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

Lab Record

Object-Oriented Modeling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

ANAGHA BHARADWAJ
1BM22CS038

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 Modeling (22CS5PCOOM) laboratory has been carried out by Anagha Bharadwaj (1BM22CS038) during the 5th Semester Oct 24-Jan2025.

Signature of the Faculty Incharge:

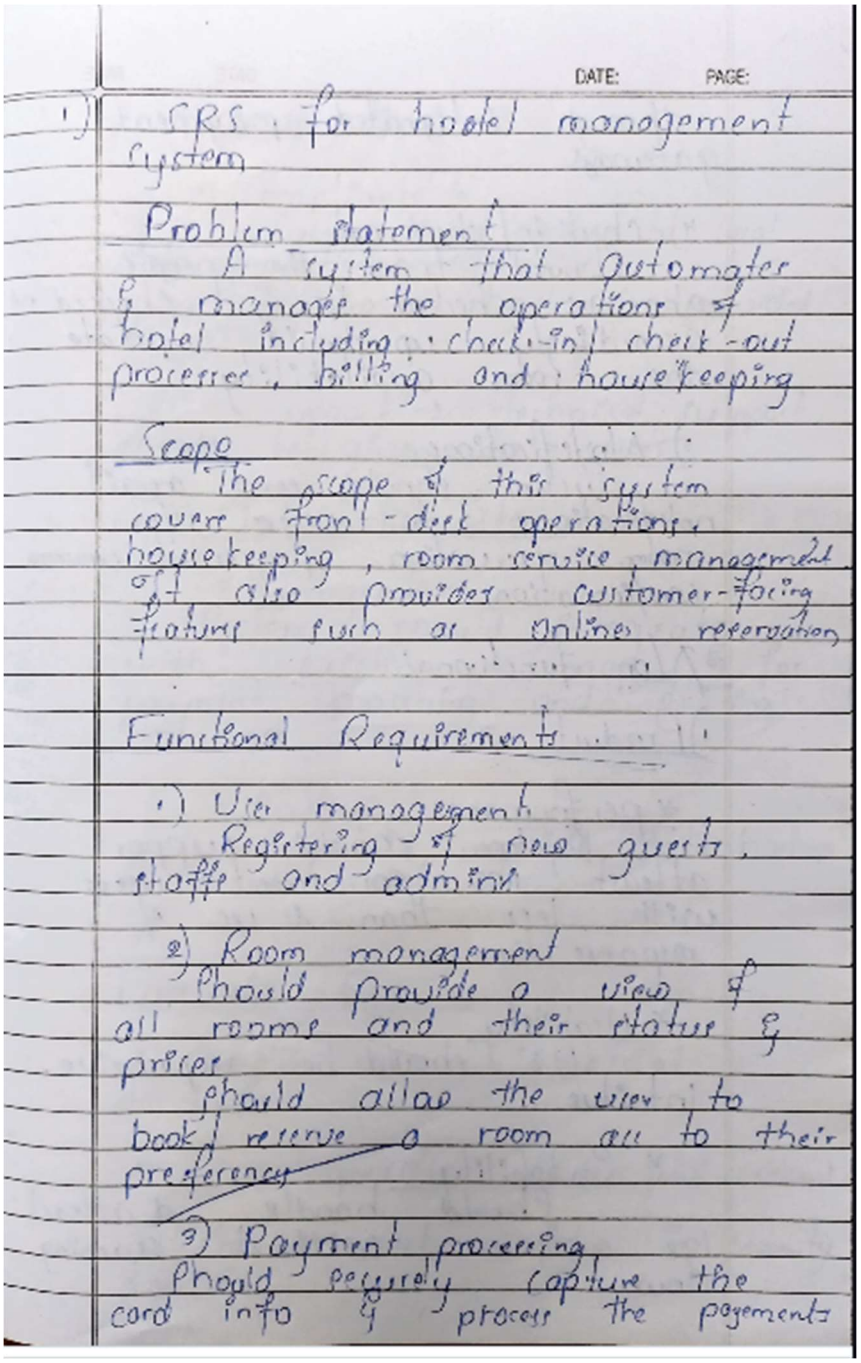
Dr Laxmi Neelima
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

1. Hotel Management System

1.1 Problem Statement: Hotels often face challenges in managing their daily operations efficiently, such as handling reservations, managing room availability, billing, staff management, and customer service. Traditional systems or manual methods are time-consuming, error-prone, and do not scale well with growing customer demands.

1.2 SRS-Software Requirements Specification



DATE: PAGE:

1) SRS for hotel management system

Problem statement

A system that automates & manages the operations of hotel including: check-in/ check-out processes, billing and housekeeping

Scope

The scope of this system covers front-desk operations, housekeeping, room service management & also provides customer-facing features such as online reservation,

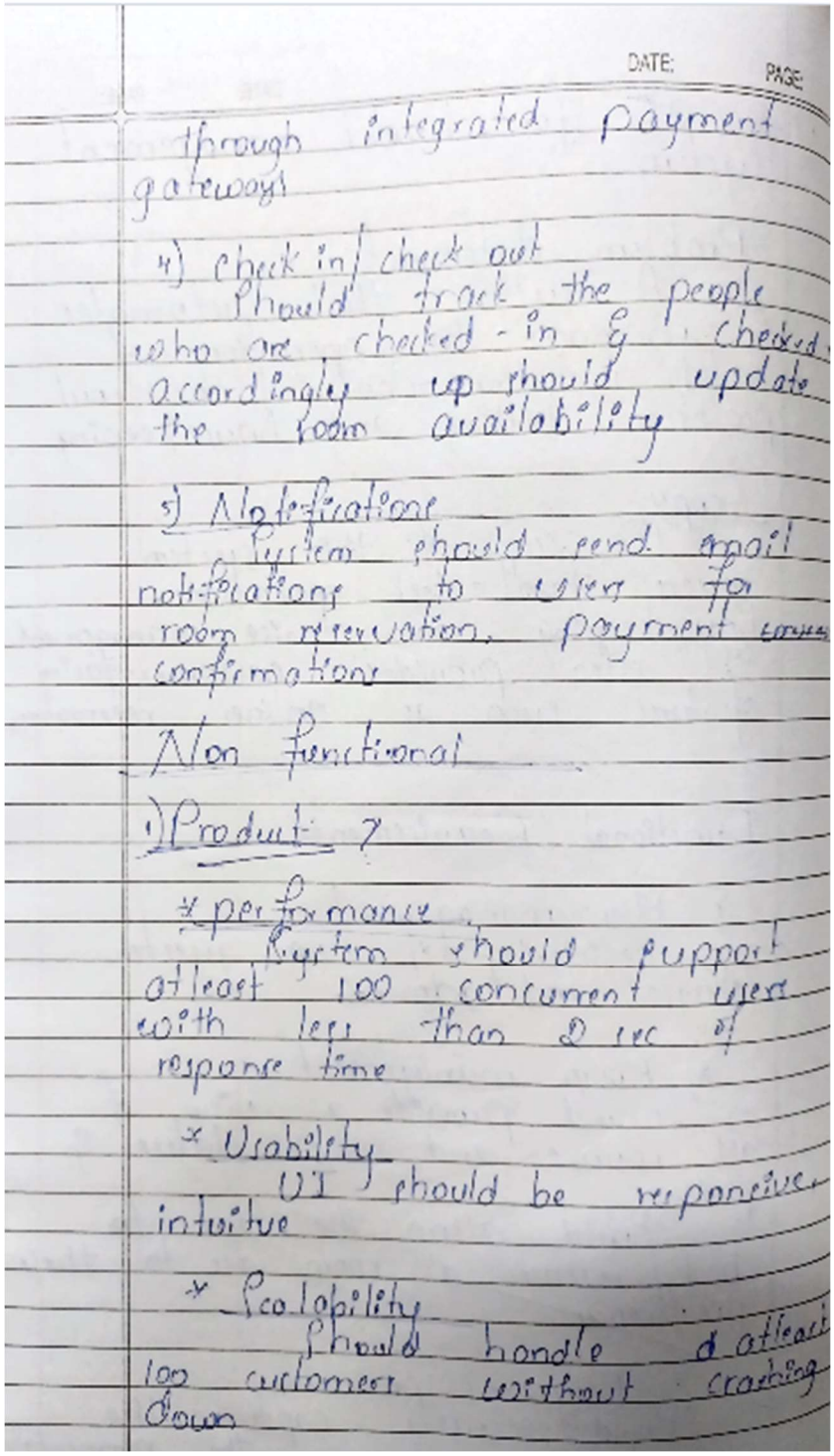
Functional Requirements

1) User management
Registering of new guests, staffs and admins

2) Room management
Should provide a view of all rooms and their status & prices
Should allow the user to book/ reserve a room acc to their preferences

3) Payment processing
Should securely capture the card info & process the payements

Fig 1.2.1


DATE: PAGE:

through integrated payment gateways

4) check in/ check out
Should track the people who are checked-in & checked accordingly up should update the room availability

5) Notifications
System should send email notifications to user for room reservation, payment confirmations

Non functional

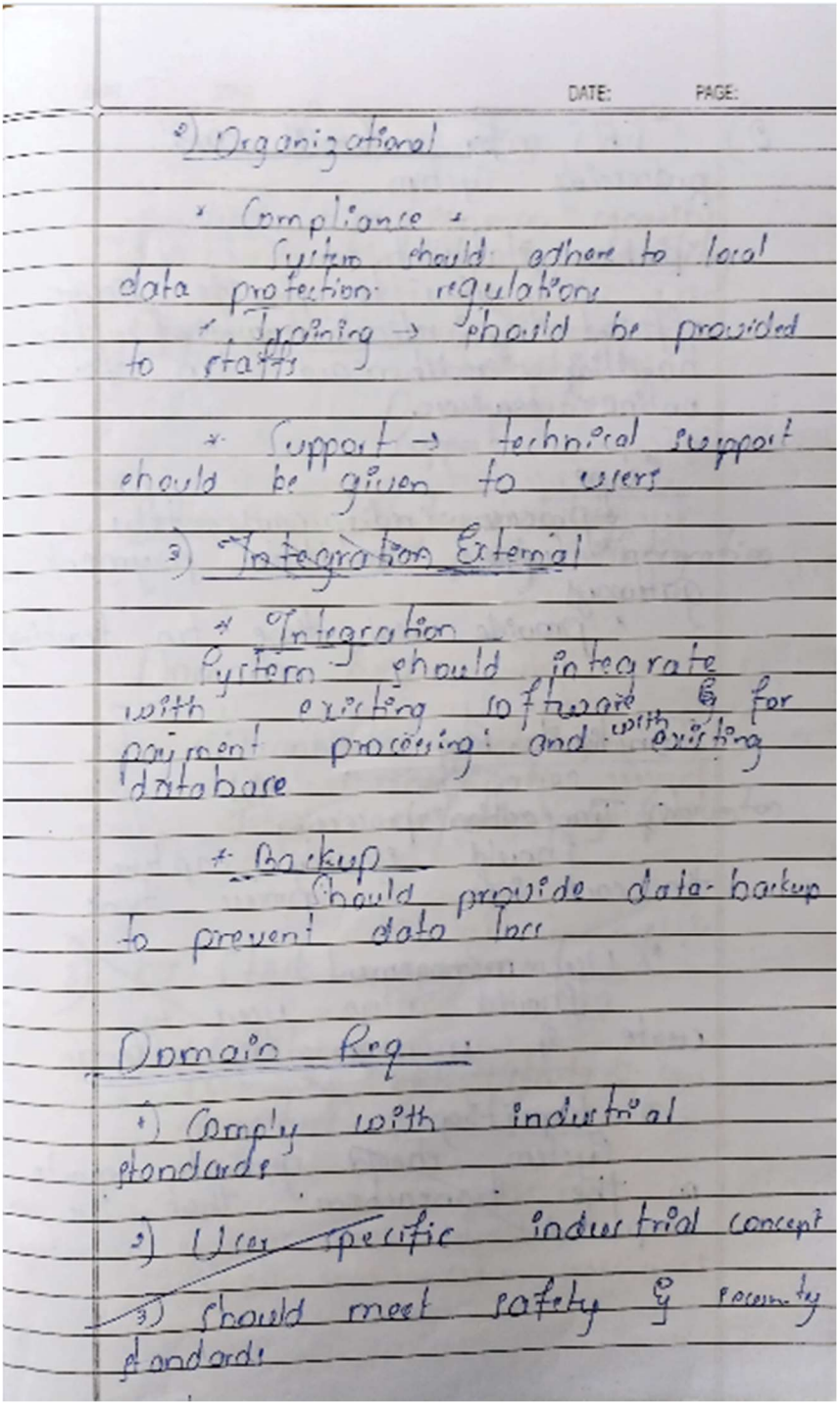
1) Product ?

