholders.

1.2 Scope of Document
It provides an overview of system functionality, including user interactions, requirements and security measures.
It details development cost and timeline

1.3 Overview
The credit card system will facilitate secure and efficient transactions to users make purchases and manage their accounts

General Description
User Objectives : Cardholders will be able to make purchases, view transaction history and manage account settings



User Characteristics : Target audience includes consumers who use cards or financial institutions using credit card services

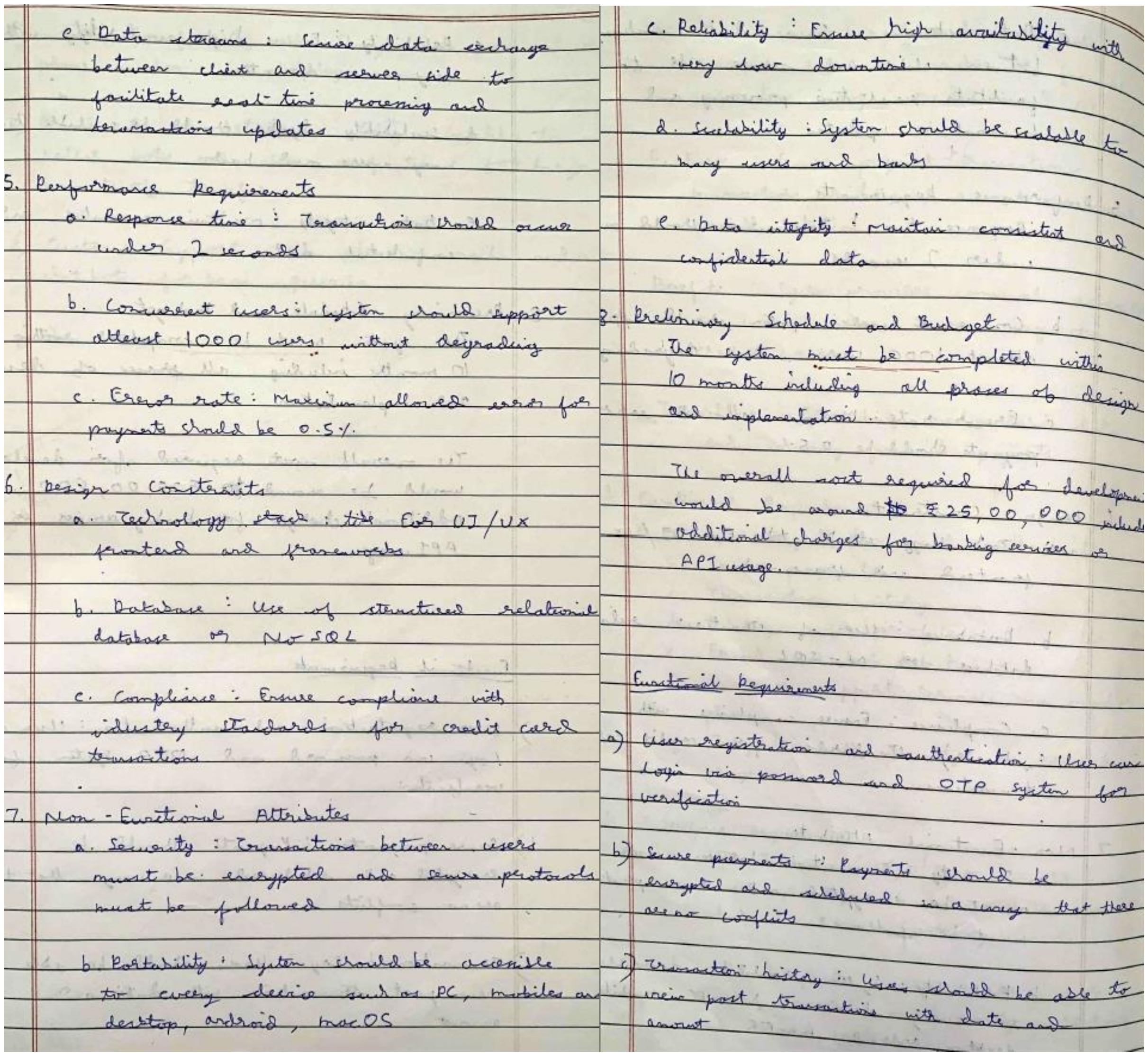
Features : Secure payment transactions, transaction history, account management and fraud detection

Benefits : System provides convenient access to to users with a secure way to use credit cards

Importance :: Streamlining credit card data and securing financial payments

3. Functional Requirements
a. User registration and authentication
b. Secure payments
c. Transaction history
d. Editing user profile
e. Fraud and bot detection
f. Customer support Services via email or phone
g. Integration with loyalty and reward programs.

4. Interface Requirements
a. User Interface (UI) : System should have a user-friendly web and mobile application which is a secure gateway
b. API Integration : Use of APIs for banking services, third party institutions and reward system


c. Data streams : Secure data exchange between client and server side to facilitate real-time processing and transactions updates

5. Performance Requirements

a. Response time : Transactions should occur under 2 seconds

b. Concurrent users: System should support atleast 1000 users without degrading

c. Error rate: Maximum allowed error for payments should be 0.5%.

6. Design Constraints

a. Technology stack : tile for UI/UX frontend and frameworks

b. Database : Use of structured relational database or No-SQL

c. Compliance : Ensure compliance with industry standards for credit card transactions

7. Non-Functional Attributes

a. Security : Transactions between users must be encrypted and secure protocols must be followed

b. Portability : System should be accessible to every device such as PC, mobiles and desktop, android, macOS

c. Reliability : Ensure high availability with very low downtime.

d. Scalability : System should be scalable to many users and banks

e. Data integrity : maintain consistent and confidential data

8. Preliminary Schedule and Budget

The system must be completed within 10 months including all phases of design and implementation.

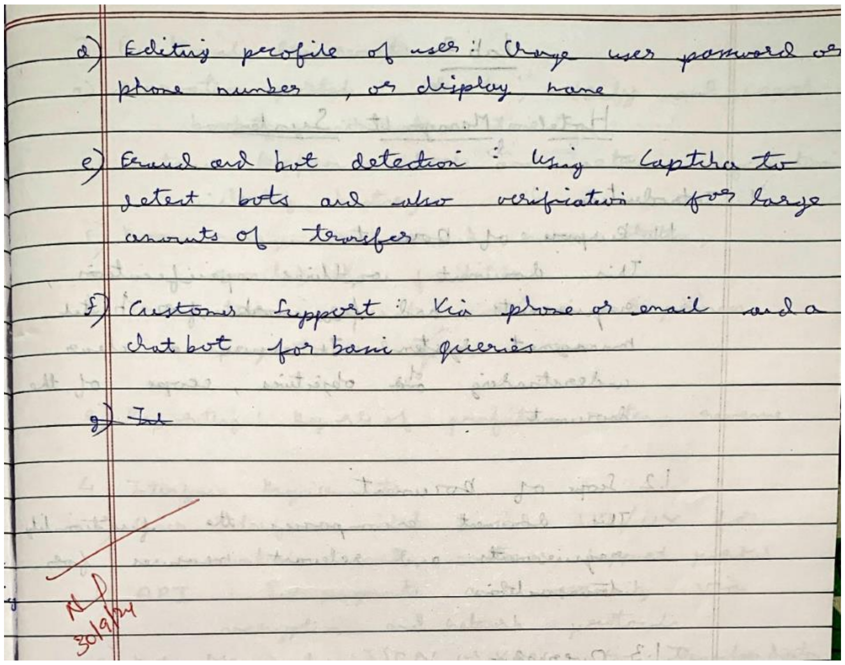
The overall cost required for development would be around ~~$~~ ₹25,00,000 include additional charges for banking services or API usage.

Functional Requirements

a) User registration and authentication : Users can Login via password and OTP system for verification

b) Secure payments : Payments should be encrypted and scheduled in a way that there are no conflicts

c) Transaction history : Users should be able to view past transactions with date and amount


d) Editing profile of users : Change user password or phone number, or display name

e) Fraud and bot detection : Using Captcha to detect bots and also verification for large amounts of transfers

f) Customer Support : Via phone or email and a chat bot for basic queries

g) ~~Ind~~

[illegible] 30/9/24

Class Diagram

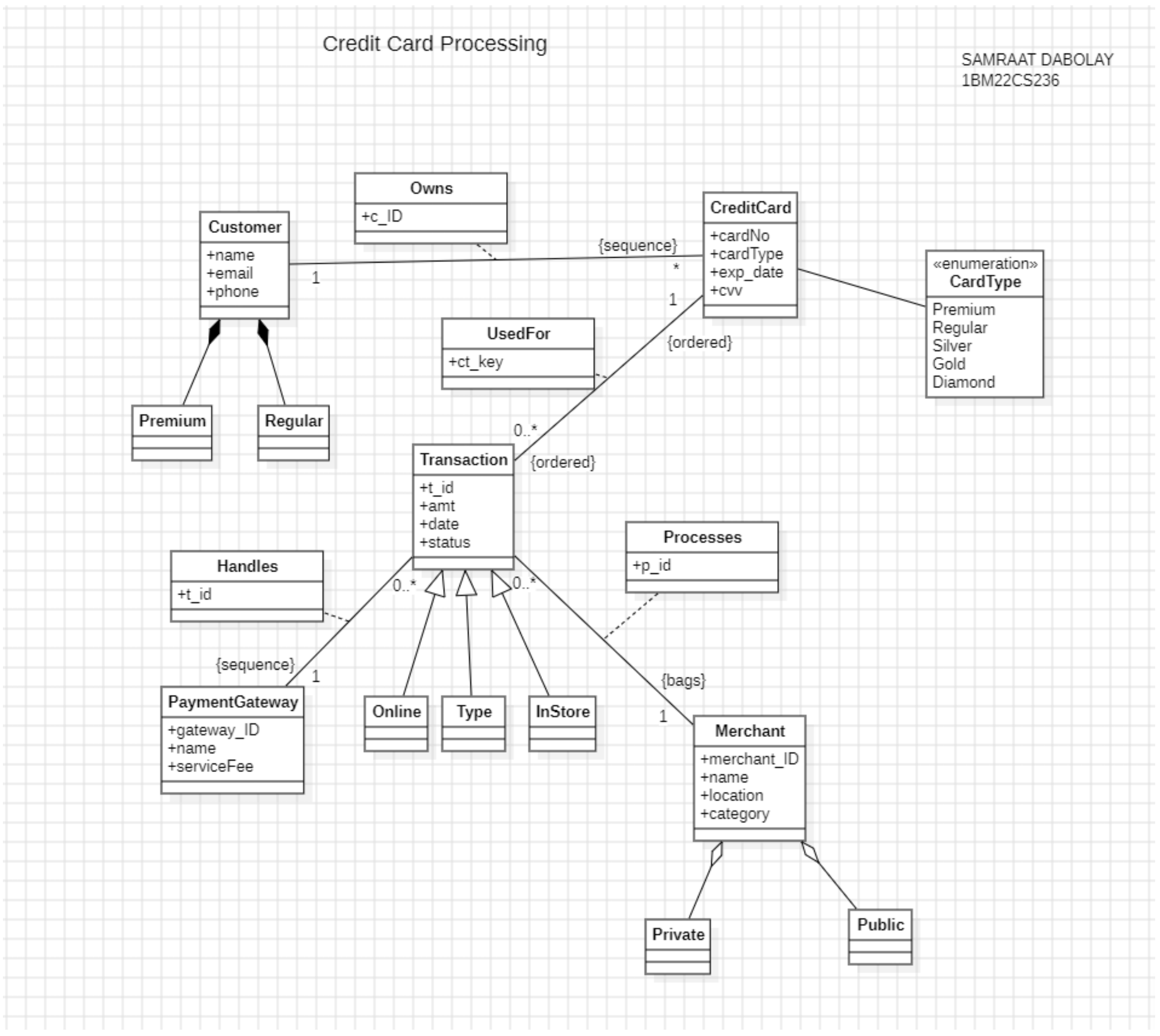


Fig 2.1

This ER Diagram presents a conceptual model for a credit card processing system. It outlines the key entities, their attributes, and the relationships between them. The Customer entity, with attributes like name, email, and phone, is associated with a collection of CreditCard entities. The CreditCard entity, with attributes such as cardNo, cardType, exp_date, and CVV, can be of different types, including Premium, Regular, Silver, Gold, and Diamond, as defined by the CardType enumeration.

The Customer entity is further categorized into Premium and Regular customers, likely representing different levels of service or benefits. The Transaction entity captures details about each transaction, including its ID, amount, date, and status. Transactions can be associated with different PaymentGateways, each with its own gateway_ID, name, and serviceFee. PaymentGateways can handle various transaction types, such as Online and InStore.

The Merchant entity represents businesses that accept credit card payments. It has attributes like merchant_ID, name, location, and category. Merchants can be categorized as either Private or Public, indicating the type of business they operate.

The diagram illustrates the relationships between these entities, such as the "Owns" relationship between Customer and CreditCard, and the "UsedFor" relationship between Customer and Transaction. It also shows the cardinality of these relationships, indicating how many instances of one entity can be associated with another. For example, a Customer can own multiple CreditCards, but a CreditCard can only be owned by one Customer.

State Diagram

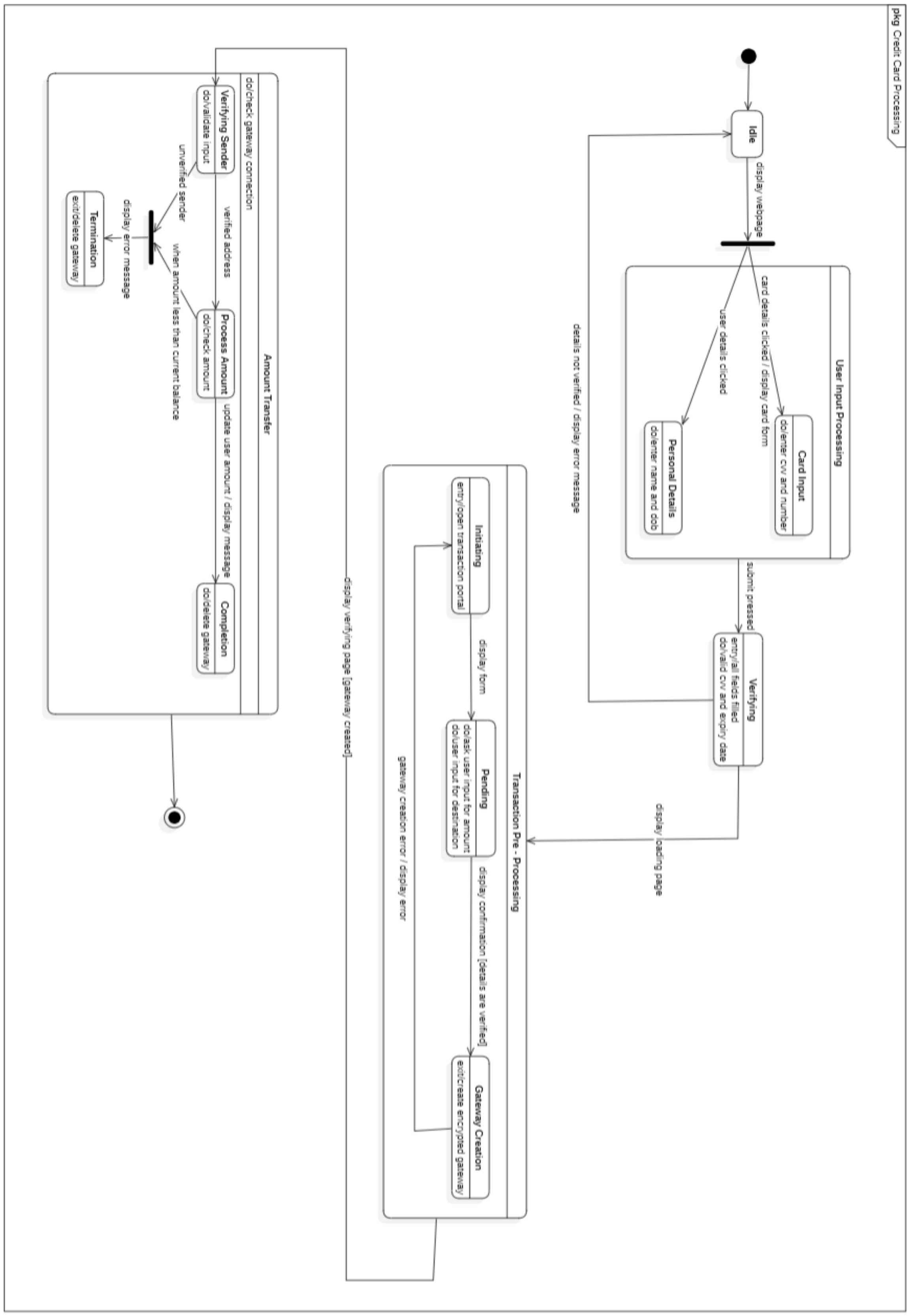

pkg Credit Card Processing
Credit Card Processing
Idle
display webpage
User Input Processing
card details clicked / display card form
user details clicked
Card Input
do/enter cvv and number
Personal Details
do/enter name and dob
details not verified / display error message
submit pressed
Verifying
entry/all fields filled
do/valid cvv and expiry date
display loading page
Transaction Pre - Processing
Initiating
entry/open transaction portal
display form
Pending
do/ask user input for amount
do/user input for destination
display confirmation [details are verified]
Gateway Creation
exit/create encrypted gateway
gateway creation error / display error
display verifying page [gateway created]
Amount Transfer
do/check gateway connection
Verifying Sender
do/validate input
verified address
unverified sender
Process Amount
do/check amount
when amount less than current balance
display error message
Termination
exit/delete gateway
update user amount / display message
Completion
do/delete gateway
SAMRAAT DABOLAY
1BM22CS236

Fig 2.2

This state diagram depicts the workflow involved in a credit card transaction. It starts with the "Idle" state, where the system is waiting for user input. Once the user initiates the process, the system transitions to the "Card Input" state, where the user enters their card details. These details are then verified, and if valid, the system proceeds to the "Personal Details" state where the user enters their personal information.