* performance
System should support atleast 100 concurrent users with less than 2 sec of response time

* Usability
UI should be responsive, intuitive

* Scalability
Should handle atleast 100 customers without crashing down

Fig 1.2.2


DATE: PAGE:

2) Organizational

* Compliance :
System should adhere to local data protection regulations

* Training → should be provided to staffs

* Support → technical support should be given to users

3) Integration External

* Integration
System should integrate with existing software & for payment processing and with existing database

* Backup
Should provide data backup to prevent data loss

Domain Req :

1) Comply with industrial standards

2) User specific industrial concept

3) Should meet safety & security standards

Fig 1.2.3

1.3 Class Diagram

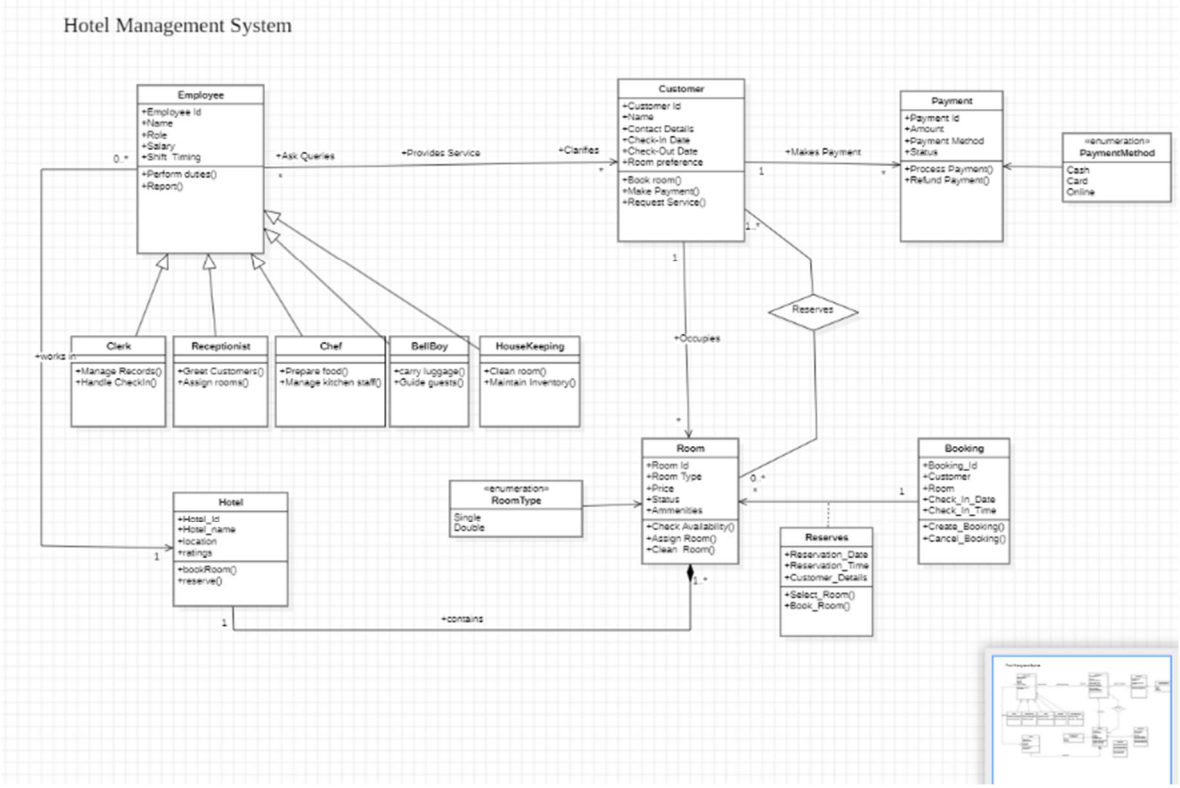


Fig 1.3.1

The class diagram represents the structure of a **Hotel Management System**, detailing the relationships and interactions among various entities. The primary entities include Employee, Customer, Room, Booking, Reserves, Payment, and Hotel. The Employee class serves as the parent for roles such as Clerk, Receptionist, Chef, BellBoy, and Housekeeping, each having specific duties like managing records, assigning rooms, preparing food, assisting guests, and maintaining room cleanliness. The Customer class holds details such as name, contact information, check-in/out dates, and room preferences, along with methods for booking rooms, making payments, and requesting services. The Room class defines attributes like room type (Single or Double), price, status, and amenities, and includes methods for checking availability, assigning, and cleaning rooms.

The Booking class manages the room booking process with details such as booking ID, customer, room, and check-in time, while the Reserves class handles room reservations and facilitates the selection and booking of rooms. The Payment class manages financial transactions, supporting various payment methods (cash, card, or online) and processes such as payment and refund handling.

1.4 State Diagram

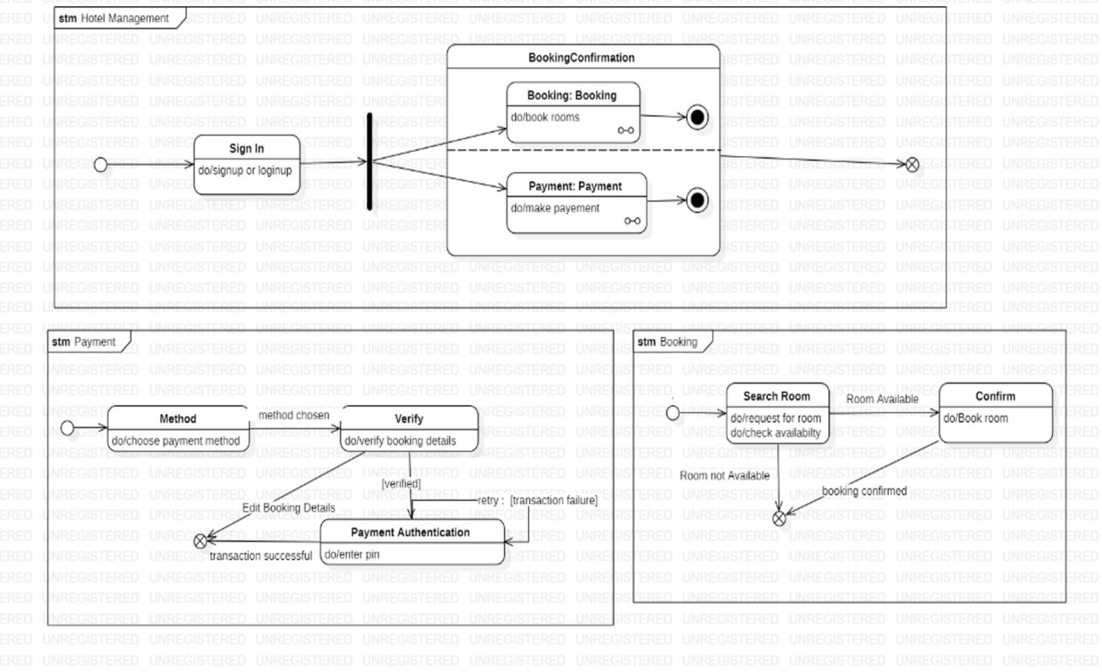


Fig 1.4.1

The state diagram represents a Hotel Management System and outlines the workflow for managing hotel operations efficiently. It begins with the **Staff Login**, where employees access the system. The primary state is the **Staff Menu**, which branches into three key functions: **Room Status**, **Check-In**, and **Check-Out**. In **Room Status**, staff can view, update, and display the current status of rooms (e.g., vacant, occupied, or under maintenance). The **Check-In** process involves assigning a room, preparing it with necessary essentials, and allowing guests to occupy it. The **Check-Out** process handles clearing the room, cleaning it, and updating its status for the next guest. Finally, the system transitions to the **Staff Logout**, where employees log out after completing their tasks. This diagram ensures a structured approach to hotel management, covering room readiness, guest handling, and status updates.

1.5 Use Case Diagram

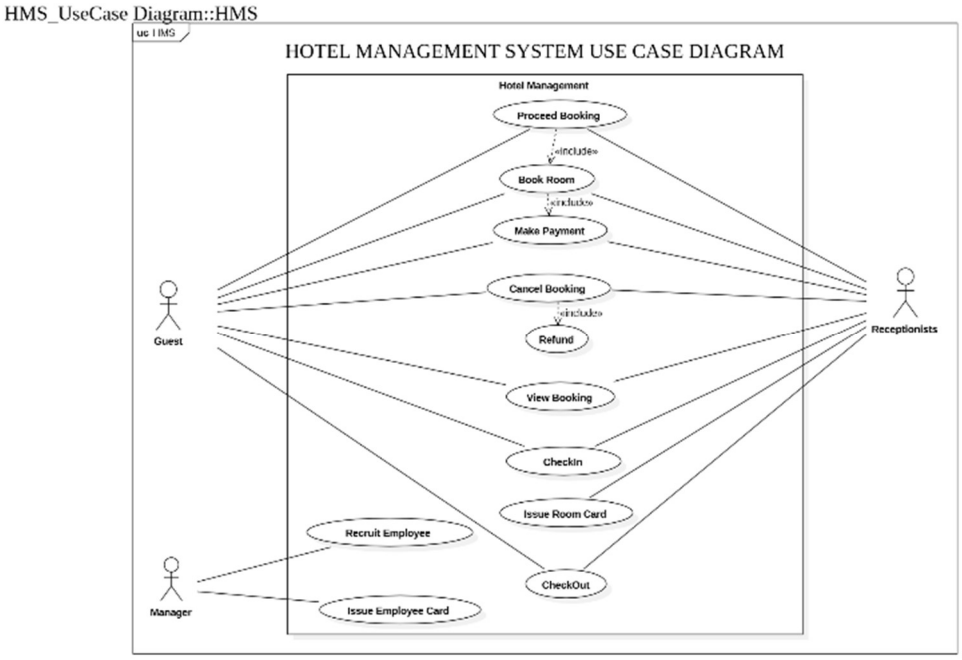


Fig 1.5.1

The use case diagram illustrates the functionality of a Hotel Management System by detailing the interactions between different actors and system processes. The primary actors include **Guests**, **Receptionists**, and a **Manager**. Guests can perform actions such as booking rooms, making payments, canceling bookings, requesting refunds, viewing bookings, checking in, and checking out. Receptionists facilitate these operations by assisting with room bookings, issuing room cards, and managing the check-in and check-out processes. The **Manager** has additional responsibilities like recruiting employees and issuing employee cards. Key use cases such as **Proceed Booking**, **Book Room**, and **Make Payment** are interconnected to ensure a smooth booking workflow. The system is designed to handle cancellations and refunds effectively while maintaining efficient guest services. This diagram provides a clear overview of the roles and responsibilities within the hotel management workflow.