After entering personal details, the system enters the "Transaction Pre-Processing" state, where it prepares for the transaction by creating an encrypted gateway. If the gateway creation is successful, the system transitions to the "Amount Transfer" state. In this state, the user inputs the amount to be transferred, and the system verifies the sender and checks the available balance. If the verification and balance check are successful, the system processes the amount and updates the user's balance. Finally, the system deletes the gateway and transitions to the "Completion" state, indicating a successful transaction.

If any errors occur during the process, such as invalid card details, insufficient balance, or gateway creation failure, the system transitions to the "Termination" state, and an appropriate error message is displayed to the user.

Use-Case Diagram

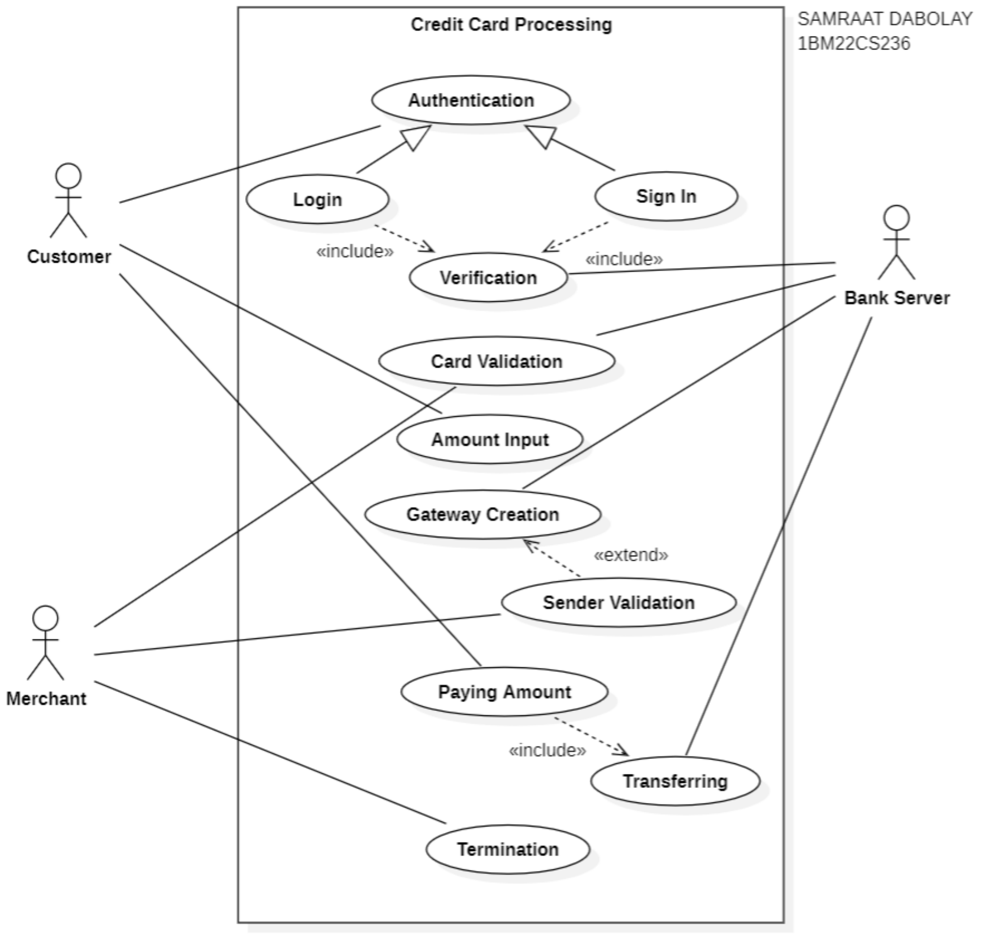


Fig 2.3

This use case diagram illustrates the various interactions involved in a credit card processing system. It depicts three key actors: Customer, Bank Server, and Merchant. The diagram highlights the core use case of "Authentication," which encompasses two sub-use cases: "Login" and "Sign In." These sub-use cases are included within the broader "Authentication" use case, indicating that they are essential steps within the authentication process.

The diagram also shows the "Card Validation" use case, which is likely triggered after successful authentication. Following card validation, the "Amount Input" use case allows the user to enter the transaction amount. The "Gateway Creation" use case facilitates the establishment of a secure payment gateway. This use case is extended by the "Sender Validation" use case, suggesting that sender validation is an optional or additional step that may occur during gateway creation.

Finally, the "Paying Amount" use case, which includes the "Transferring" use case, represents the actual payment process. The "Termination" use case indicates the end of the transaction, which can occur due to successful completion or an error.

Sequence Diagram

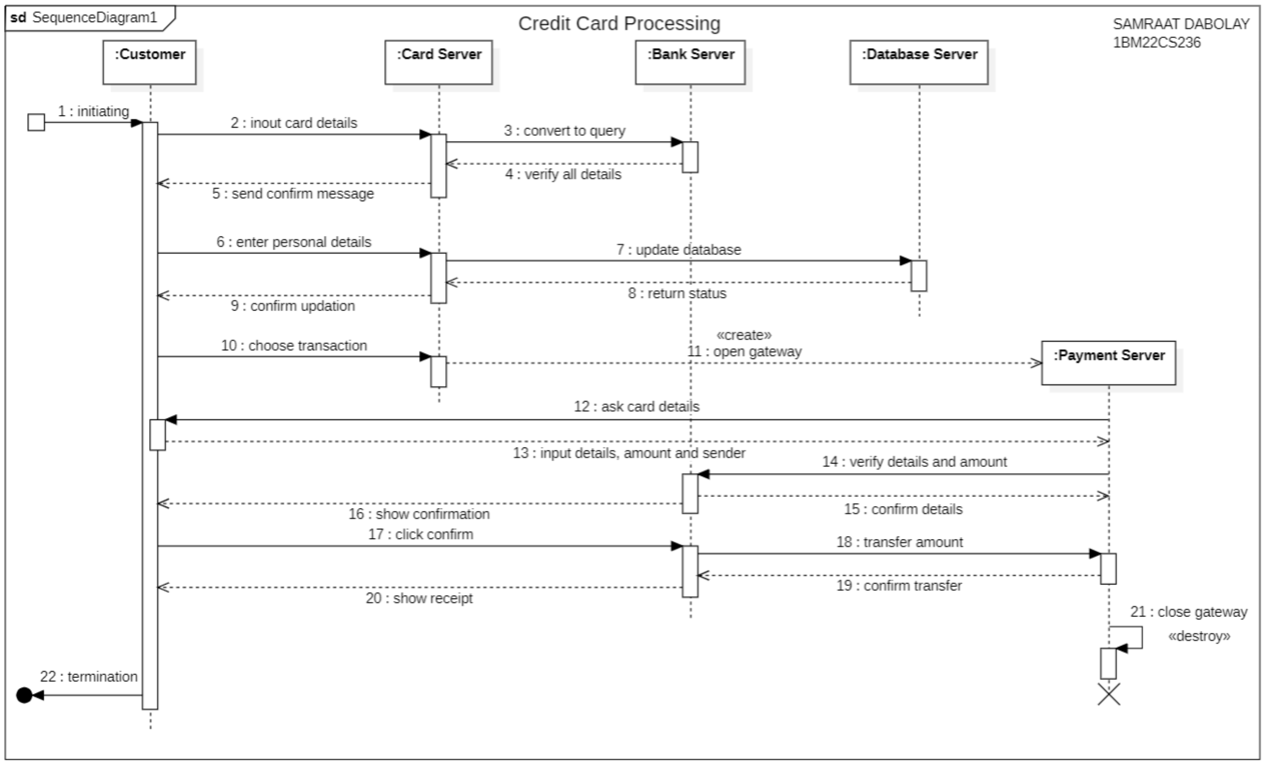


Fig 2.4

This diagram illustrates the sequence of interactions between various components of the system during a credit card transaction. The key actors involved are the Customer, Card Server, Bank Server, Database Server, and Payment Server.

The sequence begins with the Customer initiating the transaction. The Customer then inputs their card details, which are sent to the Card Server for processing. The Card Server converts the card details into a query and sends it to the Database Server for verification. The Database Server checks the card details and returns the verification status to the Card Server. If the card details are valid, the Card Server sends a confirmation message to the Customer.

Next, the Customer enters their personal details, which are then sent to the Database Server for updating the customer information. The Database Server updates the information and returns a confirmation status to the Card Server.

The Customer then chooses the type of transaction they wish to perform. Based on this selection, the Card Server creates a new instance of the Payment Server.

The Payment Server then requests the card details and transaction amount from the Customer. The Customer inputs the required details, and the Payment Server verifies the information. Upon successful verification, the Payment Server processes the transaction and transfers the requested amount.

Once the transfer is complete, the Payment Server sends a confirmation message to the Customer. The Customer then receives a receipt, indicating the successful completion of the transaction. Finally, the Payment Server closes the connection and is destroyed.

Activity Diagram

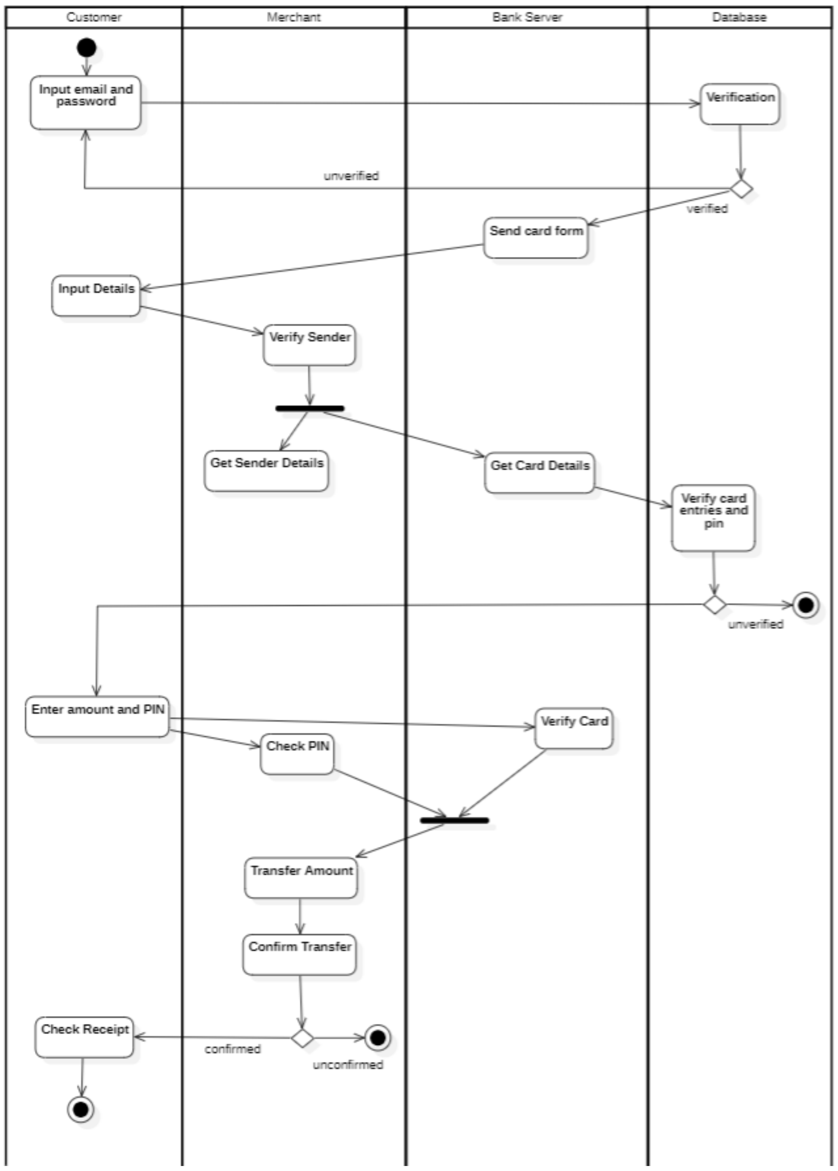


Fig 2.5

This Activity Diagram illustrates the workflow involved in an online money transfer process. The diagram is organized into swimlanes, representing the different actors involved: Customer, Merchant, Bank Server, and Database. The process begins with the Customer entering their email and password to initiate the transaction. The Merchant then verifies the provided credentials. If the verification fails, the process terminates with an "unverified" outcome.

If the verification is successful, the Merchant sends a card form to the Customer to input their card details. The Bank Server retrieves the sender details and the Database verifies the entered card details and PIN. If the verification fails, the process terminates with an "unverified" outcome.

If the card details are verified, the Customer enters the amount to be transferred and the PIN. The Merchant checks the PIN and the Bank Server verifies the card again. If the verification is successful, the Merchant processes the amount transfer and confirms the transfer.

Finally, the Customer checks the receipt, and if the transfer is successful, the process terminates with a "confirmed" outcome. Otherwise, it terminates with an "unconfirmed" outcome.