1.6 Sequence Diagram

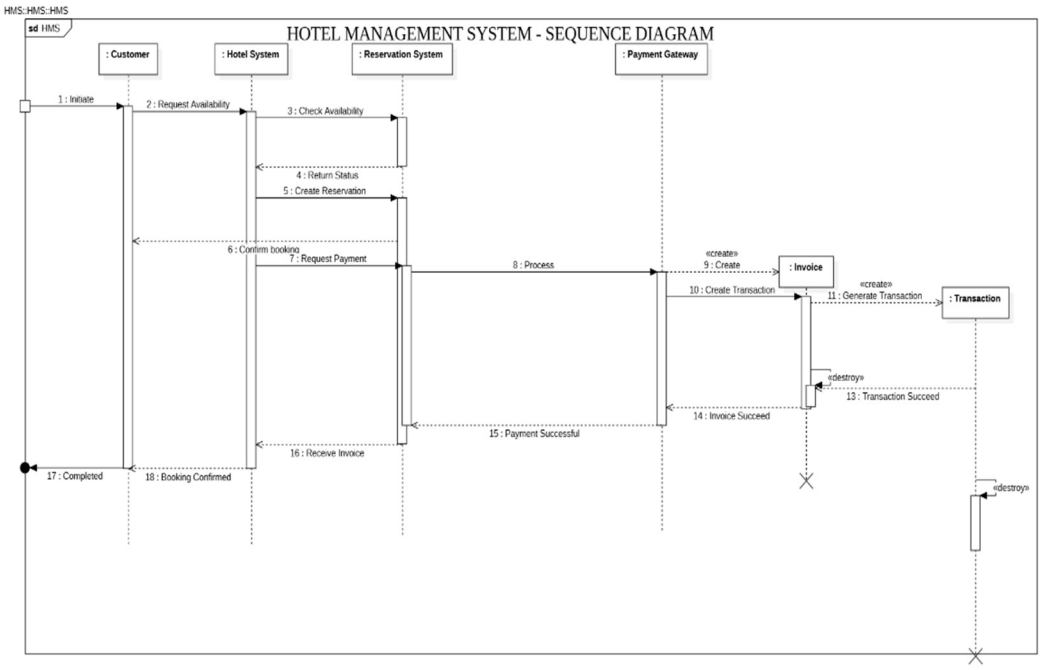


Fig 1.6.1

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 through the **Hotel System**, which communicates with the **Reservation System** to verify availability. The **Reservation System** returns the status to the **Hotel System**, and if a room is available, a reservation is created. The **Hotel System** then confirms the booking and sends a payment request to the **Payment Gateway**. The **Payment Gateway** processes the payment by generating an invoice and a transaction. Once the transaction is successful, the invoice is returned to the **Hotel System**, which acknowledges the payment as successful. Finally, the customer receives the booking confirmation, marking the process as completed. This diagram ensures a streamlined interaction between customer requests, reservation handling, and secure payment processing.

1.7 Activity Diagram

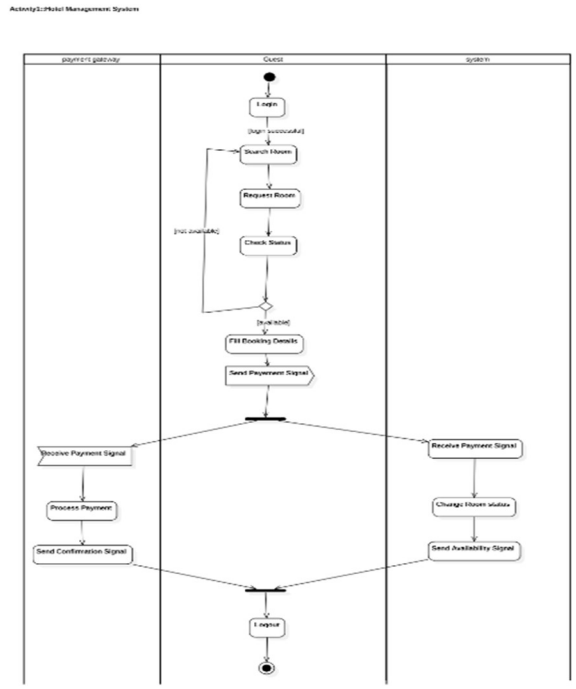


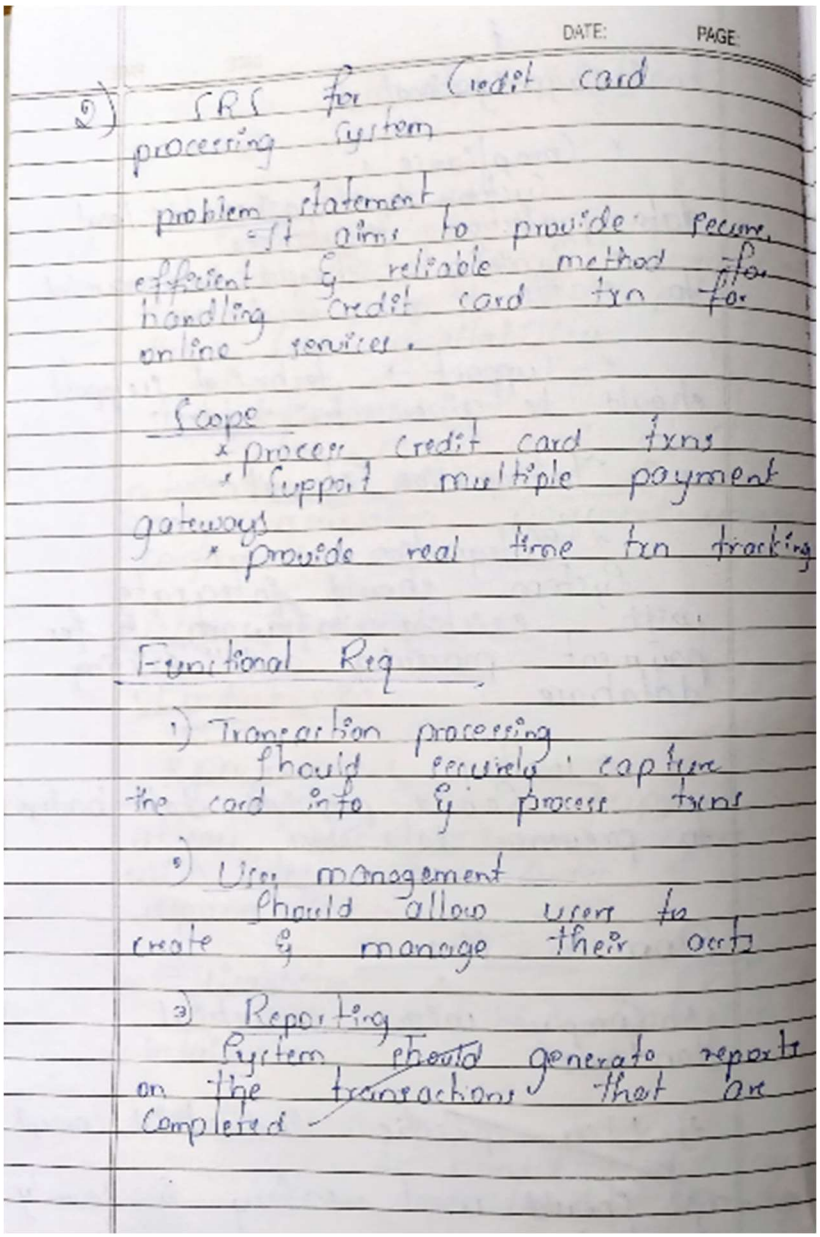
Fig 1.7.1

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 Processing 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-Software Requirements Specification


DATE: PAGE:

2) SRS for Credit card processing system

problem statement
- It aims to provide secure, effficient & reliable method for handling credit card txn for online services.

Scope
* process credit card txns
* Support multiple payment gateways
* provide real time txn tracking

Functional Req
1) Transaction processing
Should securely capture the card info & process txns
2) User management
Should allow users to create & manage their accts
3) Reporting
System should generate reports on the transactions that are completed

Fig 2.2.1

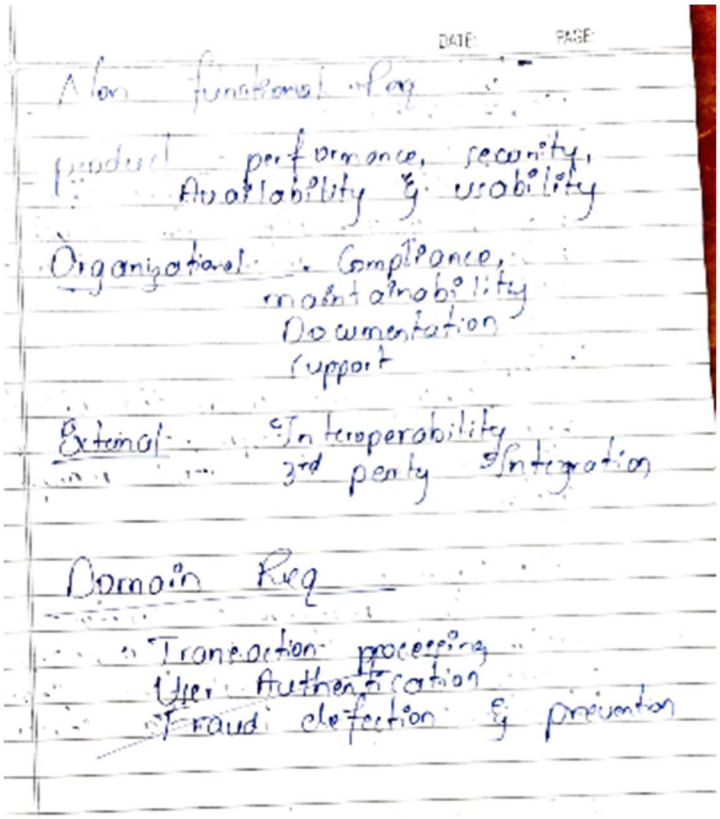

DATE
PAGE
Non functional Req
Product performance, security,
Availability & usability
Organizational Compliance,
maintainability
Documentation
Support
External Interoperability
3rd party Integration
Domain Req
Transaction processing
User Authentication
Fraud detection & prevention

Fig 2.2.2

2.3 Class Diagram

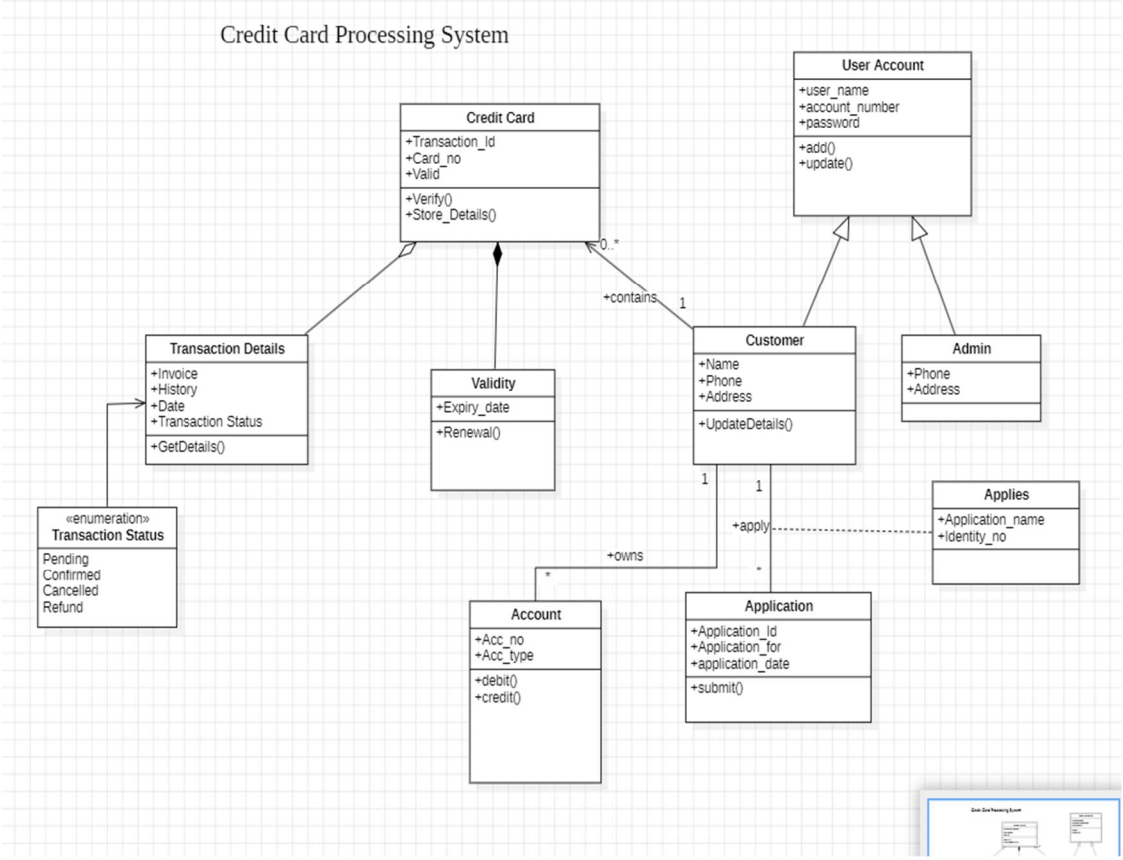


Fig 2.3.1

This UML class diagram models a credit card processing system. It shows that a Transaction uses a Credit Card and contains Transaction Details. Each Credit Card has a Validity status. A Customer can have multiple Accounts and submit Applications for credit cards, which are also linked to an Account. An Admin manages Customers and their Applications. The diagram also lists attributes for each class, like card number for Credit Card and transaction date for Transaction, as well as operations like verify() for Credit Card, debit() and credit() for Account, and submit() for Application. This structure defines the core components and their interactions within the credit card processing system.