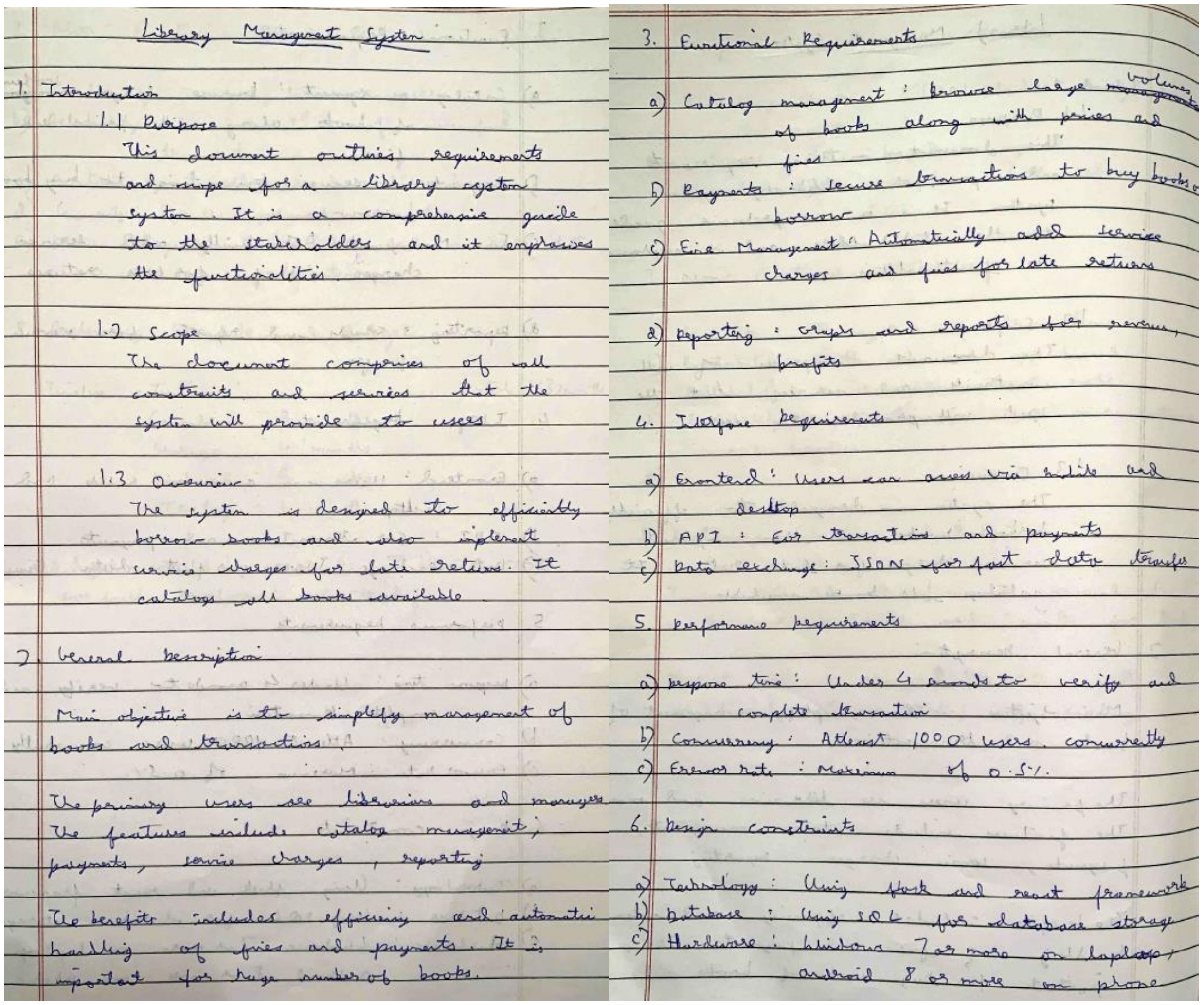
3. Library Management System

Problem Statement

Libraries often face challenges in managing a large number of books, tracking borrowings and returns, and maintaining accurate records manually. These manual processes lead to issues such as misplaced books, delayed returns, difficulty in searching for books, and inefficient management of inventory. Additionally, users and staff may experience frustration due to lack of real-time information and poor tracking of overdue books.

The proposed Library Management System aims to streamline these processes by automating the management of book records, borrowings, returns, and fines. The system will provide an efficient, user-friendly interface for both staff and library members, improving the accuracy of records, speeding up book searches, and ensuring better resource management and overall library operations.

SRS-Software Requirements Specification



Library Management System

1. Introduction

1.1 Purpose
This document outlines requirements and scope for a library system system. It is a comprehensive guide to the stakeholders and it emphasises the functionalities.

1.2 Scope
The document comprises of all constraints and services that the system will provide to users

1.3 Overview
The system is designed to efficiently borrow books and also implement service charges for late returns. It catalogs all books available.

2. General Description

Main objective is to simplify management of books and transactions.

The primary users are librarians and managers. The features include catalog management, payments, service charges, reporting

The benefits includes efficiency and automatic handling of fines and payments. It is important for huge number of books.

3. Functional Requirements

a) Catalog management : Browse large ~~management~~ volumes of books along with prices and fines

b) Payments : secure transactions to buy books or borrow

c) Fine Management : Automatically add service charges and fines for late returns

d) Reporting : Graphs and reports for revenue, profits

4. Interface Requirements

a) Frontend : Users can access via mobile and desktop

b) API : For transactions and payments

c) Data exchange : JSON for fast data transfer

5. Performance Requirements

a) Response time : Under 4 seconds to verify and complete transaction

b) Concurrency : Atleast 1000 users concurrently

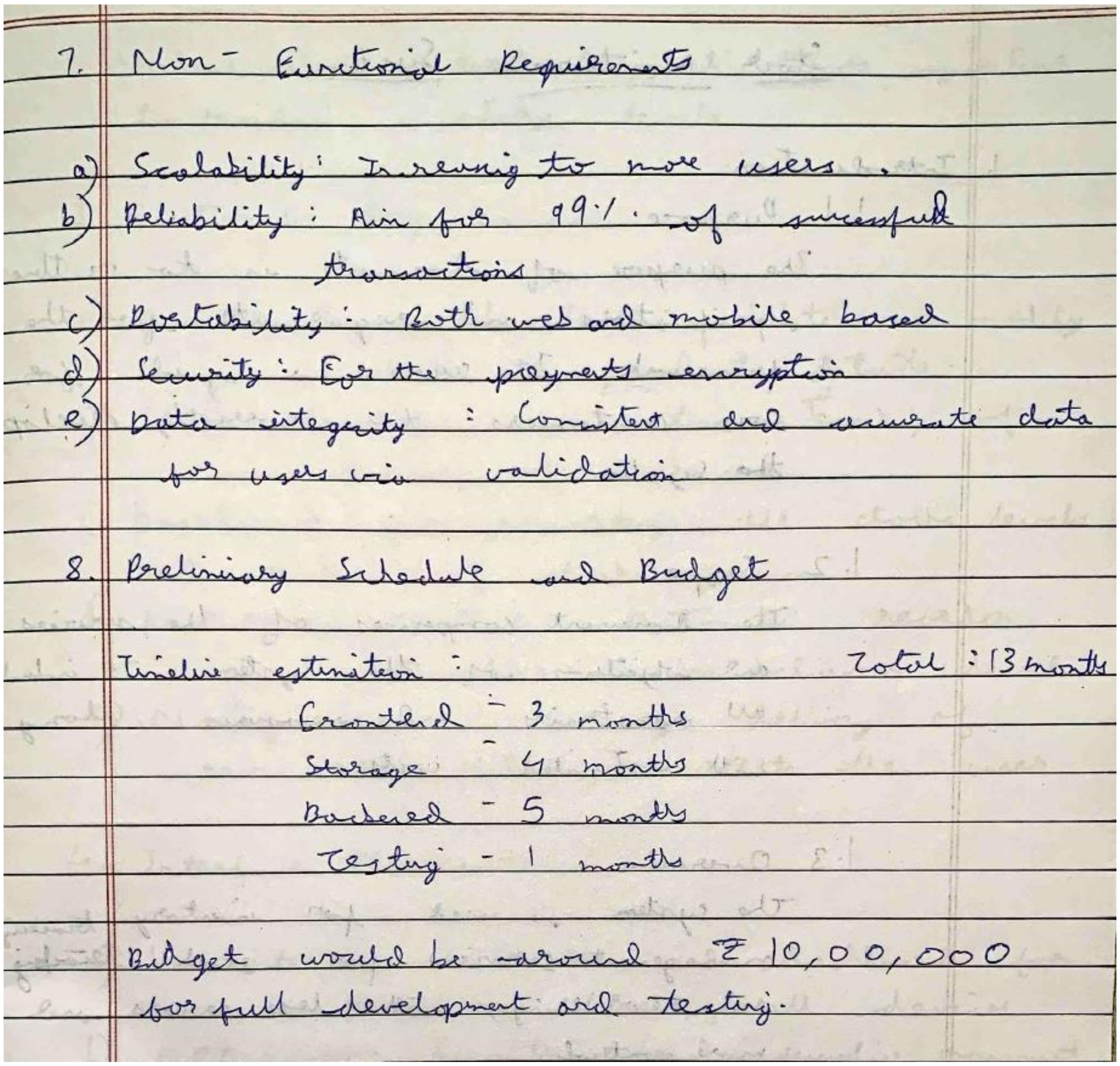
c) Error rate : Maximum of 0.5%.

6. Design constraints

a) Technology : Using flask and react framework

b) Database : Using SQL for database storage

c) Hardware : Windows 7 or more on laptops, android 8 or more on phone



7. Non- Functional Requirements

a) Scalability: Increasing to more users.
b) Reliability: Aim for 99% of successful transactions.
c) Portability: Both web and mobile based
d) Security: For the payments encryption
e) Data integrity: Consistent and accurate data for users via validation

8. Preliminary Schedule and Budget

Timeline estimation: Total: 13 months

Frontend	3 months
Storage	4 months
Backend	5 months
Testing	1 months

Budget would be around ₹ 10,00,000 for full development and testing.

Class Diagram

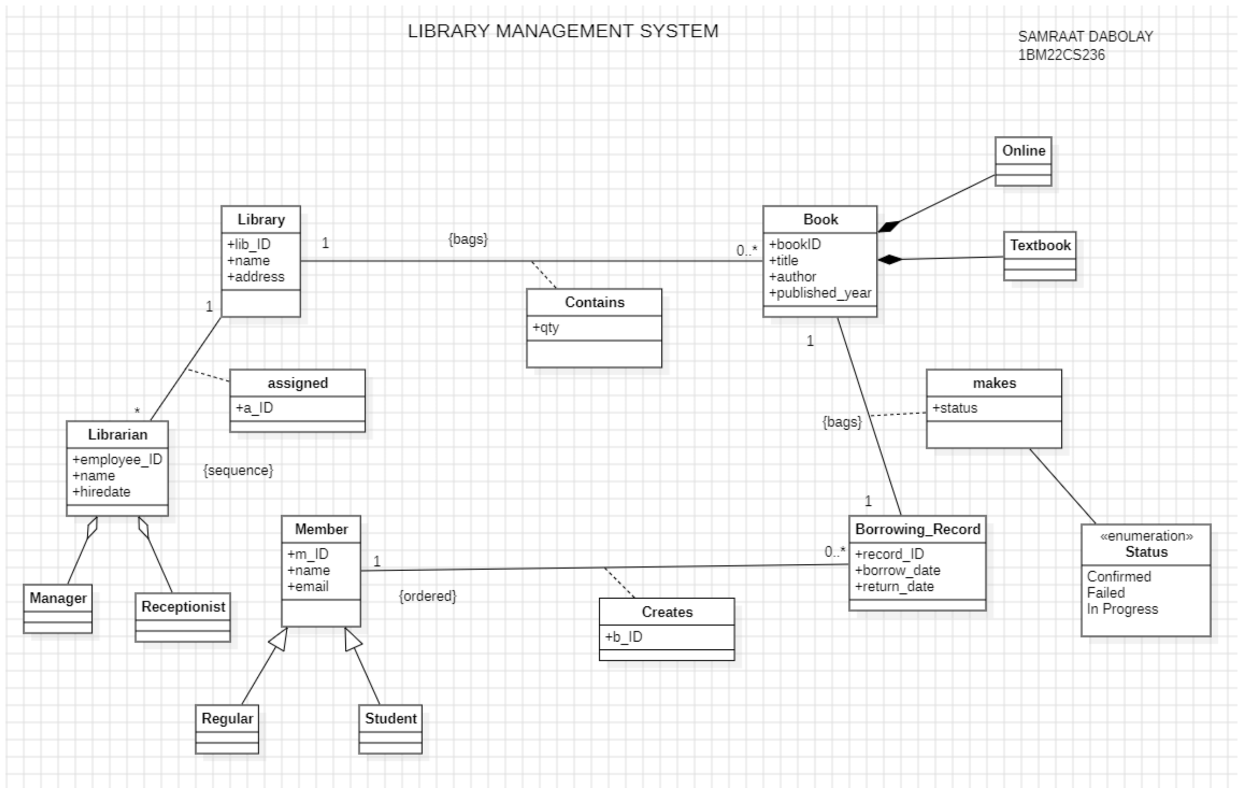


Fig 3.1

This ER Diagram provides a conceptual model for a library management system, outlining the key entities, their attributes, and the relationships between them. Entities such as Library, Book, Librarian, Member, and Borrowing_Record are represented, each with specific attributes describing their characteristics. For example, the Book entity includes attributes like bookID, title, author, and published_year, while the Member entity has attributes like m_ID, name, and email.

The diagram illustrates various relationships between these entities. A Library contains multiple Books, and Librarians are assigned to specific Libraries. Members can borrow multiple Books, and Librarians create Borrowing_Records when a Member borrows a book. The Librarian entity is further specialized into Manager and Receptionist, indicating different roles within the library staff. Similarly, the Member entity is specialized into Regular and Student, representing different types of library members.

The diagram also incorporates advanced features like enumeration and cardinality. The Status attribute in the Borrowing_Record entity is defined as an enumeration, restricting its values to a predefined set (Confirmed, Failed, In Progress). Cardinality notations, such as 1:1 and 1:N, are used to represent the number of instances involved in each relationship.

State Diagram

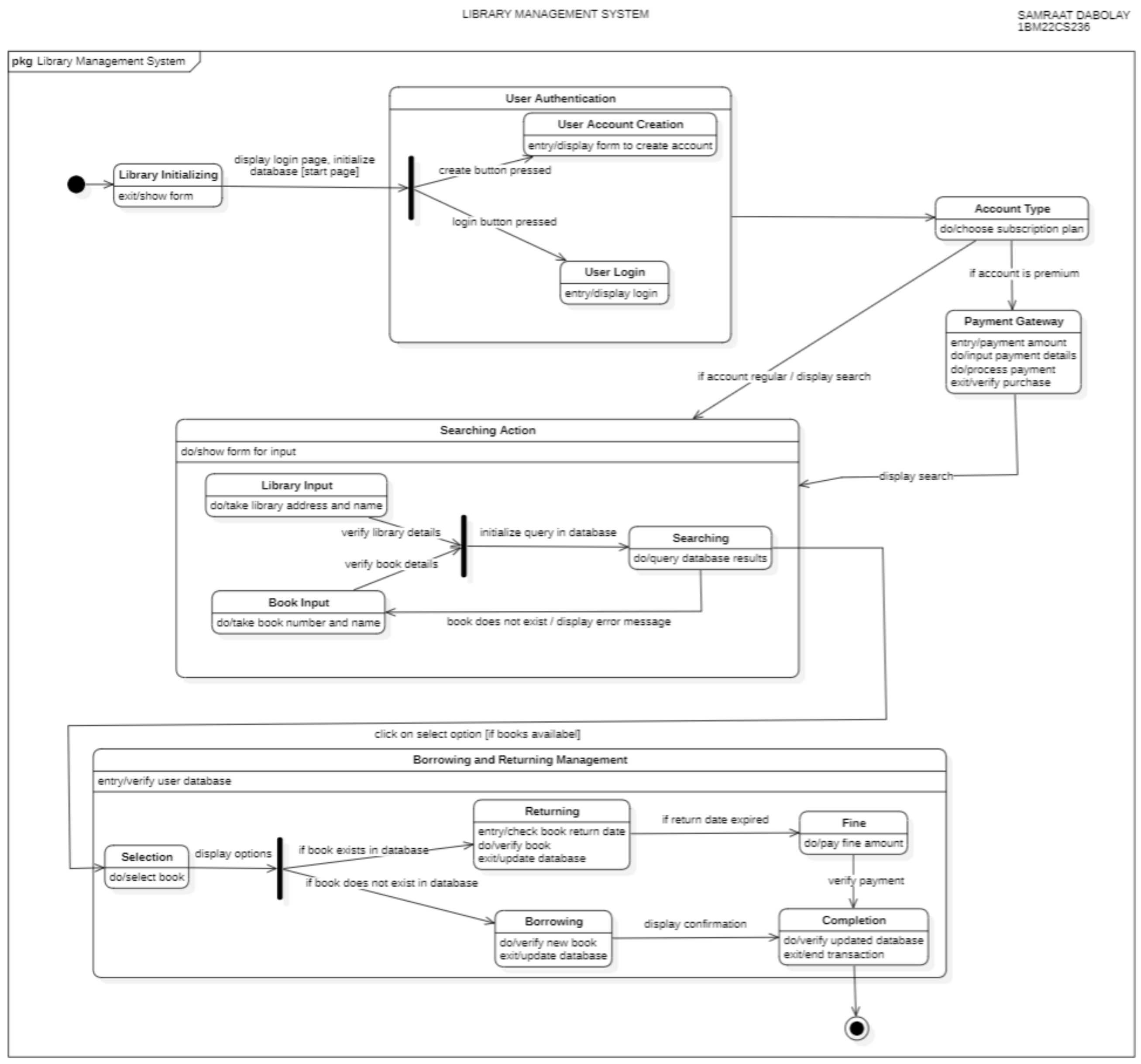


Fig 3.2

This Activity Diagram illustrates the workflow of a library management system. It begins with the "Library Initializing" activity, where the system displays the login page and initializes the database. The user can then engage in "User Authentication." This involves either creating a new user account or logging in. If creating an account, the user chooses a subscription plan. Premium users proceed to the "Payment Gateway" to complete their payment. Regular users directly proceed to the "Searching Action."

In the "Searching Action," the user enters their search criteria, and the system queries the database to retrieve relevant results. If the search criteria match available books, the user can proceed to the "Borrowing and Returning Management" activity.

The "Borrowing and Returning Management" activity involves selecting a book, verifying its availability, and initiating the borrowing process. If the book is available, the system creates a new borrowing record and updates the database. If the return date is expired, the system calculates and displays any fines. The user then pays the fine and completes the transaction.

The diagram demonstrates the use of a fork and join mechanism. When the user creates a new account, the flow splits into two parallel paths: one for premium users to proceed to the "Payment Gateway" and another for regular users to directly proceed to the "Searching Action." These two paths then converge at the "Searching Action" activity, indicating that both premium and regular users can proceed to search for books after completing their respective account creation or payment processes.

Use-Case Diagram

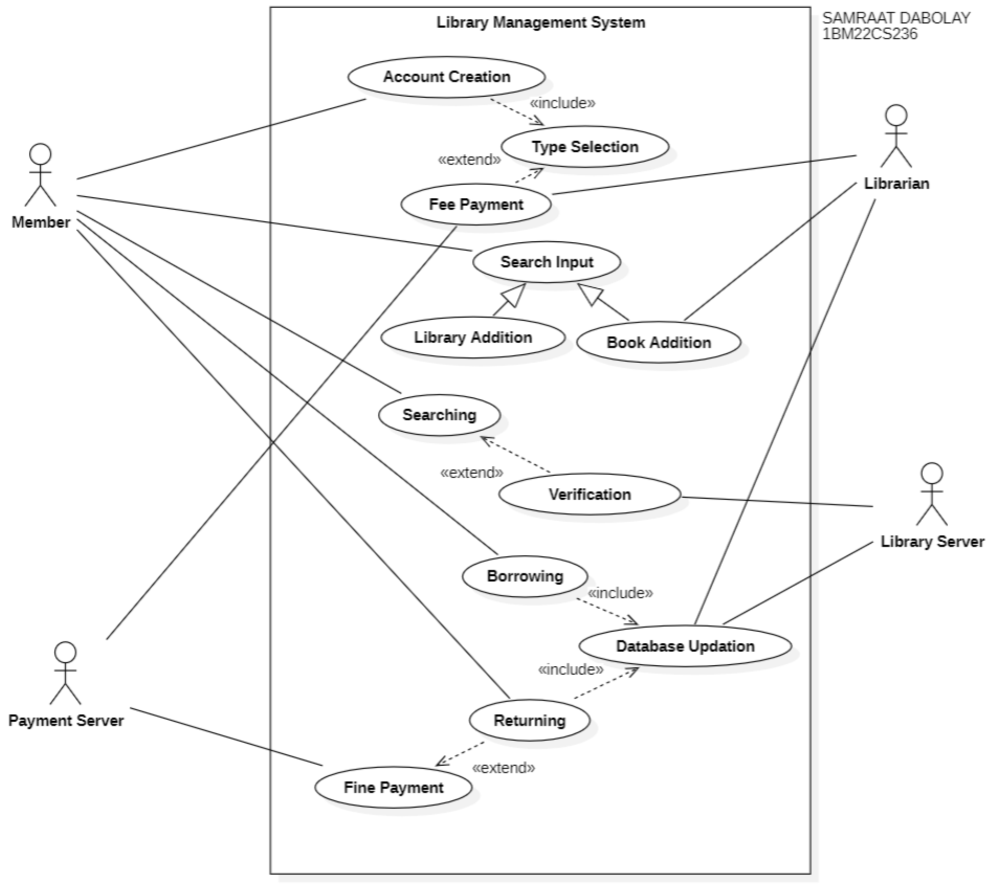


Fig 3.3

This use case diagram illustrates the various functionalities and interactions within a library management system. It depicts three key actors: Member, Librarian, and Library Server. The diagram shows several core use cases. The "Account Creation" use case is extended by "Type Selection" and "Fee Payment," indicating that these are optional or additional steps within the account creation process.

The "Search Input" use case is further specialized into "Library Addition" and "Book Addition," suggesting that the search functionality can be used for adding new libraries or books to the system. The "Searching" use case includes the "Verification" use case, indicating that verification is an essential step within the searching process. The "Borrowing" use case includes "Database Updation," suggesting that updating the database is necessary when a member borrows a book.

Finally, the "Returning" use case is extended by "Fine Payment," indicating that a fine payment may be required when a book is returned late.

Sequence Diagram

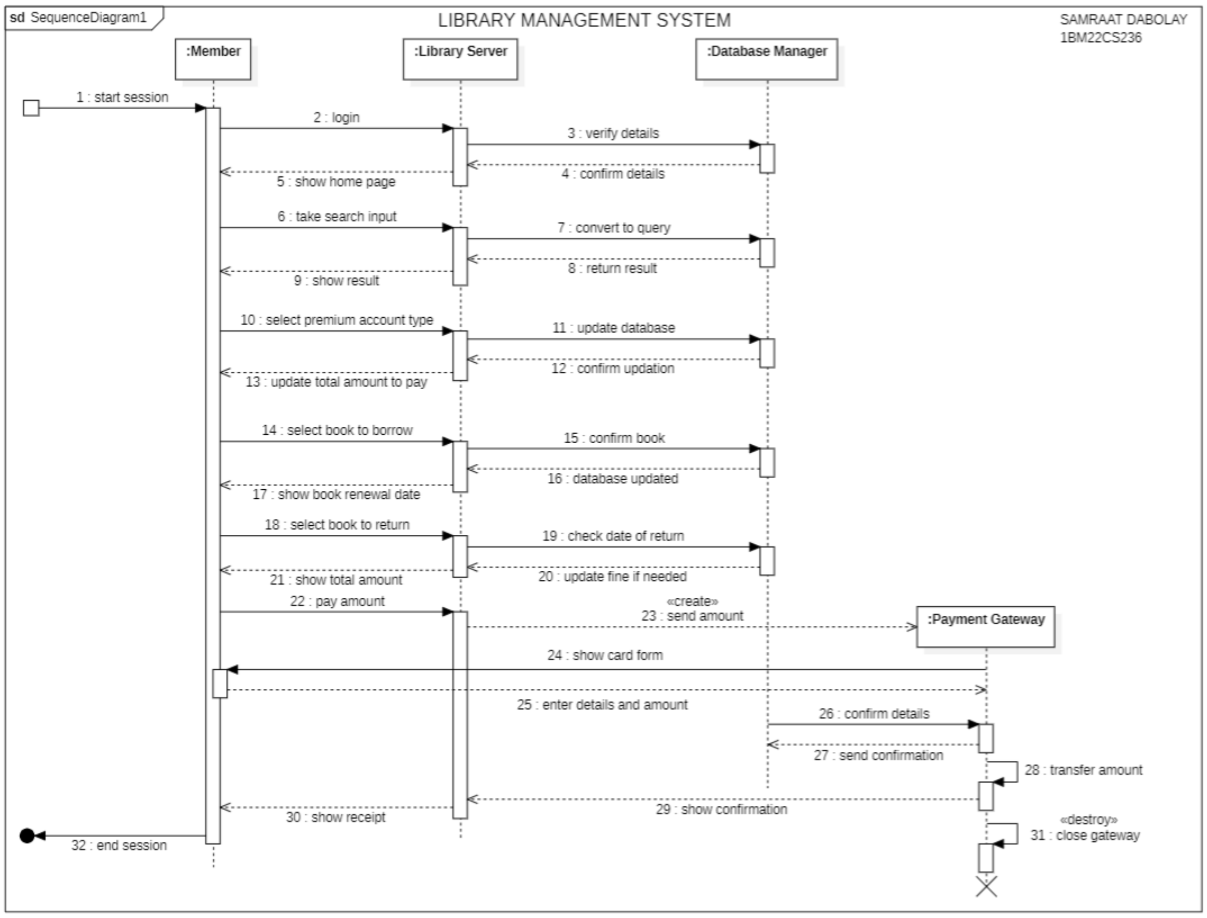


Fig 3.4

This diagram depicts the interactions between various components of the system during a typical library transaction. The sequence starts with the Member initiating a session by logging into the system. The Library Server verifies the Member's details and, if successful, displays the home page.

The Member can then take search input to find available books. The Library Server converts this input into a query and sends it to the Database Manager, which retrieves the relevant results. The Library Server then displays the search results to the Member. If the Member selects a premium account type, the Library Server calculates the total amount to be paid and updates the database accordingly. The Member then selects the book they wish to borrow, and the Library Server confirms the booking and updates the database.

The Library Server then shows the book's renewal date to the Member. If the Member wishes to return a book, they select the book, and the Library Server checks the date of return. If the return date is overdue, the Library Server calculates and displays any fines. The Member then pays the fine amount.

For payment processing, a new instance of the Payment Gateway is created (transient lifeline). The Payment Server displays a card form, and the Member enters their details and the amount. The Payment Server confirms the details and sends a confirmation message. The Payment Server then transfers the amount and displays a confirmation message to the Member. Finally, the Payment Server closes the gateway connection and the Member ends the session.

Activity Diagram

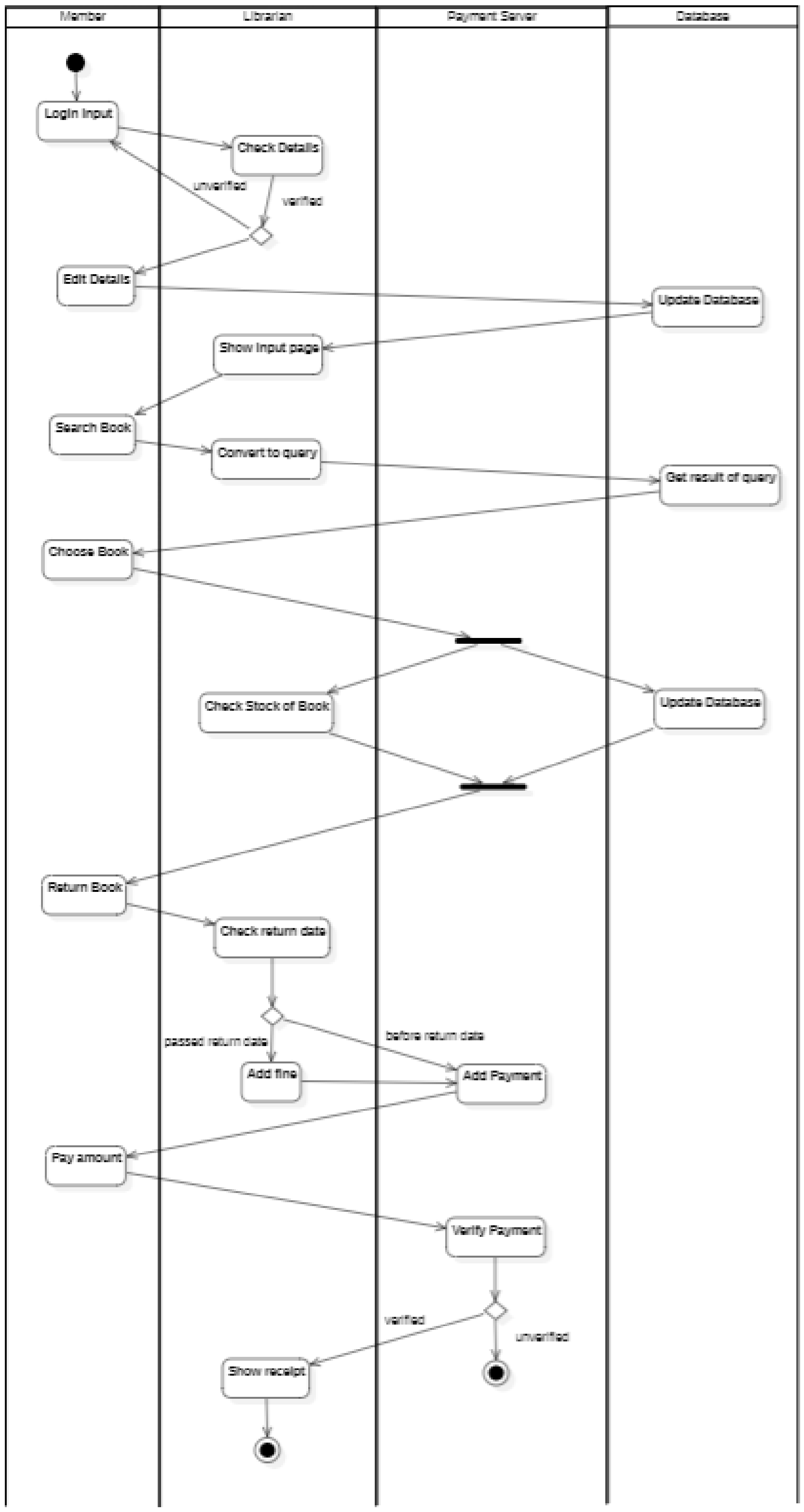


Fig 3.5

This Activity Diagram illustrates the workflow of a library management system, organized into swimlanes to represent the different actors involved: Member, Librarian, Payment Server, and Database.

The process begins within the Member swimlane, where the Member initiates a login. The Librarian, in their respective swimlane, checks the login details. If successful, the Member can proceed to edit their details within their swimlane.

Next, the Member searches for books within their swimlane. The Librarian, in their swimlane, converts the search input into a query and sends it to the Database. The Database processes the query and returns the results, which are then displayed to the Member.

The Member then chooses a book to borrow. The Librarian, in their swimlane, checks the stock of the chosen book and updates the database accordingly. When the Member returns the book, the Librarian checks the return date. If the book is returned late, a fine is calculated and added to the member's account.

The Member then pays the fine amount, and the Payment Server, within its swimlane, verifies the payment. If the payment is successful, the Member is issued a receipt, and the process is complete.

This Activity Diagram effectively visualizes the sequence of activities and decision points within the library management system, highlighting the interactions between the different actors and the flow of information between them. The use of swimlanes enhances the clarity of the diagram by clearly demarcating the responsibilities and actions of each actor involved in the process.

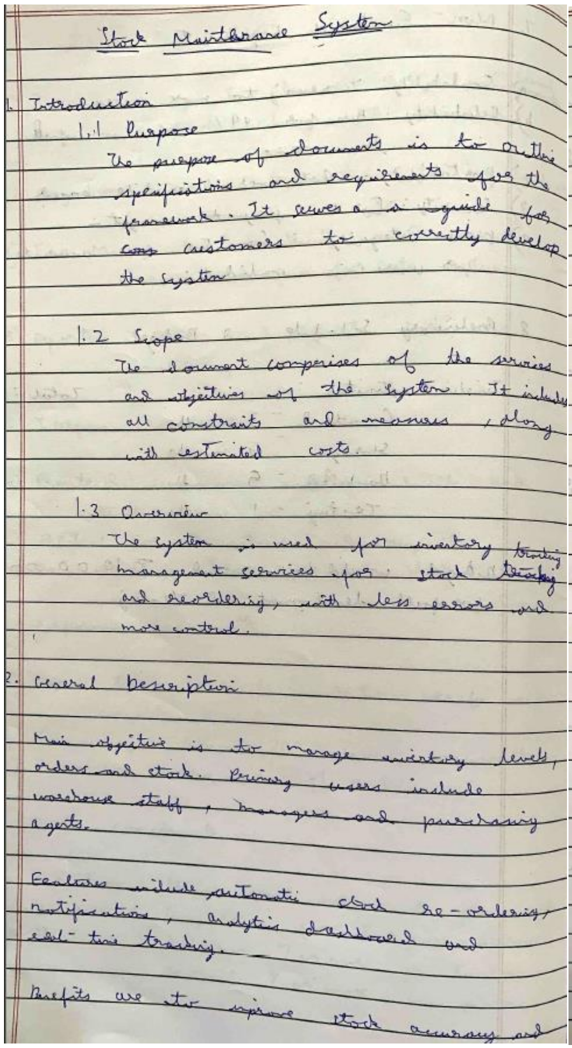
4. Stock Maintenance System

Problem Statement

Businesses often struggle with managing their inventory effectively, leading to issues such as stockouts, overstocking, inaccurate inventory records, and delays in order fulfilment. Manual tracking or outdated systems can cause inefficiencies, errors in stock levels, and a lack of real-time visibility into inventory, resulting in poor decision-making and potential loss of sales or wasted resources.