2.4 State Diagram

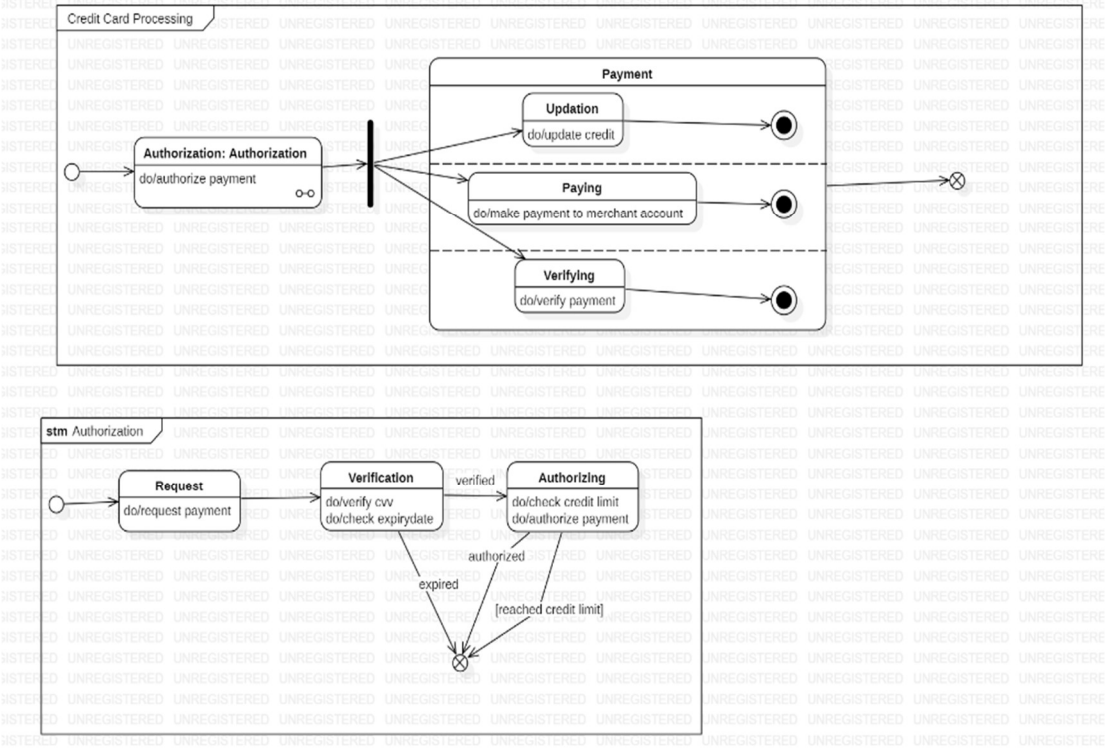


Fig 2.4.1

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 Use Case Diagram

CPS_UseCase Diagram::CPS

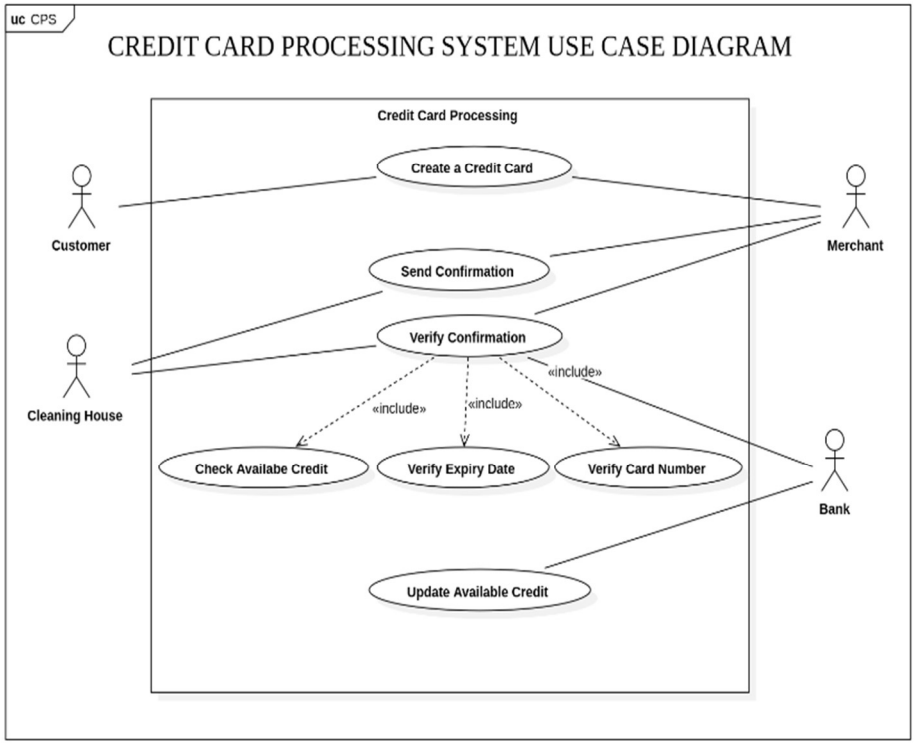


Fig 2.5.1

This use case diagram illustrates interactions within a Credit Card Processing System (CPS). It shows four actors: Customer, Merchant, Cleaning House (likely for transaction reconciliation), and Bank. The primary use case is "Credit Card Processing," which involves several included use cases. The Customer can "Create a Credit Card." The Merchant "Sends Confirmation" of a transaction, which triggers the "Verify Confirmation" use case. This verification process *includes* three sub-processes handled by the Bank: "Check Available Credit," "Verify Expiry Date," and "Verify Card Number." Finally, the Bank also "Updates Available Credit" after a transaction. The diagram highlights the dependencies and interactions between the different actors and their respective roles in the credit card transaction process.

2.6 Sequence Diagram

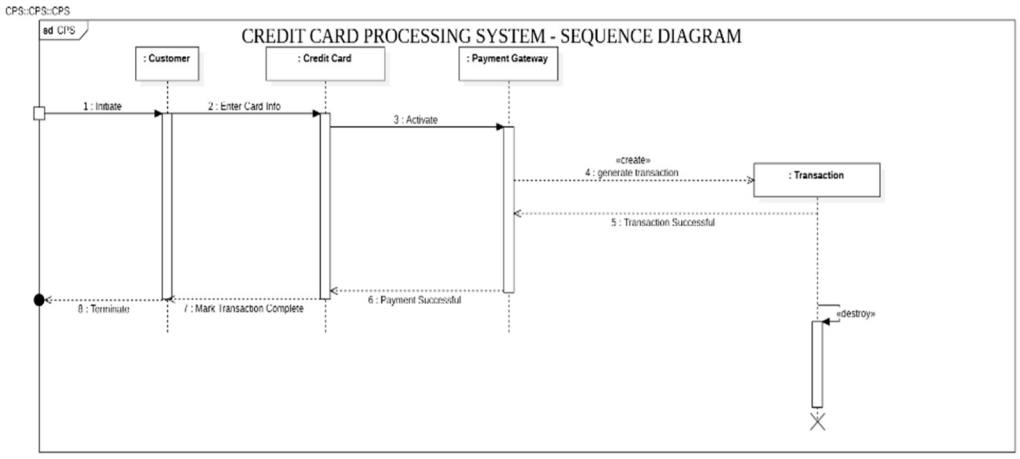


Fig 2.6.1

This sequence diagram illustrates the flow of a credit card transaction. It involves three participants: Customer, Credit Card (representing the card itself/card reader interaction), and Payment Gateway. The process begins with the Customer initiating a transaction (1: Initiate). The Customer then enters their card information (2: Enter Card Info), which activates the Payment Gateway (3: Activate). The Payment Gateway then generates a new Transaction object (4: generate transaction). Assuming the transaction is successful, the Payment Gateway sends a success message (5: Transaction Successful) back to the Customer. The Customer then marks the transaction as complete (Mark Transaction Complete), and the Payment Gateway confirms payment success (6: Payment Successful). Finally, the Transaction object is destroyed, completing the sequence. In short, the diagram shows the step-by-step interaction between the customer, their card, and the payment system during a successful transaction.

2.7 Activity Diagram

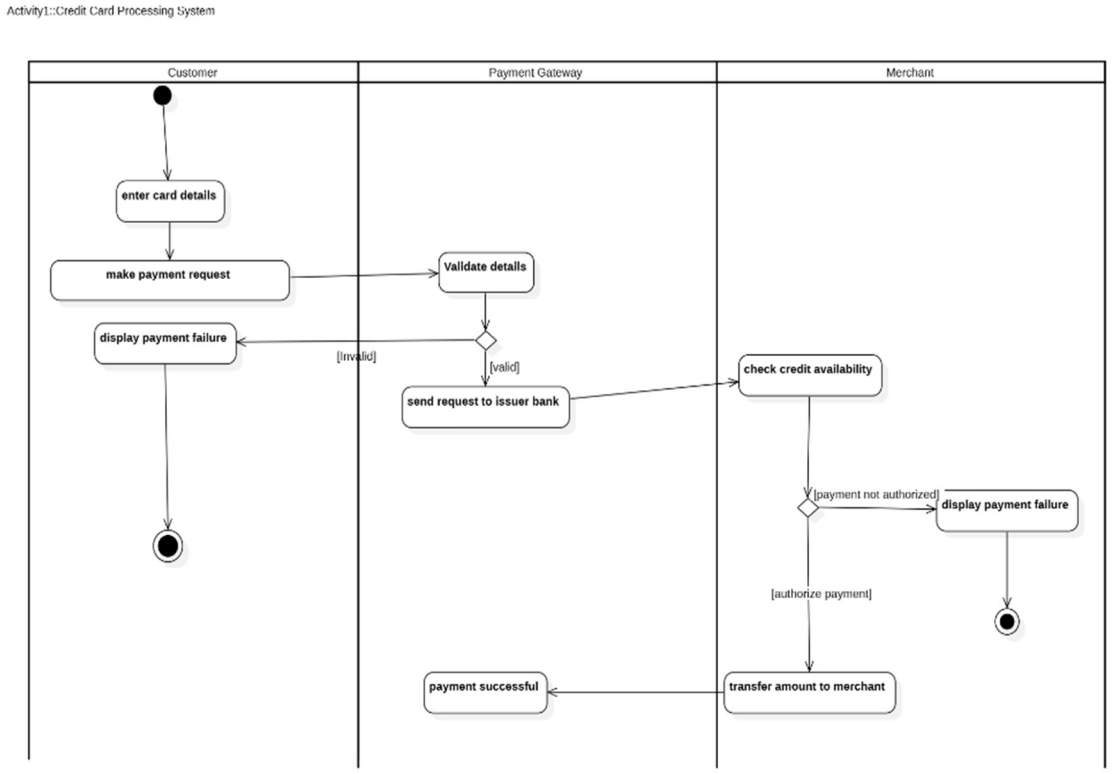


Fig 2.7.1

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 Management 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-Software Requirements Specification