The proposed Stock Maintenance System aims to solve these problems by automating inventory tracking, providing real-time updates, and ensuring accurate stock levels. The system will streamline stock management, reduce errors, improve decision-making, and enhance overall operational efficiency by offering features like automatic reordering, stock tracking, and detailed reporting.

SRS-Software Requirements Specification



Stock Maintenance System

1. Introduction

1.1 Purpose

The purpose of documents is to outline specifications and requirements for the framework. It serves a a guide for ~~cons~~ customers to correctly develop the system

1.2 Scope

The document comprises of the services and objectives of the system. It includes all constraints and measures, along with estimated costs.

1.3 Overview

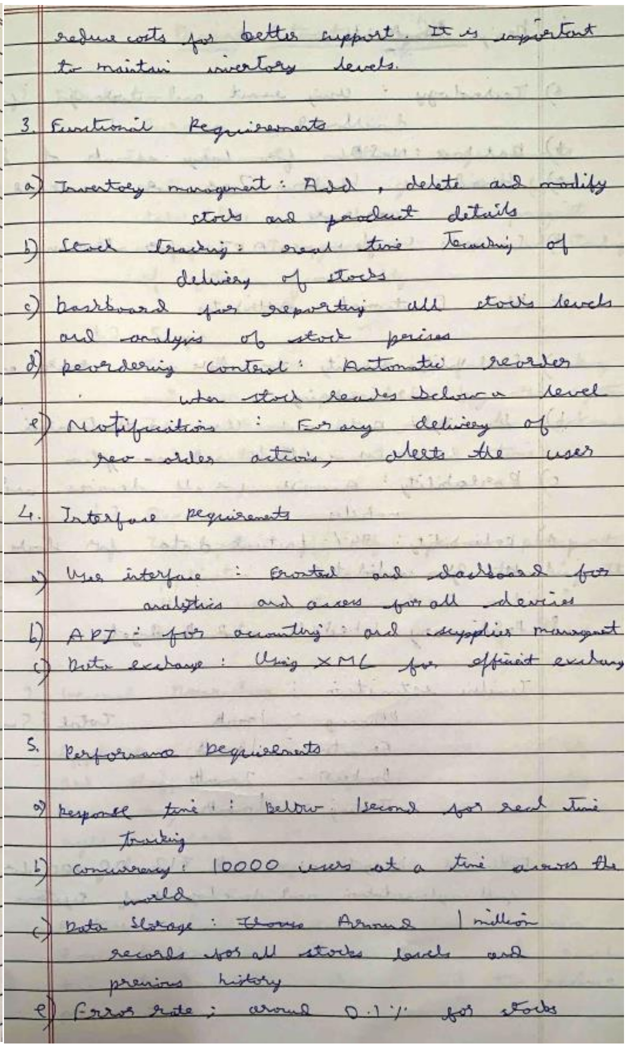
The system is used for inventory tracking management services for stock tracking and reordering, with less errors and more control.

2. General Description

Main objective is to manage inventory levels, orders and stock. Primary users include warehouse staff, managers and purchasing agents.

Features include automatic stock re-ordering, notifications, analytics dashboard and real-time tracking.

Benefits are to improve stock accuracy and



reduce costs for better support. It is important to maintain inventory levels.

3. Functional Requirements

a) Inventory management: Add, delete and modify stocks and product details

b) Stock tracking: real time tracking of delivery of stocks

c) Dashboard for reporting all stocks levels and analysis of stock prices

d) Reordering control: Automatic reorder when stock reaches below a level

e) Notifications: For any delivery of reo-order actions, alerts the users

4. Interface Requirements

a) User interface: Frontend and dashboard for analytics and access for all devices

b) API: for accounting and supplier management

c) Data exchange: Using XML for efficient exchange

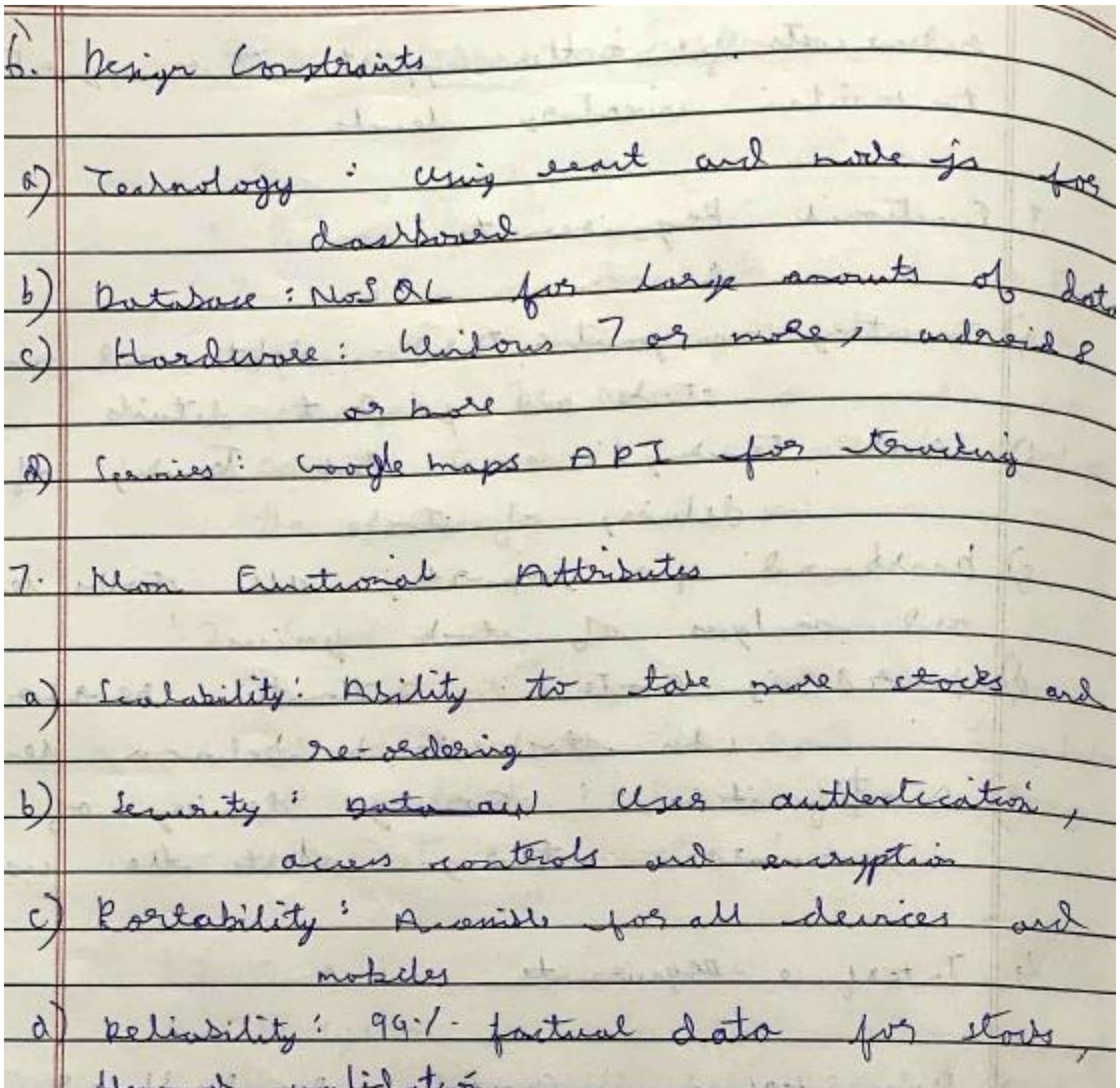
5. Performance Requirements

a) Response time: Below 1second for real time tracking

b) Concurrency: 10000 users at a time across the world

c) Data Storage: ~~Thous~~ Around 1 million records for all stocks levels and previous history

e) Error rate; around 0.1% for stocks



6. Design Constraints

a) Technology : Using react and node js for dashboard
b) Database : NoSQL for large amounts of dat
c) Hardware : Windows 7 or more, android 8 or more
d) Services : Google maps API for tracking

7. Non Functional Attributes

a) Scalability : Ability to take more stocks and re-ordering
b) Security : Data and User authentication, access controls and encryption
c) Portability : Accessible for all devices and mobiles
d) Reliability : 99% factual data for stocks, through validation

8. Preliminary Schedule and Budget

Timeline estimation :

Planning - 1 month	Total : 5 month
Frontend - 1 month	
Backend - 2 months	
Testing - 1 month	

Estimated budget is around ₹12,00,000 for full implementation and development of system

Class Diagram

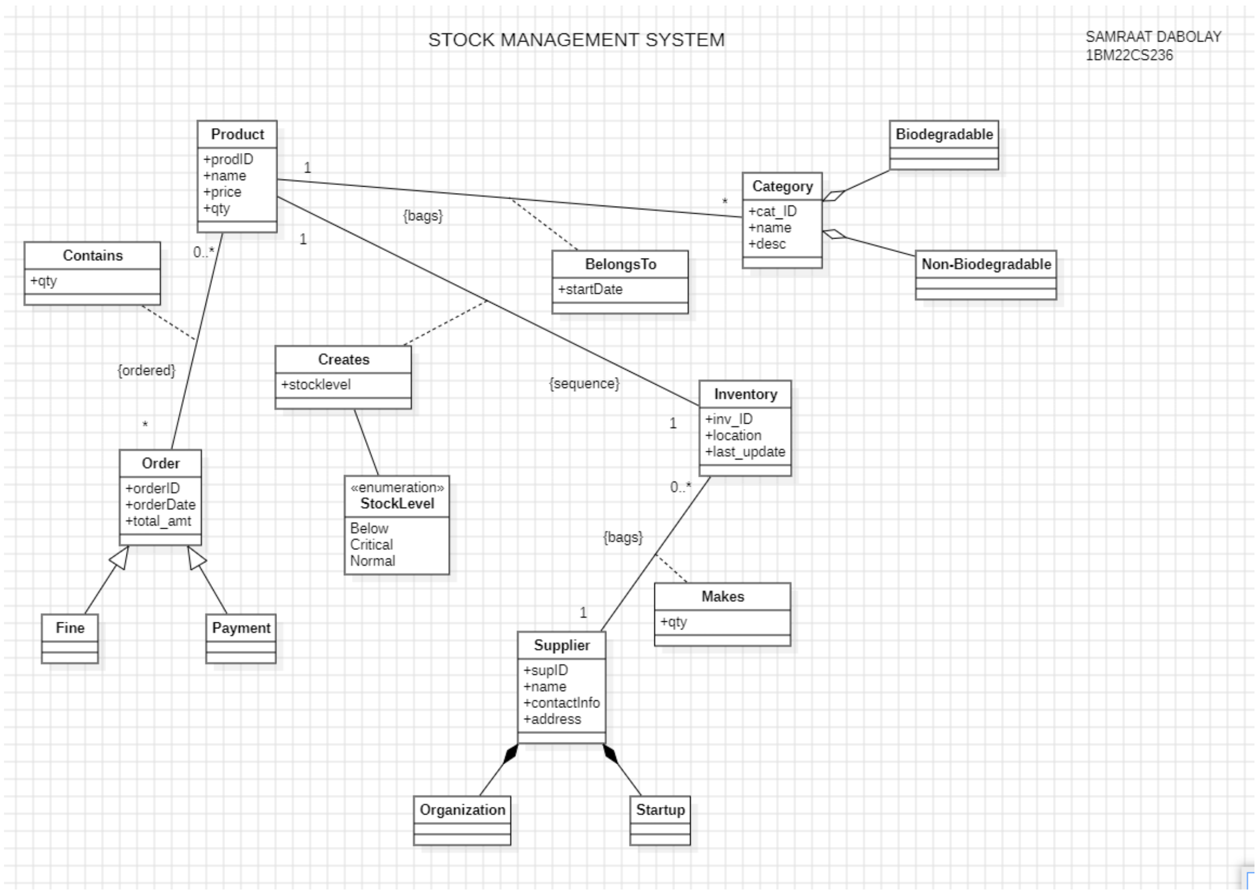


Fig 4.1

This ER Diagram provides a conceptual model for a stock management system, outlining the key entities and their relationships. Entities like Product, Category, Inventory, Supplier, and Order are represented, each with attributes that define their characteristics. For example, the Product entity has attributes like prodID, name, and price, while the Inventory entity includes attributes like inv_ID, location, and last_update.

The diagram illustrates various relationships between these entities. A Product can have multiple inventories stored in different locations, represented by the "Contains" relationship. An Inventory belongs to a specific Category, and this relationship is further categorized into Biodegradable and Non-Biodegradable. Suppliers are responsible for making specific products, and Orders result in the creation of new inventory.

The diagram utilizes cardinality notations (e.g., 1:1, 1:N, 0:N) to represent the number of instances involved in each relationship. It also employs an enumeration for the StockLevel attribute in the Inventory entity, defining possible values as Below, Critical, and Normal.

State Diagram

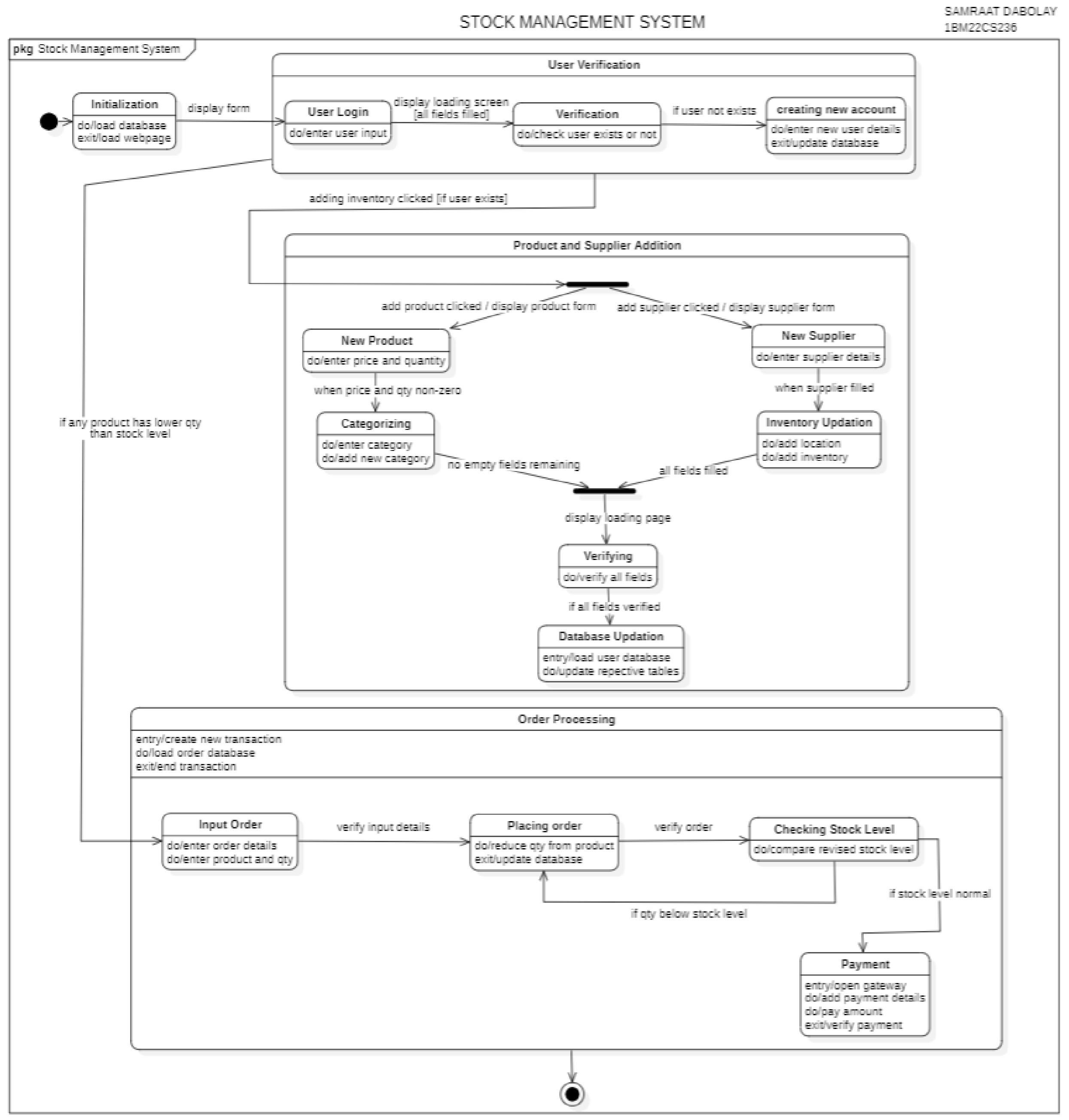


Fig 4.2

This diagram depicts the workflow of a stock management system. It begins with the "Initialization" state, where the database is loaded and the system displays a login form.