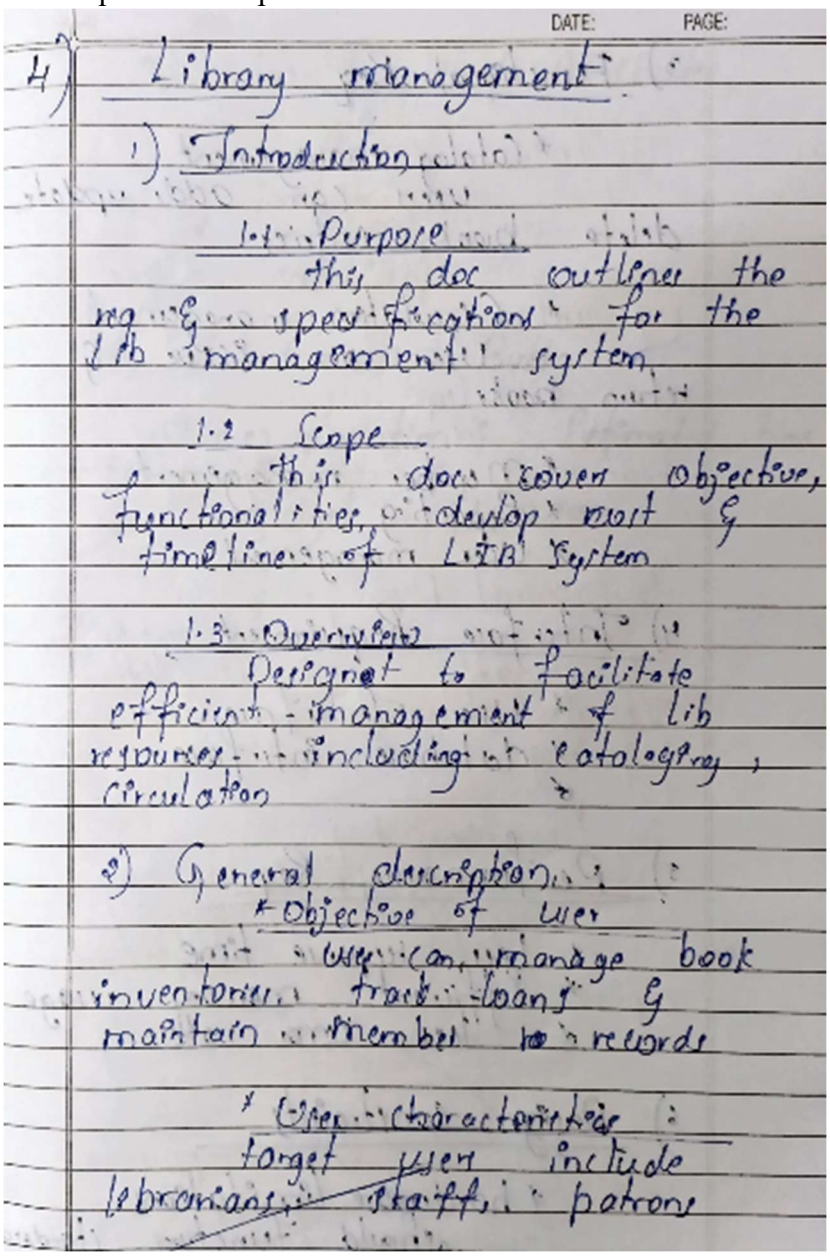


Fig 3.2.1

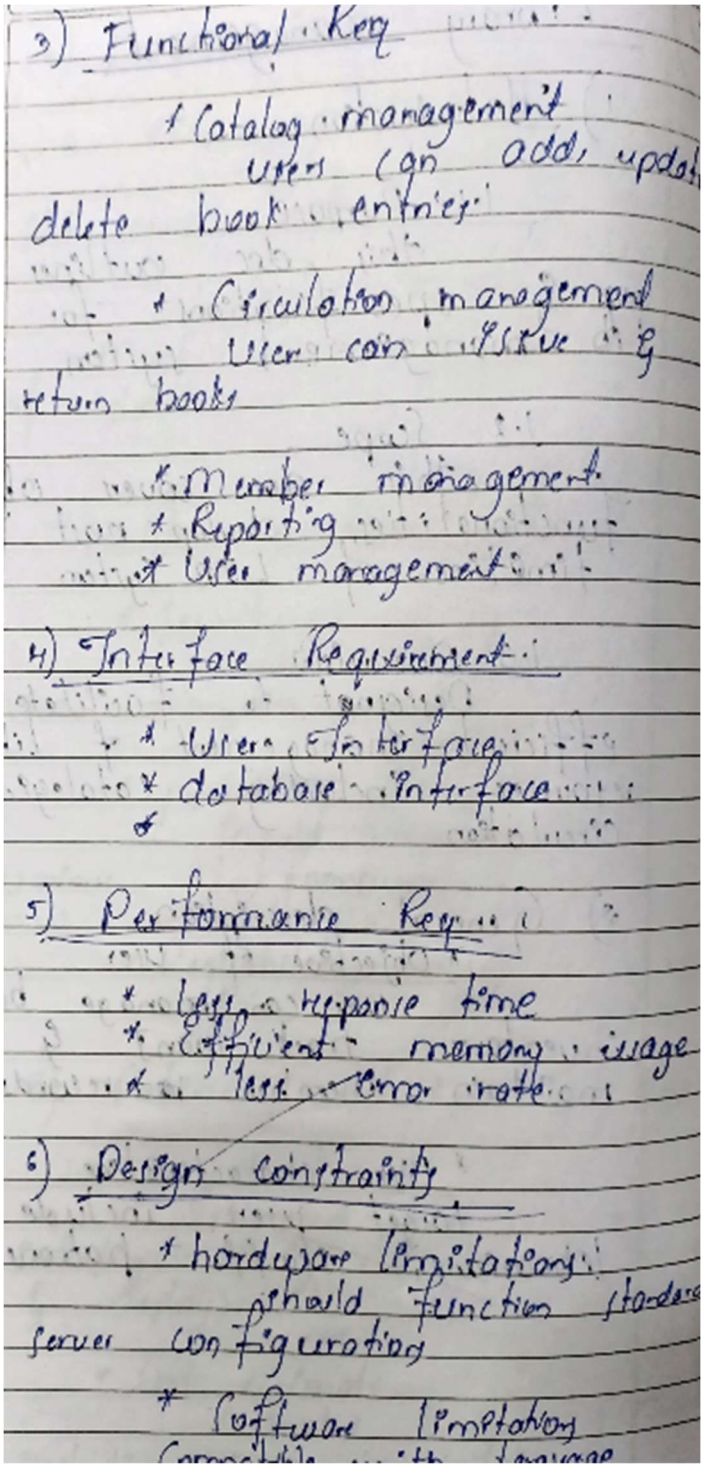

3) Functional Req
* Catalog management
Users can add, updat
delete book entries
* Circulation management
User can issue &
return books
* Member management
* Reporting
* User management
4) Interface Requirement
* User Interface
* database Interface
5) Performance Req
* less response time
* Efficient memory usage
* less error rate
6) Design constraints
* hardware limitations
should function standa
server configuration
* Software limitations

Fig 3.2.2

3.3 Class Diagram

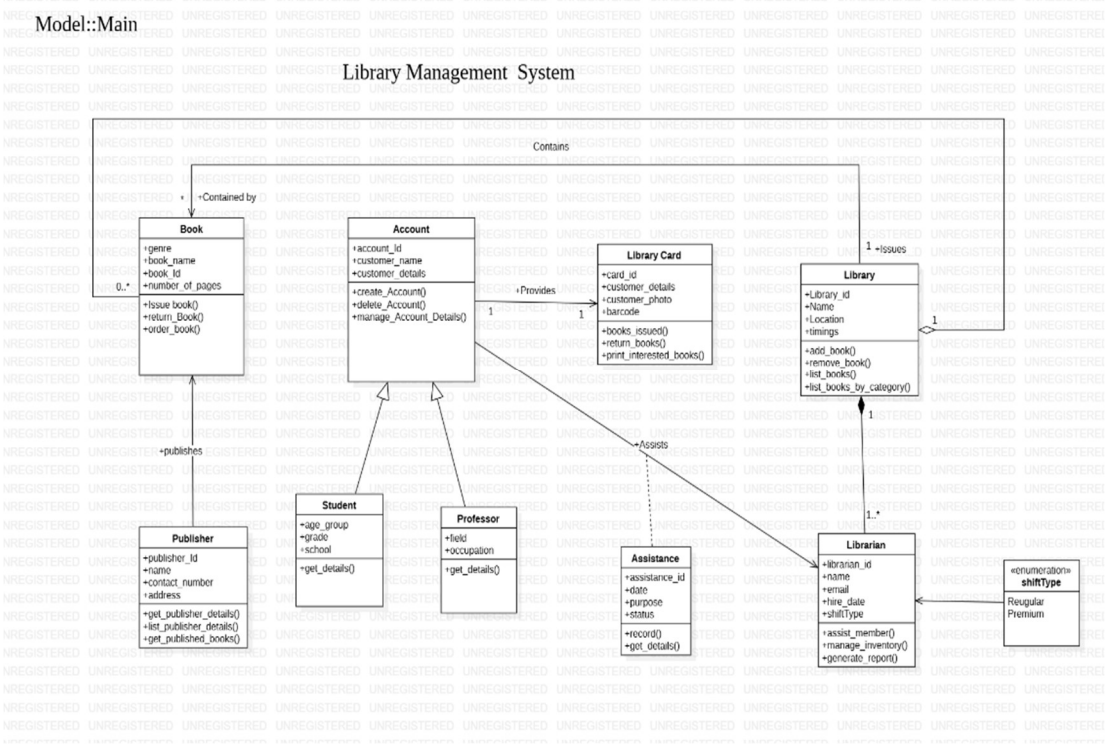


Fig 3.3.1

This UML depicts a class diagram for a Library Management System. It outlines the key entities involved and their relationships. The core entities include Book (with attributes like genre, name, ID, and number of pages, and operations like issuing, returning, and ordering), Account (with customer details and account management operations), Library Card (linking accounts to the library and tracking issued books), and Library (containing books, managing inventory, and handling book lists). These are further connected to Publisher (responsible for publishing books) and two types of library members: Student and Professor, both inheriting from Account. Finally, a Librarian assists members and manages the library, potentially with different ShiftType (Regular or Premium), and an Assistance entity tracks the details of assistance provided by librarians. The diagram illustrates how these entities interact, providing a structured view of the system's data and functionality.