The "User Verification" state is a composite state, encompassing the activities of user login and new account creation. If the user exists, the system proceeds to the "Product and Supplier Addition" state.

The "Product and Supplier Addition" state is another composite state, allowing for the addition of new products and suppliers. The addition of a new product triggers the "Categorizing" activity to determine the product's category. If any product has a low stock level, the system enters the "Inventory Updation" state to restock the inventory.

The "Order Processing" state is also a composite state, encompassing the activities of inputting an order, placing the order, checking the stock level, and processing the payment. The "Checking Stock Level" activity uses a fork and join mechanism, splitting the flow to check the stock level for each product in the order and then rejoining to proceed with payment if stock is sufficient.

Overall, this Activity Diagram provides a comprehensive view of the stock management system's workflow, utilizing composite states and fork/join constructs to model complex processes and parallel activities. It effectively visualizes the system's behavior and the interactions between different components.

Use-Case Diagram

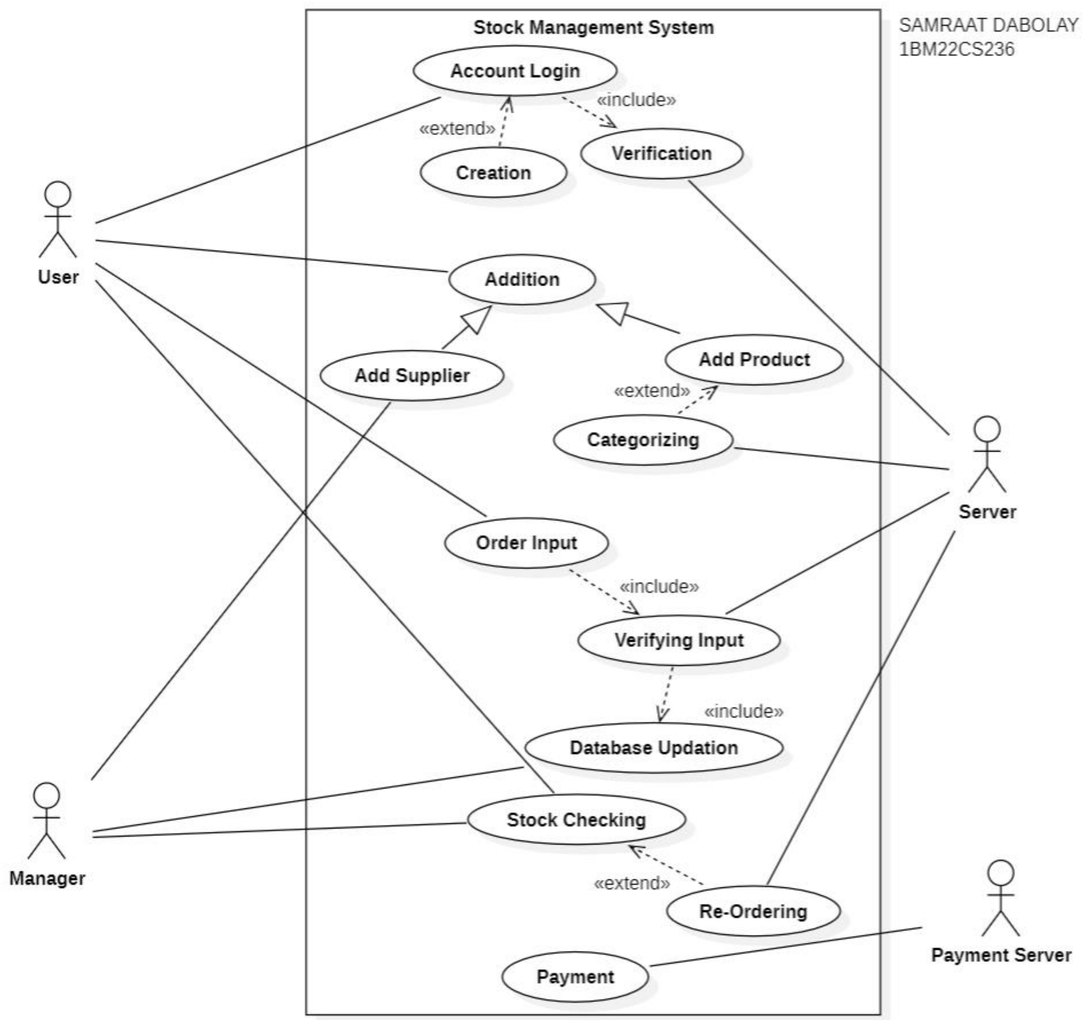


Fig 4.3

This diagram depicts the various functionalities and interactions within a stock management system. It illustrates three key actors: User, Server, and Manager.

The diagram shows several core use cases. The "Account Login" use case includes the "Verification" use case, indicating that verification is an essential step within the login process. The "Creation" use case extends the "Account Login" use case, suggesting that account creation is an optional or additional step after login.

The "Addition" use case is further specialized into "Add Supplier" and "Add Product," indicating that the addition functionality can be used for adding new suppliers or products to the system. The "Addition" use case also includes the "Categorizing" use case, suggesting that categorizing is an essential step within the addition process.

The "Order Input" use case includes the "Verifying Input" use case, indicating that input verification is an essential step before processing the order. The "Order Input" use case also includes the "Database Updation" use case, suggesting that updating the database is necessary when processing an order.

The "Stock Checking" use case is extended by the "Re-Ordering" use case, indicating that re-ordering is an optional or additional step after checking the stock levels.

Finally, the "Payment" use case is associated with the Payment Server actor, suggesting that payment processing is handled by an external payment server.

Sequence Diagram

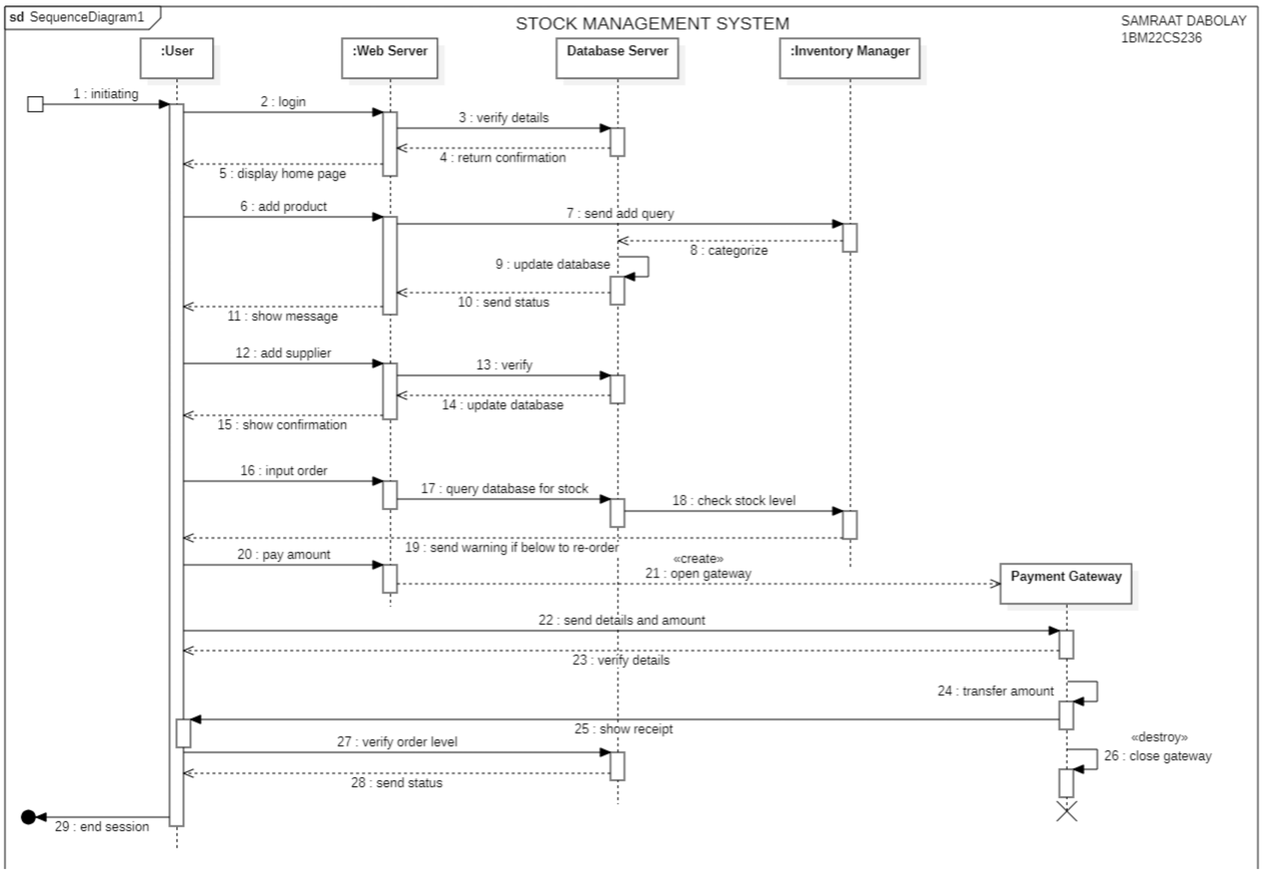


Fig 4.4

This diagram depicts the interactions between various components of the system during different operations like user login, product addition, and order processing.

The sequence starts with the User initiating the session. The Web Server verifies the user's login details and, upon successful authentication, displays the home page.

The User can then add a new product. The Web Server sends an "add query" to the Database Server, which then updates the database and sends a "categorize" message to the Inventory Manager. The Inventory Manager categorizes the product and sends the status back to the Database Server. The Web Server then displays a message to the User confirming the product addition.

Similarly, the User can add a new supplier. The Web Server sends a verification request to the Database Server, which updates the database and sends a confirmation back to the Web Server. The Web Server then displays a confirmation message to the User.

The User can also input an order. The Web Server queries the Database Server to check the stock levels. If the stock level is below a certain threshold, the Web Server sends a warning message to the Inventory Manager to re-order.

If the order can be fulfilled, the Web Server creates a new instance of the Payment Gateway. The User pays the amount, and the Payment Gateway verifies the details and transfers the amount. The Payment Gateway then closes the connection and is destroyed.

Finally, the Web Server verifies the order level and sends a status message to the User. The User then ends the session.

Activity Diagram

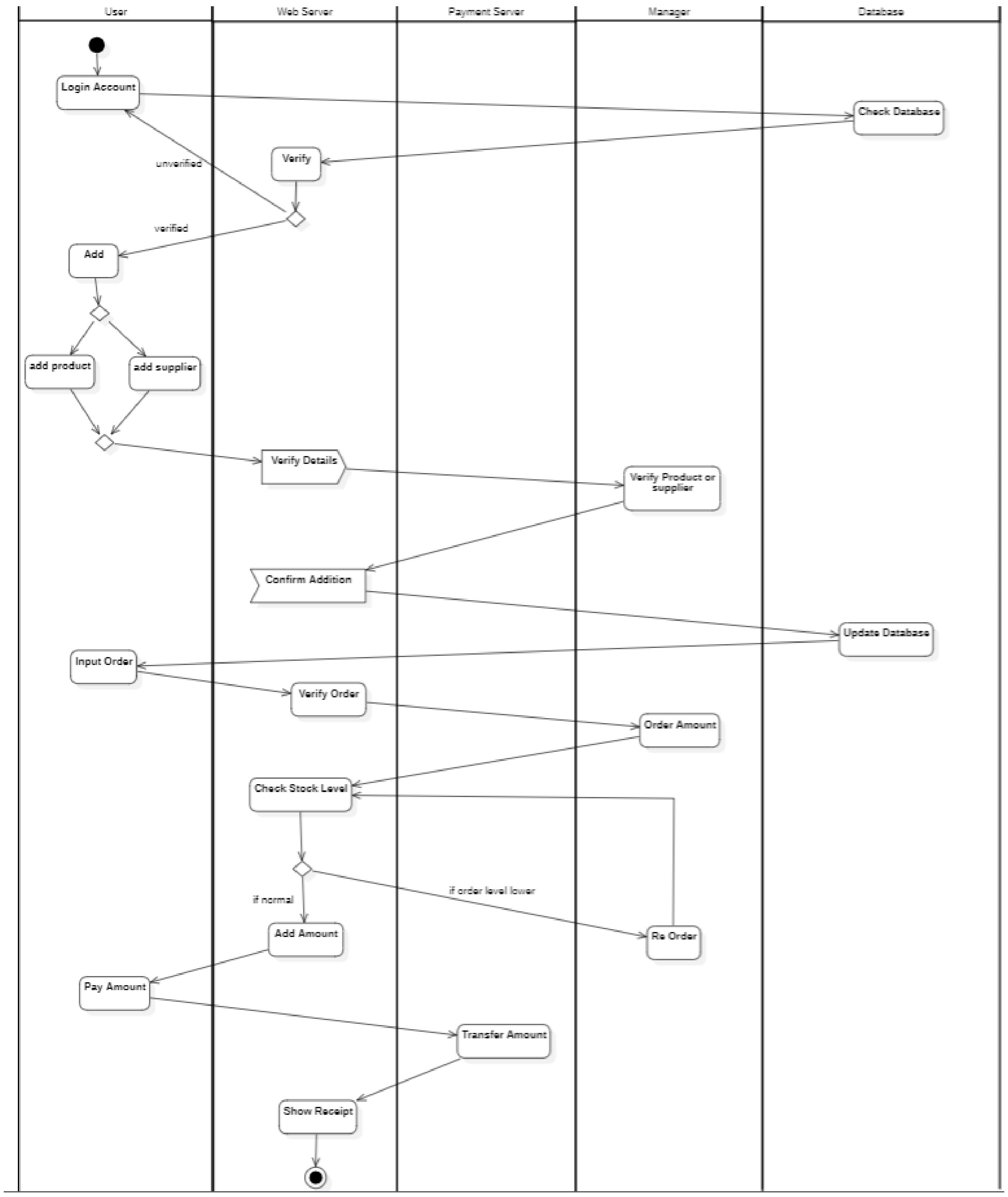


Fig 4.5

This Activity Diagram illustrates the workflow of a stock management system, organized into swimlanes to represent the different actors involved: User, Web Server, Payment Server, Manager, and Database.

The process begins within the User swimlane with the "Login Account" activity. The Web Server, in its respective swimlane, verifies the login credentials. If successful, the User proceeds to the "Add" activity.

Within the "Add" activity, the User can choose to "add product" or "add supplier." The Web Server, in its swimlane, verifies the details provided by the User. If the verification is successful, the "Confirm Addition" activity occurs.

Following this, the User can "Input Order." The Web Server verifies the order and checks the stock level. If the stock level is normal, the process continues. If the order level is lower, the Manager, in their swimlane, is notified to "Re-Order."

The User then proceeds to "Pay Amount." The Payment Server, within its swimlane, processes the payment and transfers the amount. Upon successful payment, the User receives a "Show Receipt."