3.4 State Diagram

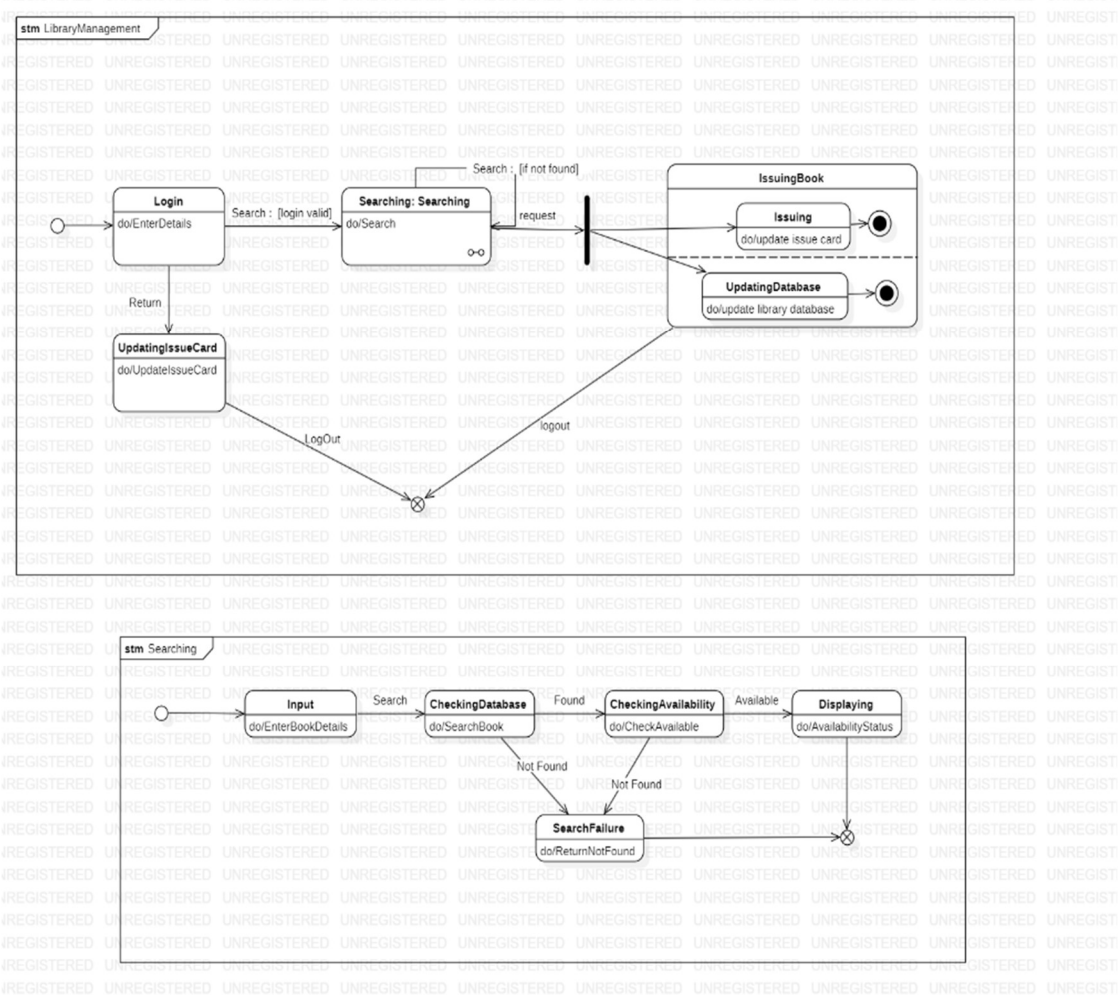


Fig 3.4.1

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 Use Case Diagram

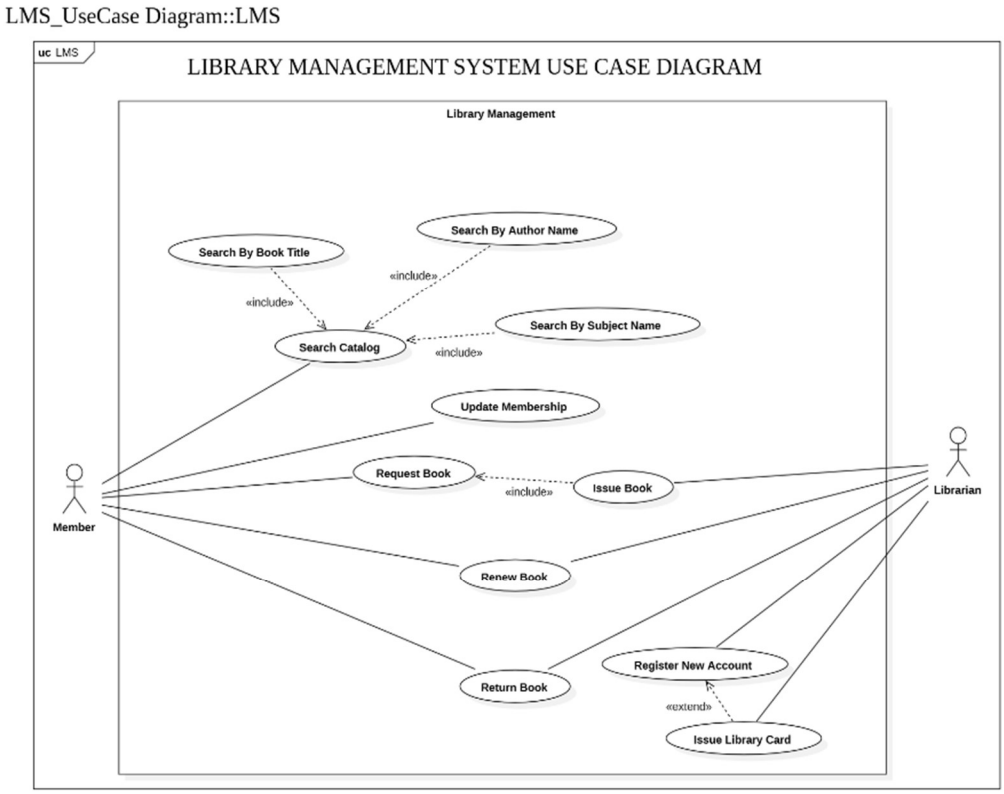


Fig 3.5.1

It represents a use case diagram for a Library Management System (LMS). It depicts the interactions between two actors, "Member" and "Librarian," and the system's functionalities represented as use cases (ovals). The "Member" actor can perform actions like searching the catalog (by book title, author name, or subject name), requesting a book, renewing a book, and returning a book. The "Librarian" actor is responsible for issuing books (which "includes" the "Request Book" use case), registering new accounts (which "extends" to "Issue Library Card"), and updating memberships. The "Search Catalog" use case is further detailed by "include" relationships with "Search By Book Title," "Search By Author Name," and "Search By Subject Name," indicating these are sub-functions of the broader search functionality.

Overall, the diagram illustrates the system's scope and the services offered to members and librarians.

3.6 Sequence Diagram

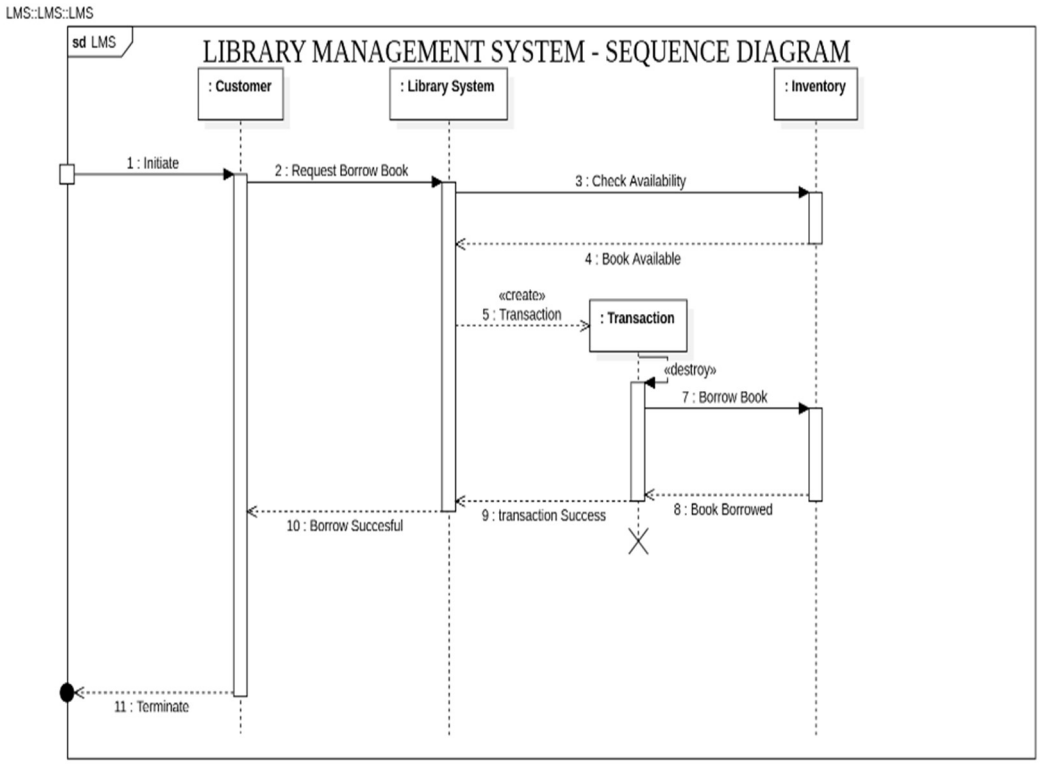


Fig 3.6.1

This is a sequence diagram for a Library Management System, specifically illustrating the "Borrow Book" use case. It shows the interactions between several objects or actors over time. The "Customer" initiates the process by requesting to borrow a book. The "Library System" then checks the availability of the book in the "Inventory." If the book is available, the system creates a "Transaction" object to record the borrowing details. The "Inventory" is then updated to reflect that the book is borrowed. The "Transaction" object is subsequently destroyed (likely after the borrowing is successfully recorded). Finally, the "Library System" informs the "Customer" that the borrow was successful, and the interaction terminates. The diagram emphasizes the chronological order of messages exchanged between the objects during the book borrowing process.

3.7 Activity Diagram

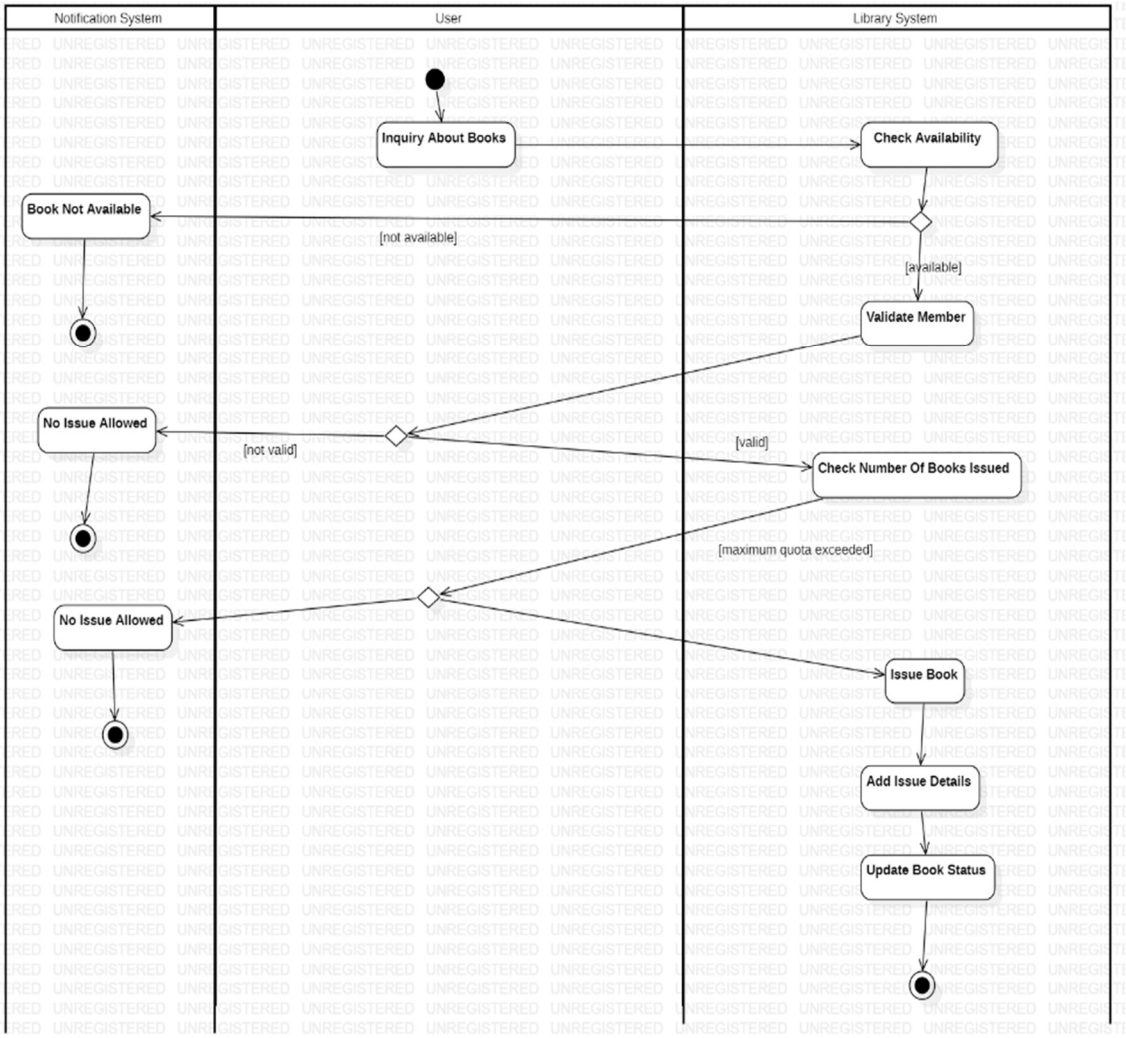


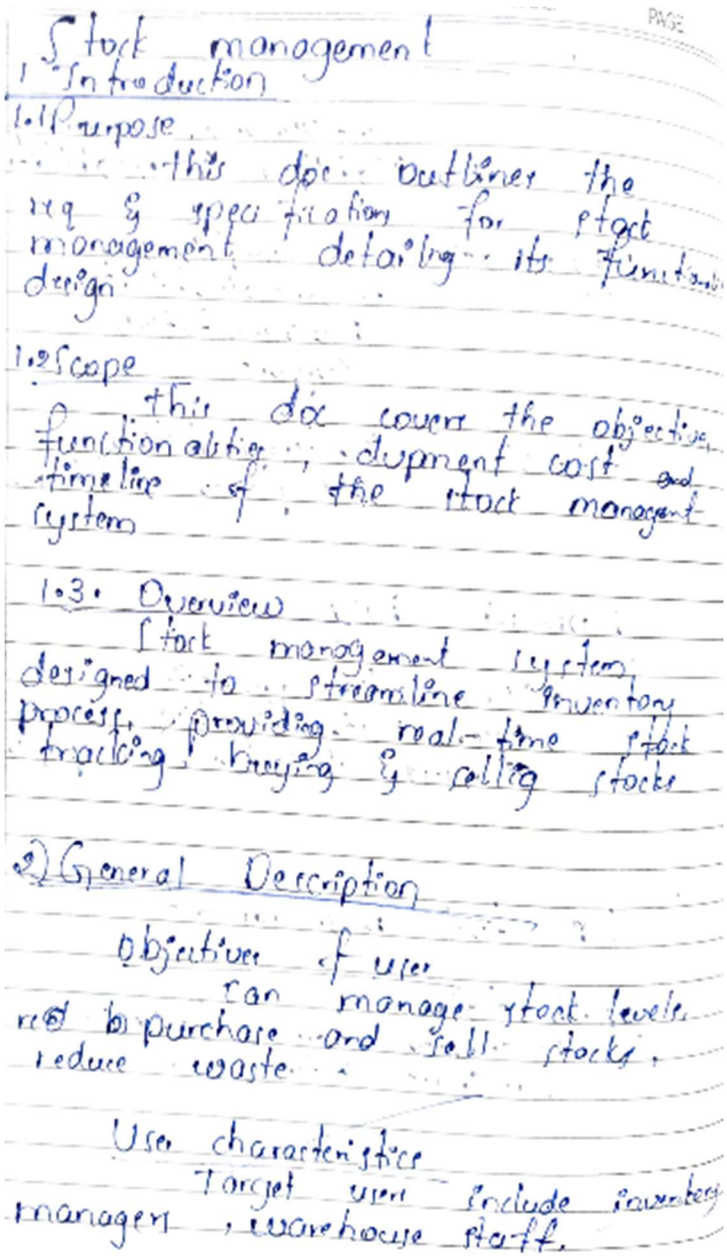
Fig 3.7.1

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. Stock Management System

4.1 Problem Statement: Managing stock levels manually in a business is inefficient, prone to human error, and can lead to overstocking or understocking issues. A well-organized system is required to monitor inventory, track stock movements, and ensure the availability of items to meet customer demands efficiently.

4.2 SRS-Software Requirements Specification



Stock management

1. Introduction

1.1 Purpose

this doc outliner the req & specifications for stock management detailing its functionality design

1.2 Scope

this doc covers the objectives functionality, dupment cost and timeline of the stock management system

1.3. Overview

Stock management system designed to streamline inventory process providing real-time stock tracking buying & selling stocks

2) General Description

objectives of user

can manage stock levels need to purchase and sell stocks, reduce waste.

User characteristics

Target user include inventory manager, warehouse staff.

Fig 4.2.1

Features and Benefits:

Key feature include real-time tracking, reporting and supplier management

3) Functional Req

* Inventory tracking
* Order processing
* Supplier management.
* Reporting
* User management.

4) Interface Req

System should support data exchanges through API's and with e-commerce platforms & accounting software

5) Performance Req

* Response time of system should be within 2s

* Memory usage

Must efficiently operate within 1GB of memory

* Error rate

Error rate should be low as possible

Fig 4.2.2

4.3 Class Diagram

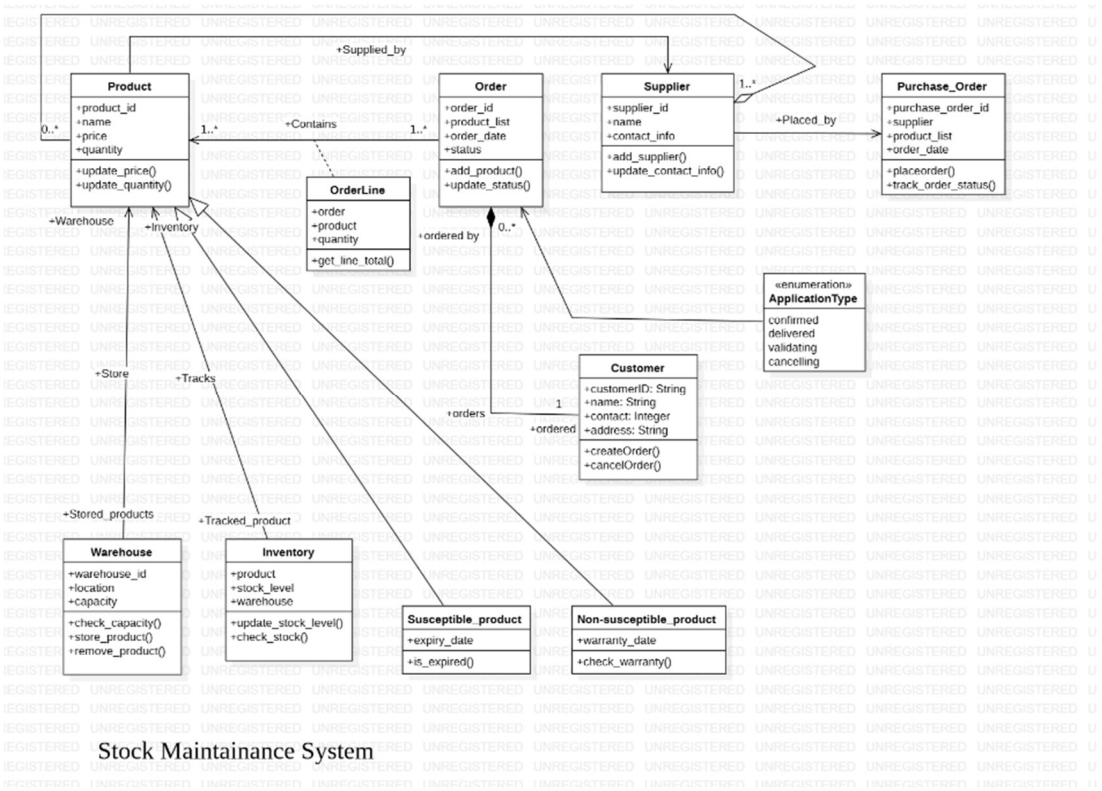


Fig 4.3.1

This represents a class diagram for a Stock Maintenance System. It models the relationships between various entities involved in managing stock, orders, and suppliers. Key entities include Product (with attributes like ID, name, price, and quantity, and operations to update price and quantity), Supplier (with name, contact information, and operations to add/update supplier details), Order (containing an order ID, product list, order date, and status, with operations to add products and update status), and Purchase_Order (linking suppliers to orders and tracking order status). An OrderLine class connects Order and Product, specifying the quantity of each product in an order. A Customer places orders and has attributes like ID, name, contact, and address. Warehouse stores products and has attributes for ID, location, and capacity, with operations to check capacity and manage stored products. The Inventory tracks the stock level of products in each warehouse. Finally, there are specialized product types: Susceptible_product (with an expiry date and an is_expired() operation) and Non-susceptible_product (with a warranty date and a check_warranty() operation), both inheriting from Product. The diagram shows the associations between these

classes, illustrating the structure of the system and how different entities relate to each other in managing stock and orders.

4.4 State Diagram

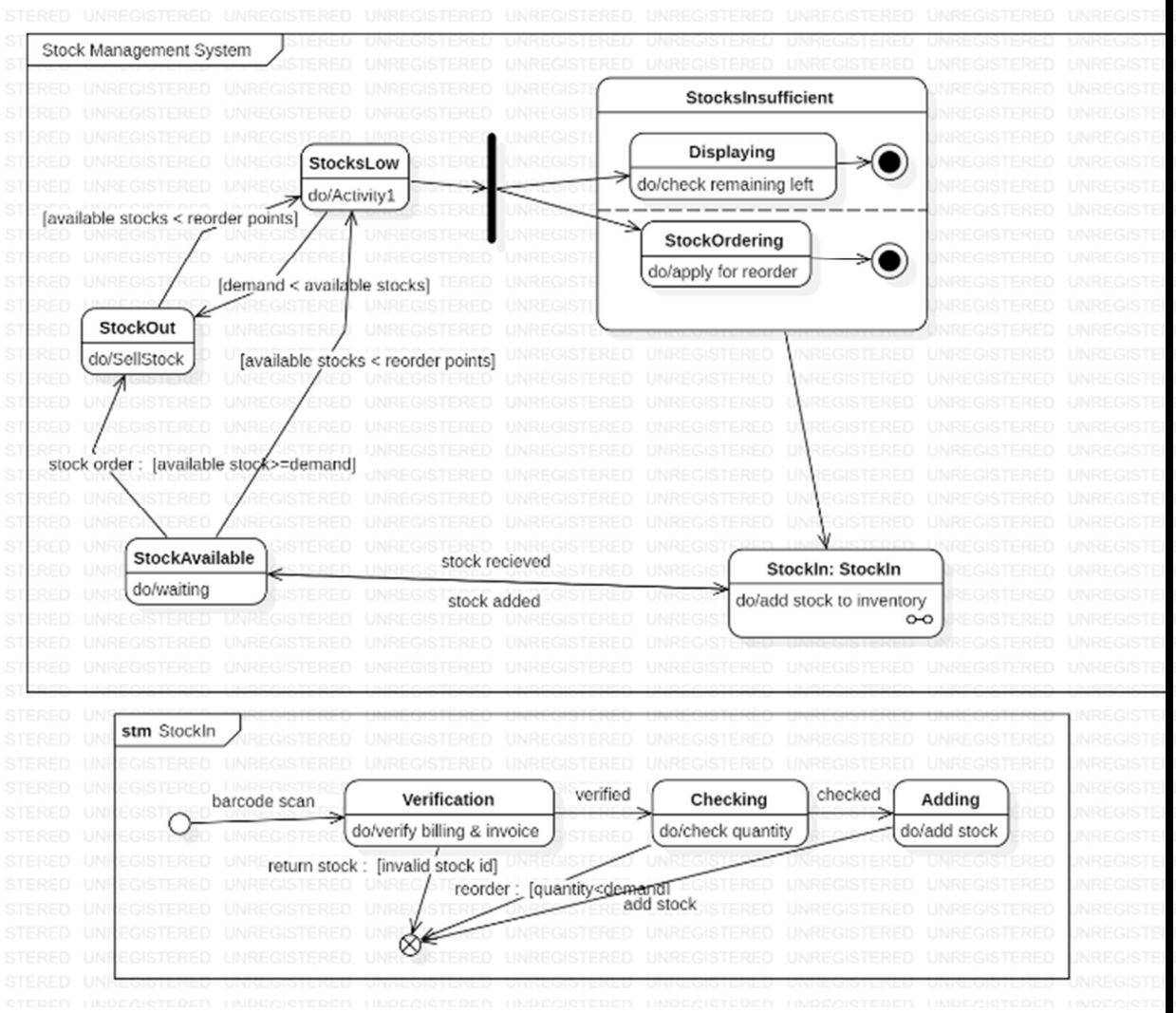


Fig 4.4.1

This state machine diagram represents the workflow of a stock maintenance system, detailing the various states and transitions involved in stock management. The process begins in an idle state, where the system waits for user login. Once authenticated, users can check the stock status, which involves fetching data from the stock database and displaying current stock levels and availability. After reviewing the stock, users can place orders by selecting the required items. Payment processing is handled through the payment system, with the outcome determining the next steps: a successful payment

transitions to the successful state, while a failed payment leads to a waiting state where the system waits for the stock to arrive after retrying the payment. In the successful state, the system confirms the order by notifying the warehouse and updating internal stock records to reflect the new order. Finally, the system generates and prints a stock receipt for the ordered items, marking the end of the process. This diagram captures the seamless interaction between stock checking, ordering, payment, and stock record management