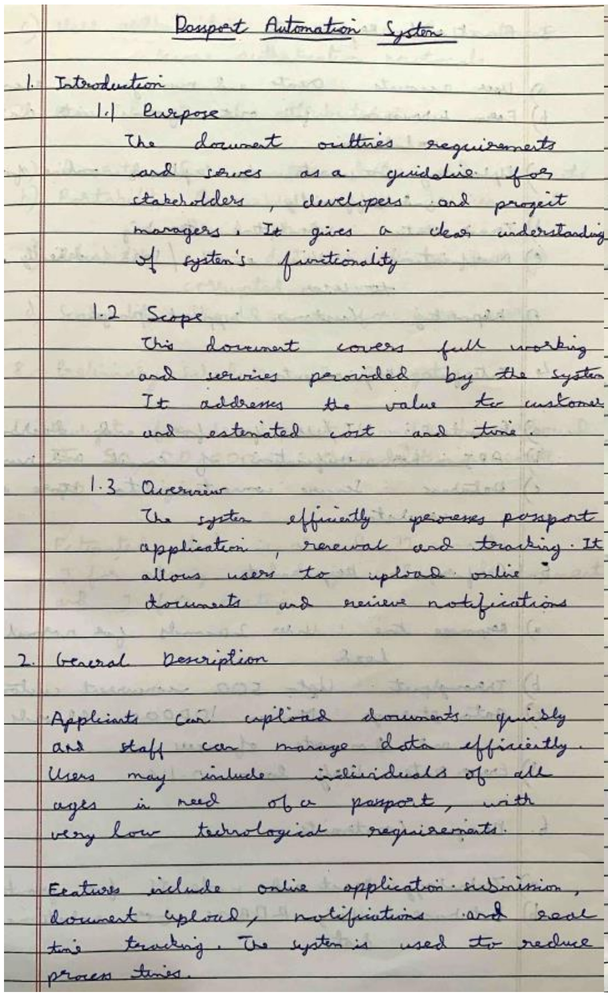
5. Passport Automation System

Problem Statement

The traditional passport application and processing system is often slow, manual, and prone to errors. Applicants face long waiting times, confusion over required documents, and delays in processing, while authorities struggle with managing large volumes of applications, ensuring security, and maintaining accurate records. These inefficiencies create a poor user experience and increase the likelihood of mistakes in data entry or document handling.

The proposed Passport Automation System aims to automate and streamline the entire passport application and processing workflow. The system will enable applicants to submit applications online, track their status in real-time, and receive automatic updates. For authorities, it will provide efficient document verification, improved data accuracy, and faster processing, ultimately enhancing the overall experience for both applicants and officials.

SRS-Software Requirements Specification



Passport Automation System

1. Introduction

1.1 Purpose

The document outlines requirements and serves as a guideline for stakeholders, developers and project managers. It gives a clear understanding of system's functionality

1.2 Scope

This document covers full working and services provided by the system It addresses the value to customer and estimated cost and time.

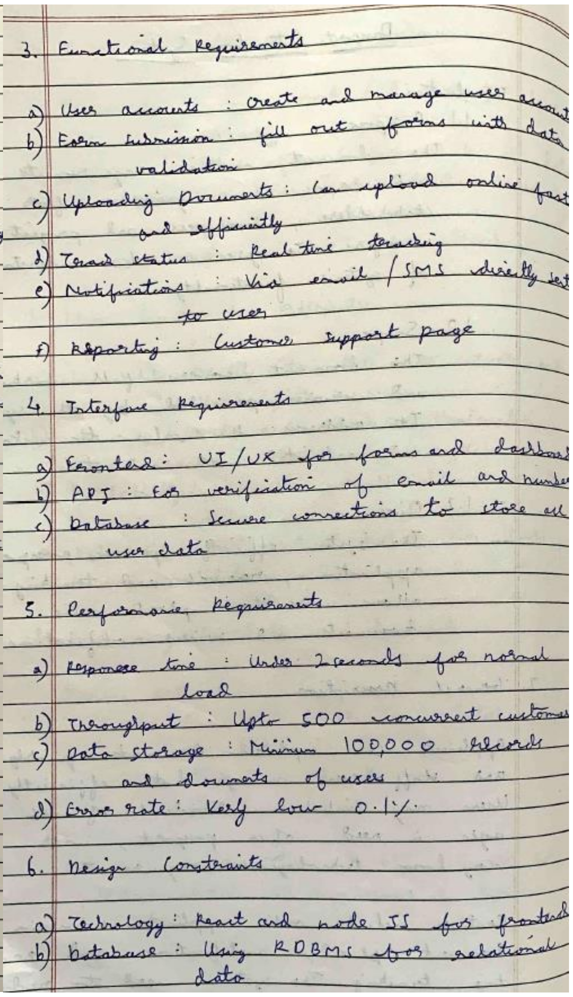
1.3 Overview

The system efficiently processes passport application, renewal and tracking. It allows users to upload online documents and recieve notifications

2. General Description

Applicants can upload documents quickly and staff can manage data efficiently. Users may include individuals of all ages in need of a passport, with very low technological requirements.

Features include online application submission, document upload, notifications and real time tracking. The system is used to reduce process times.



3. Functional Requirements

a) User accounts : Create and manage user account
b) Form submission : fill out forms with data validation
c) Uploading Documents : Can upload online fast and efficiently
d) Track status : Real time tracking
e) Notifications : Via email / SMS directly sent to user
f) Reporting : Customer support page

4. Interface Requirements

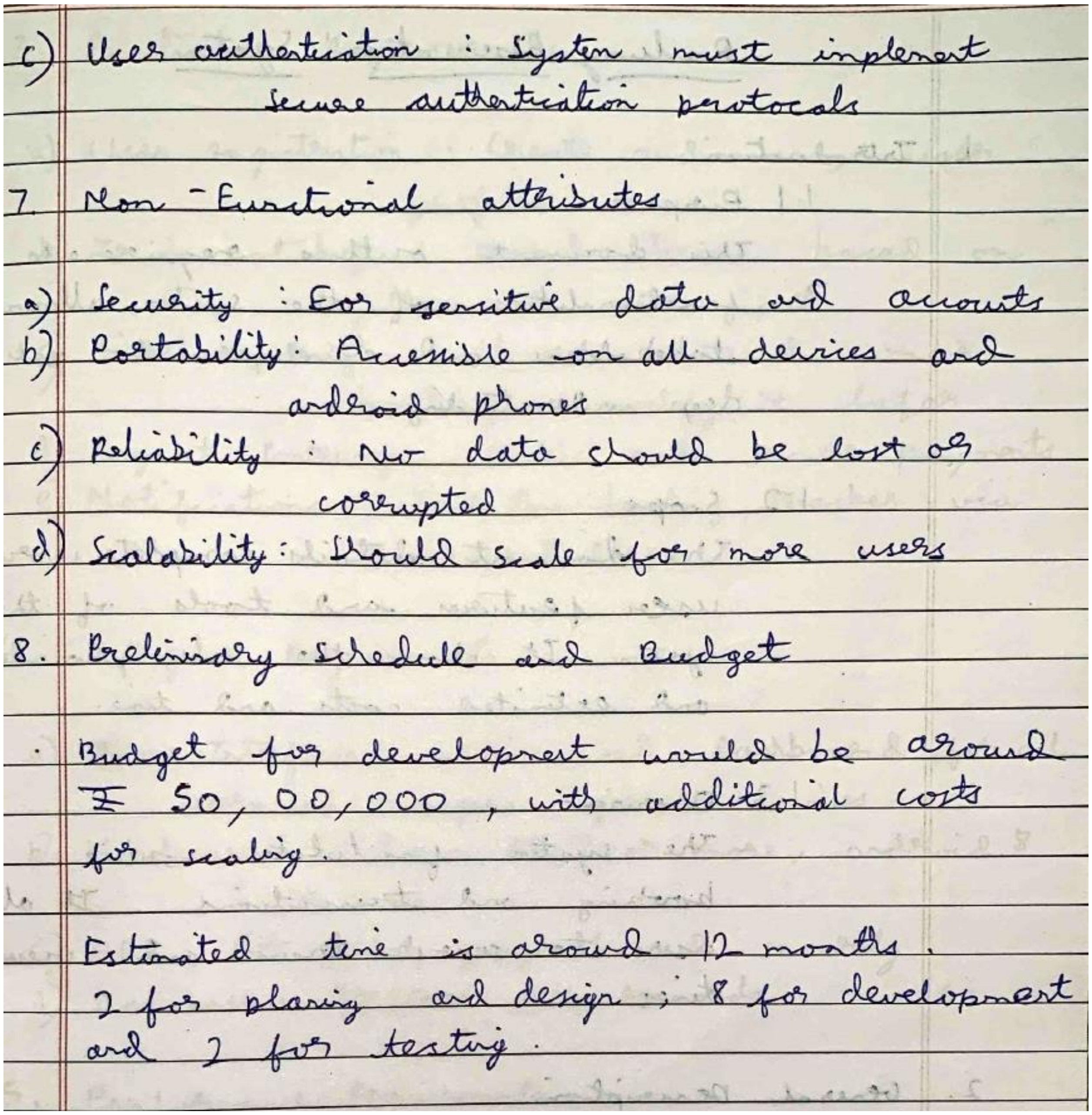
a) Frontend : UI/UX for forms and dashboard
b) API : For verification of email and number
c) Database : Secure connections to store all user data

5. Performance Requirements

a) Response time : Under 2 seconds for normal load
b) Throughput : Upto 500 concurrent customers
c) Data storage : Minimum 100,000 records and documents of users
d) Error rate : Very low 0.1%

6. Design Constraints

a) Technology : React and node JS for frontend
b) Database : Using RDBMS for relational data



c) User authentication : System must implement secure authentication protocols

7. Non - Functional attributes

a) Security : For sensitive data and accounts
b) Portability : Accessible on all devices and android phones
c) Reliability : No data should be lost or corrupted
d) Scalability : Should scale for more users

8. Preliminary schedule and Budget

· Budget for development would be around ₹ 50, 00,000 , with additional costs for scaling.

Estimated time is around 12 months .
2 for planing and design ; 8 for development and 2 for testing .

Class Diagram

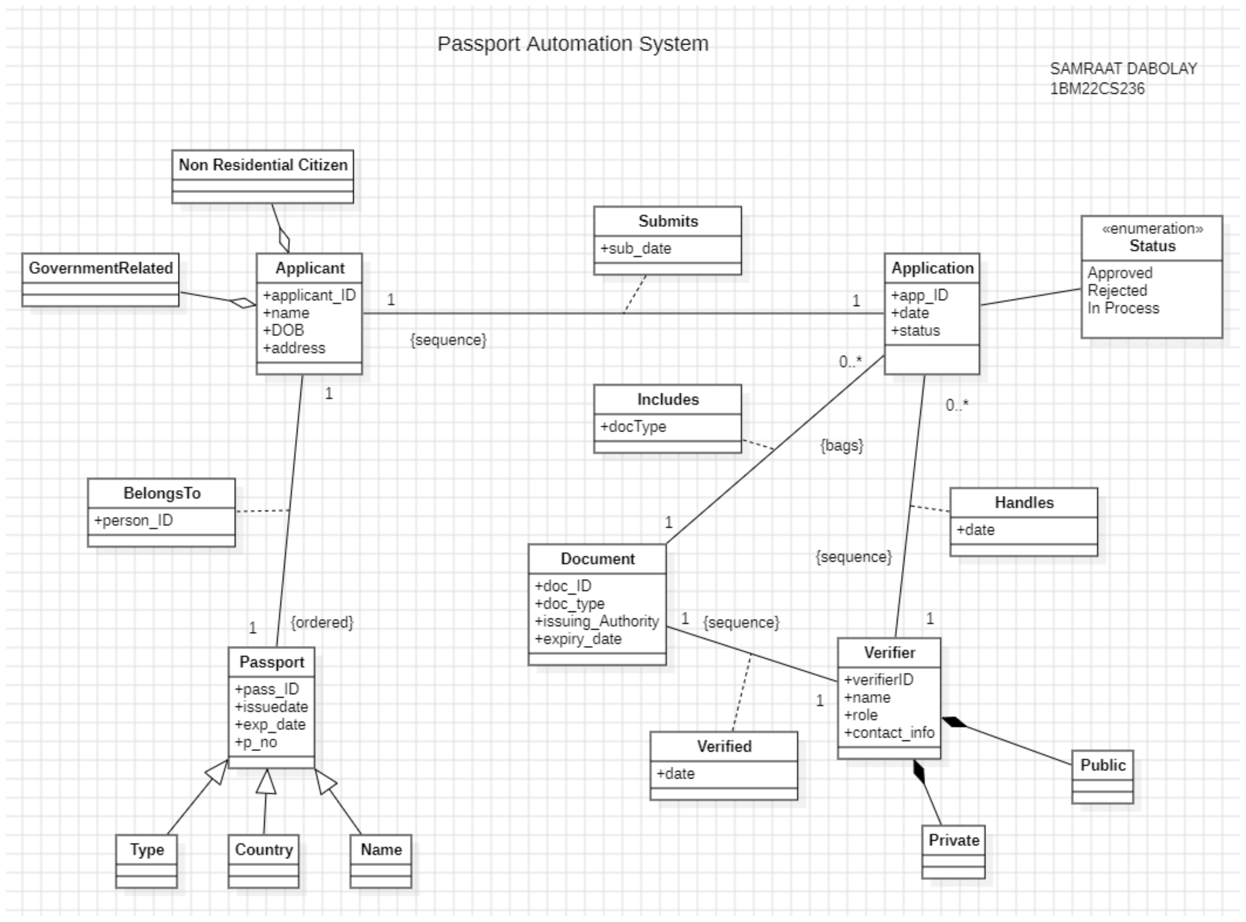


Fig 5.1

This ER Diagram presents a conceptual model for a Passport Automation System, outlining the key entities and their relationships. Entities like Non-Residential Citizen, Applicant, Document, Application, Passport, and Verifier are defined, each with relevant attributes. For instance, the Applicant entity includes attributes like name, date of birth, and address.

The diagram illustrates various relationships between these entities. A Non-Residential Citizen can become an Applicant, demonstrating a form of generalization where Applicant is a specialized type of Non-Residential Citizen. An Applicant submits an Application, and an Application can include multiple Documents, which is a clear example of aggregation as Documents can exist independently of a specific Application.

The Verifier entity is further specialized into Public and Private entities, indicating different types of verifiers with potentially different roles and responsibilities. This demonstrates inheritance where Public and Private verifiers inherit common properties from the general Verifier entity. The Application entity has a "Status" attribute defined as an enumeration, which restricts the possible values for the status to a predefined set (Approved, Rejected, In Process). This ensures data consistency and reduces the possibility of invalid status values.

State Diagram

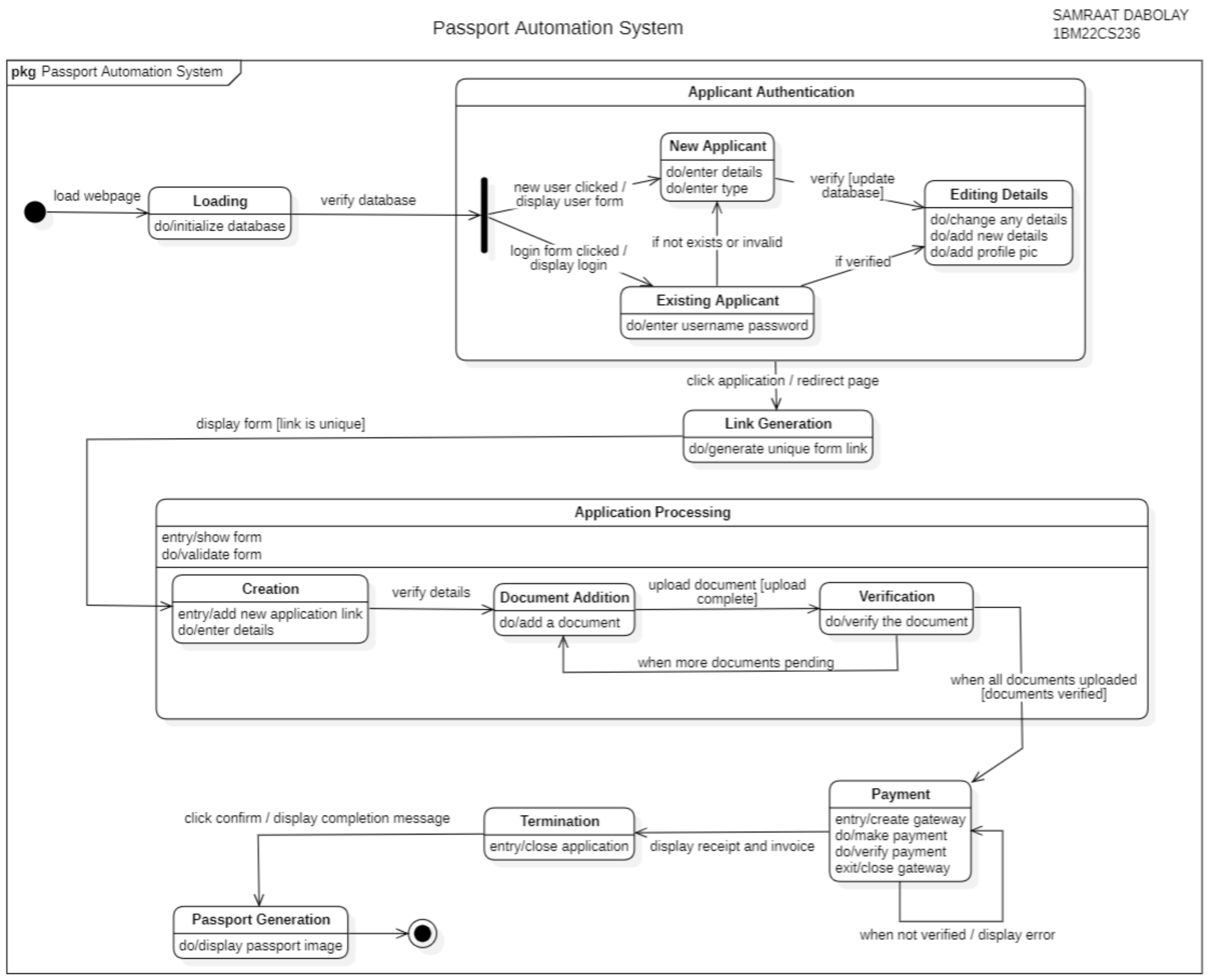


Fig 5.2

This diagram depicts the workflow involved in the passport application process. It starts with the "Loading" state, where the system initializes the database and loads the webpage. The "Applicant Authentication" state is a composite state, encompassing the activities for new and existing applicants. For new applicants, the system verifies their details and creates a new user account. For existing applicants, the system verifies their login credentials.

After authentication, the system generates a unique link for the application process. The "Application Processing" state is also a composite state, encompassing activities like application creation, document addition, verification, and payment. In the "Creation" activity, the applicant enters their application details. The "Document Addition" activity allows the applicant to upload required documents. The "Verification" state handles the verification of uploaded documents.

Upon successful verification and payment, the system transitions to the "Passport Generation" state, where the passport image is generated and displayed. If any step fails, the system transitions to the "Termination" state with an appropriate error message.

Use-Case Diagram

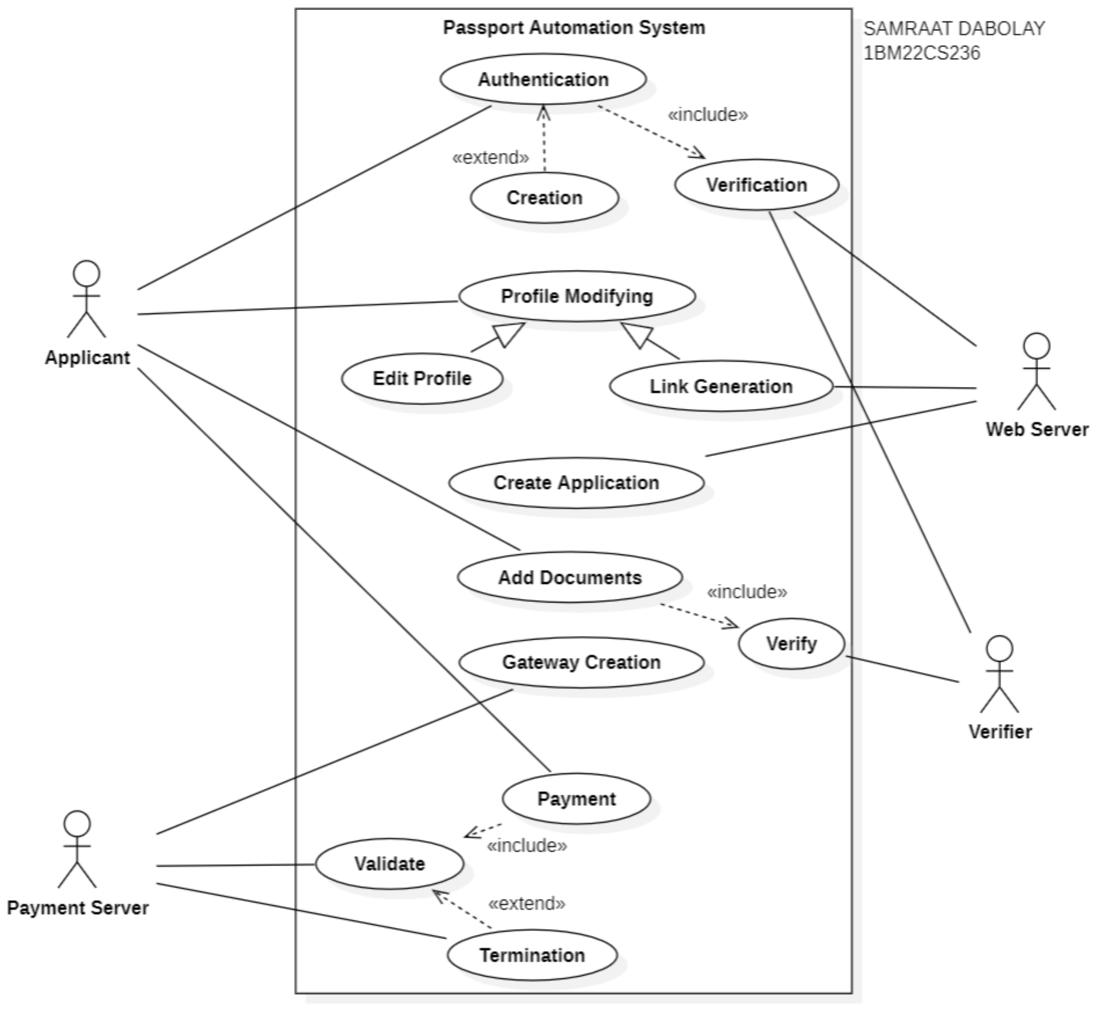


Fig 5.3

This diagram depicts the various functionalities and interactions within a passport automation system. It illustrates three key actors: Applicant, Web Server, and Verifier.

The diagram shows several core use cases. The "Authentication" use case includes the "Verification" use case, indicating that verification is an essential step within the authentication process. The "Creation" use case extends the "Authentication" use case, suggesting that account creation is an optional or additional step after authentication.

The "Profile Modifying" use case is further specialized into "Edit Profile" and "Link Generation," indicating that profile modification can involve editing personal details or generating a link for accessing the application.

The "Create Application" use case is a core function for applicants. The "Add Documents" use case includes the "Verify" use case, indicating that document verification is an essential step within the document addition process.

The "Payment" use case includes the "Validate" use case, suggesting that payment validation is an essential step within the payment process. The "Termination" use case can be extended by various scenarios, indicating different ways in which the application process can be terminated.

Sequence Diagram

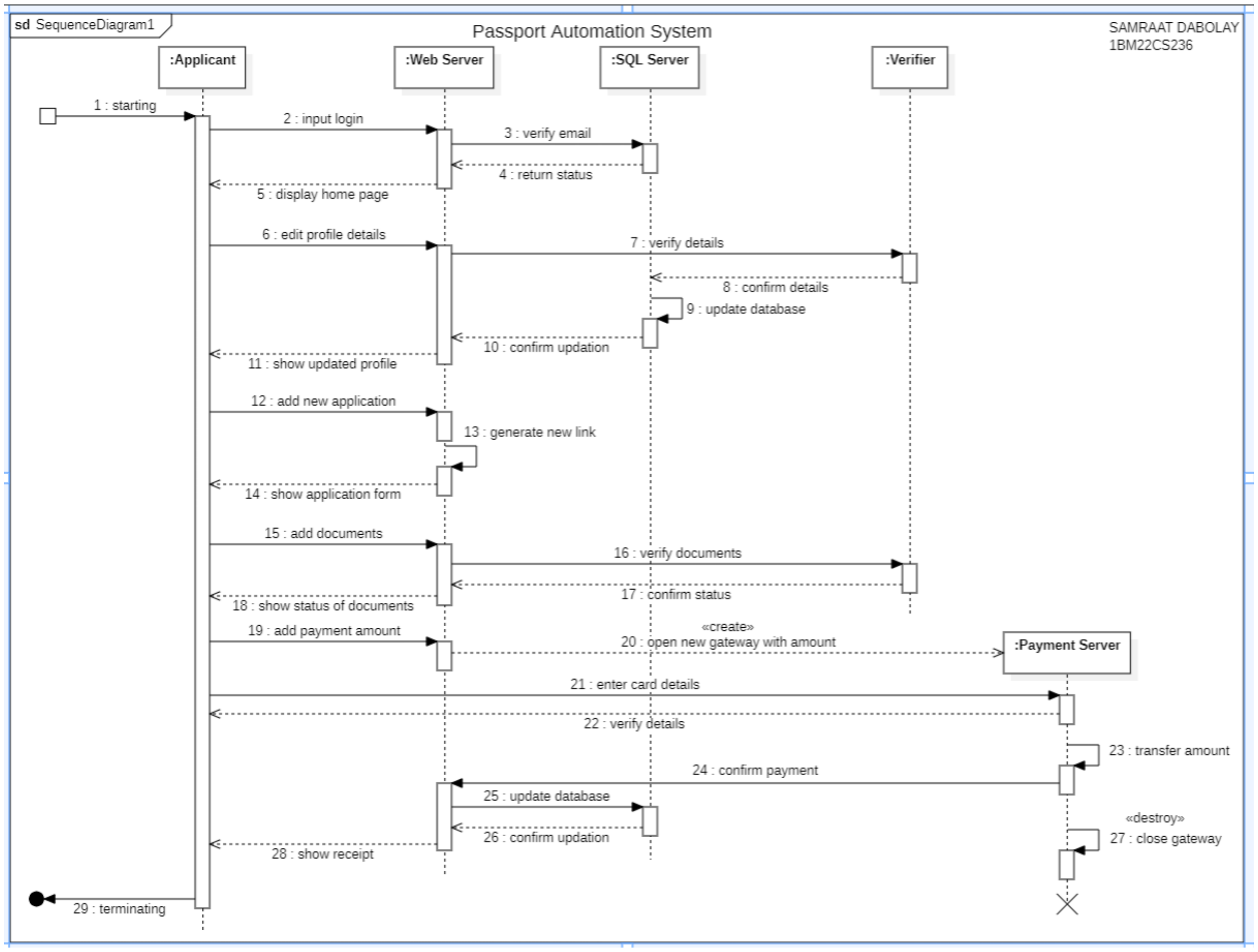


Fig 5.4

This diagram depicts the interactions between various components of the system during a passport application process. The key actors involved are the Applicant, Web Server, SQL Server, Verifier, and Payment Server.

The sequence starts with the Applicant initiating the process. The Applicant inputs their login credentials, which are sent to the Web Server. The Web Server verifies the email and returns the status to the Applicant.

If the login is successful, the Web Server displays the home page to the Applicant. The Applicant can then edit their profile details. These details are sent to the Web Server, which verifies them and updates the database. The Web Server then confirms the updation to the Applicant.

The Applicant can then add a new application. The Web Server generates a unique link for this application. The Applicant is then presented with the application form and can add the required documents. The Verifier verifies these documents and sends the status back to the Web Server.

If the documents are verified, the Web Server creates a new instance of the Payment Server (a transient lifeline as it is only created for this specific interaction). The Applicant is prompted to enter their payment details, which are verified by the Payment Server. The Payment Server then transfers the amount and confirms the payment.

Finally, the Web Server updates the database, confirms the updation, and displays a receipt to the Applicant. The Payment Server closes the gateway and is destroyed. The Applicant then terminates the session.

This sequence diagram demonstrates the dynamic flow of interactions between the various components of the system. It highlights the use of a transient lifeline for the Payment Server, which is created dynamically for each payment transaction. The diagram also showcases the interplay between active and passive objects. The Web Server and Database Server are primarily active, processing requests and responding to queries, while the Applicant and Verifier are more passive, providing input and receiving information.

Activity Diagram

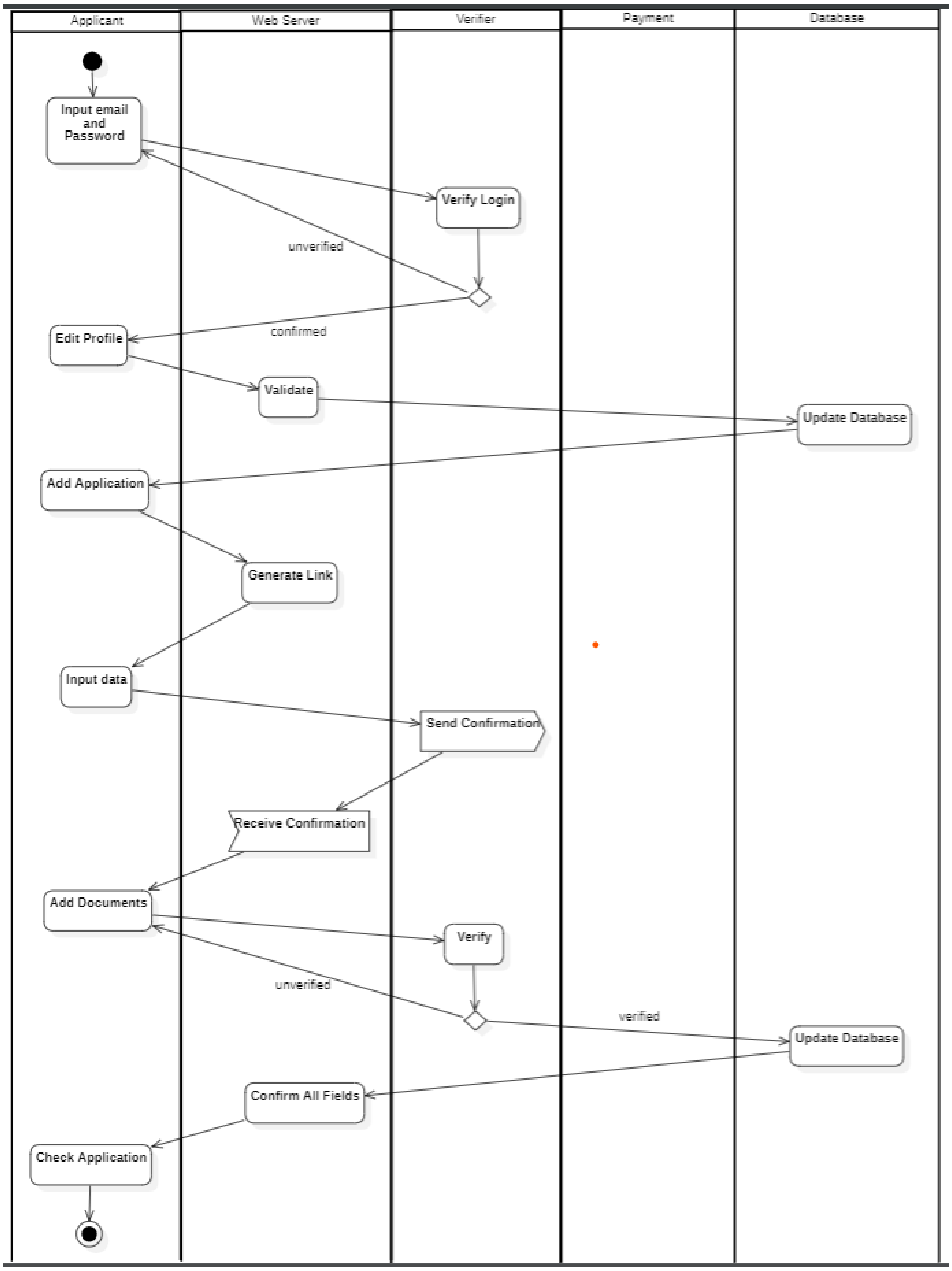


Fig 5.5

This diagram depicts the workflow of a passport application system, organized into swimlanes to represent the different actors involved: Applicant, Web Server, Verifier, Payment, and Database.

The process begins within the Applicant swimlane with the "Input email and Password" activity. The Web Server, in its respective swimlane, verifies the login credentials. If successful, the Applicant can proceed to "Edit Profile."

The Applicant can then initiate the "Add Application" activity. The Web Server generates a unique link for the application and sends a confirmation signal to the Applicant.

The Applicant receives the confirmation and proceeds to "Add Documents." The Verifier, in their swimlane, verifies the uploaded documents. If successful, the Applicant proceeds to the "Check Application" activity.

This Activity Diagram effectively visualizes the sequence of activities and decision points within the passport application system, highlighting the interactions between the different actors and the flow of information between them. The use of swimlanes enhances the clarity of the diagram by clearly demarcating the responsibilities and actions of each actor involved in the process. The inclusion of signals further emphasizes the communication and information exchange between different components of the system.