4.5 Use Case Diagram

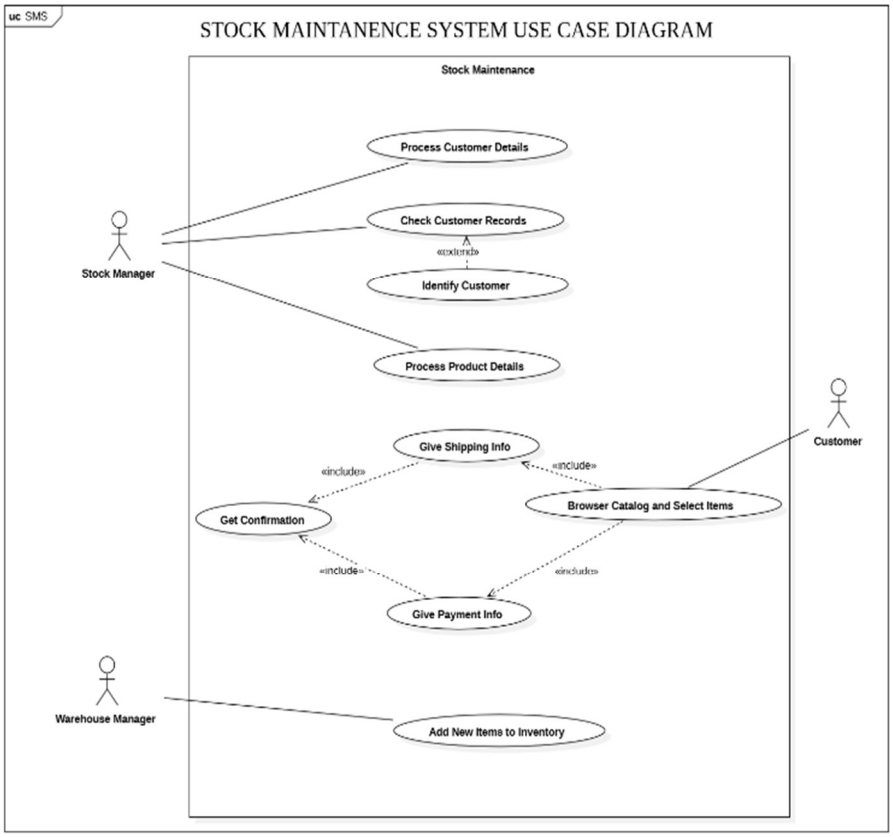


Fig 4.5.1

This use case diagram illustrates the interactions within a stock maintenance system, focusing on three main actors: the stock manager, customer, and warehouse manager. The stock manager oversees customer management by processing customer details, checking customer records, and identifying customers. They also handle product details and coordinate with customers by providing shipping information, receiving payment details, and confirming transactions. Customers actively participate by browsing the catalog, selecting items, providing shipping and payment information, and receiving

confirmation. Meanwhile, the warehouse manager is responsible for adding new items to inventory, ensuring stock levels are maintained. The diagram highlights the collaboration between these actors and their roles in the stock management process, ensuring smooth and efficient operations.

4.6 Sequence Diagram

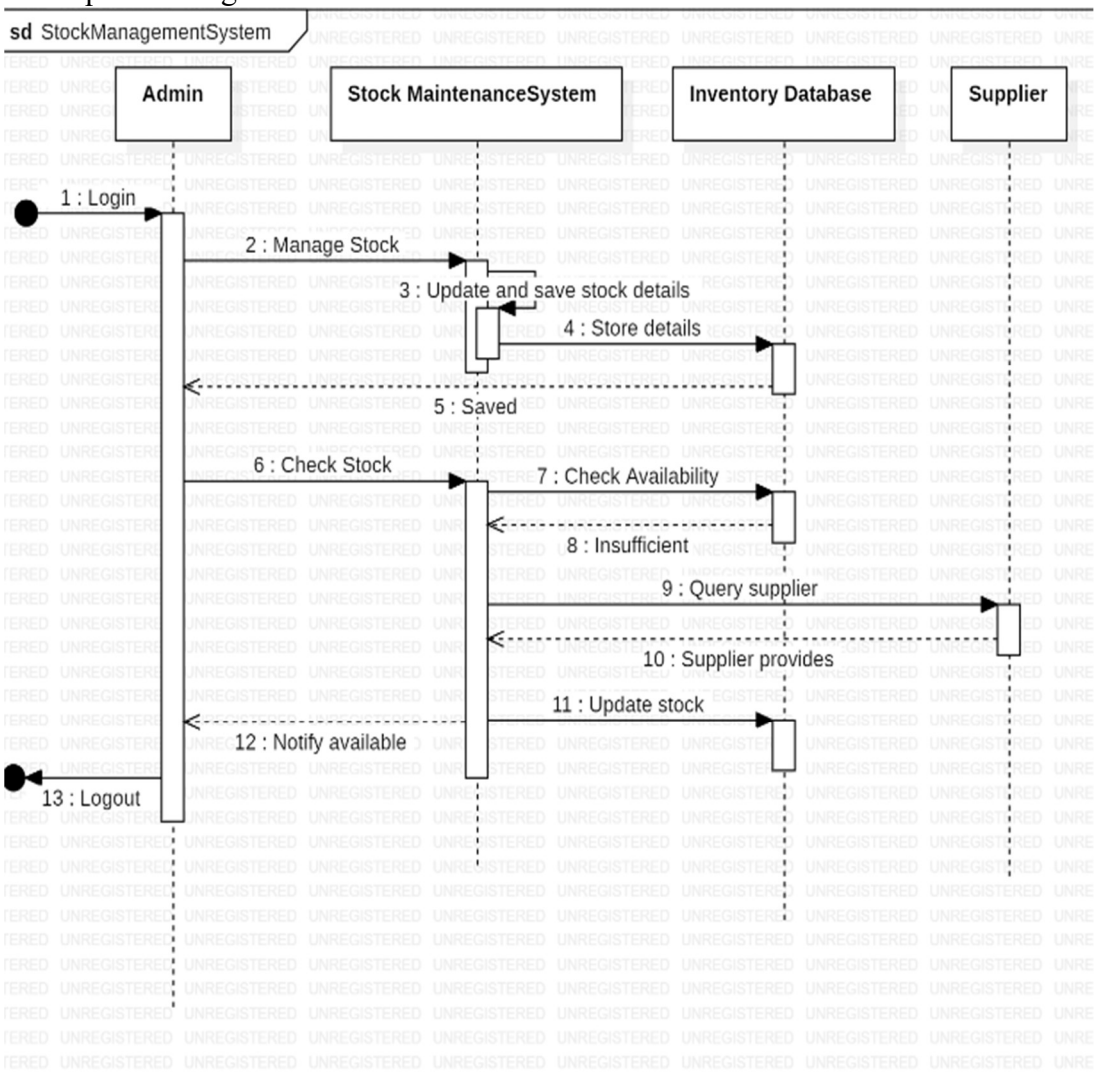


Fig 4.6.1

This sequence diagram illustrates the process flow in a stock management system involving three main entities: the customer, supplier, and warehouse. The interaction begins with the customer initiating a product inquiry, prompting the supplier to check product availability with the warehouse. The warehouse confirms availability to the supplier, who then shares the details with the customer. After reviewing the details, the customer places an order with the supplier. The supplier subsequently places the order with the warehouse, which updates stock records, confirms the order, and deducts the

ordered quantity. Once the order is placed, the customer completes the payment, and the process proceeds to delivery. This diagram effectively captures the sequential actions and interactions ensuring smooth stock management.

4.7 Activity Diagram

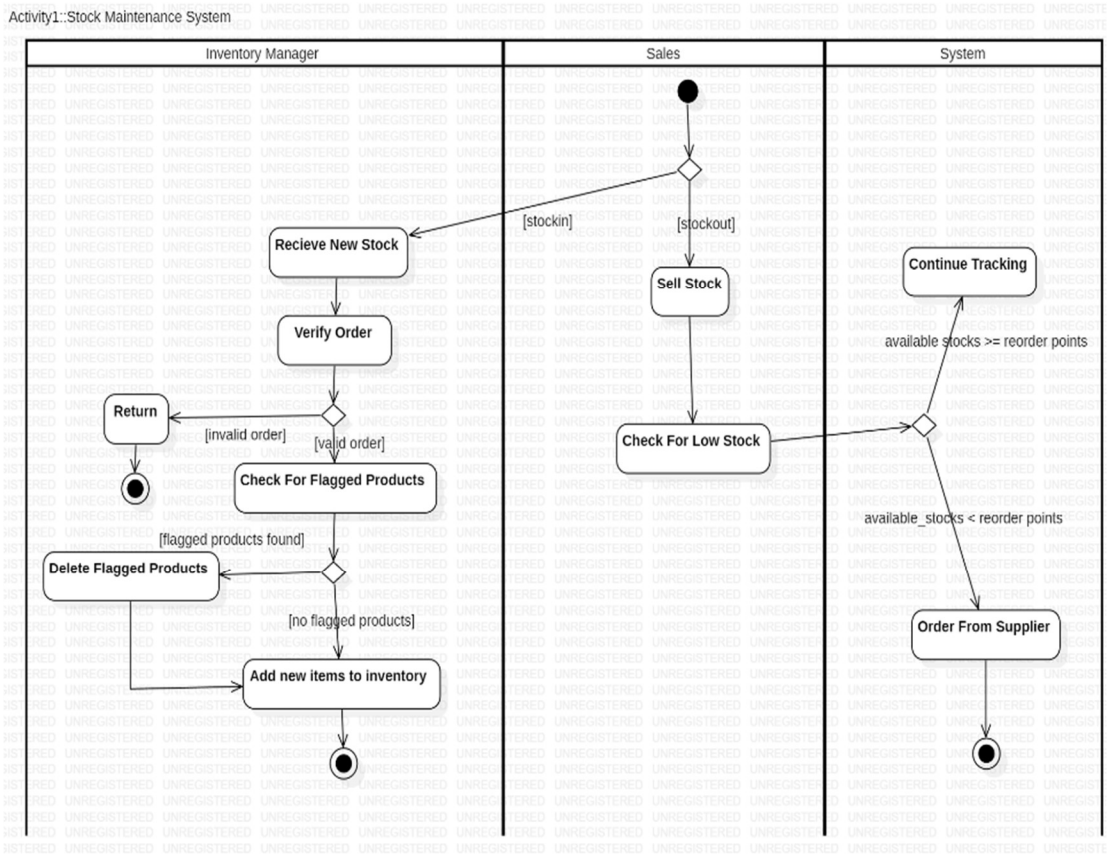


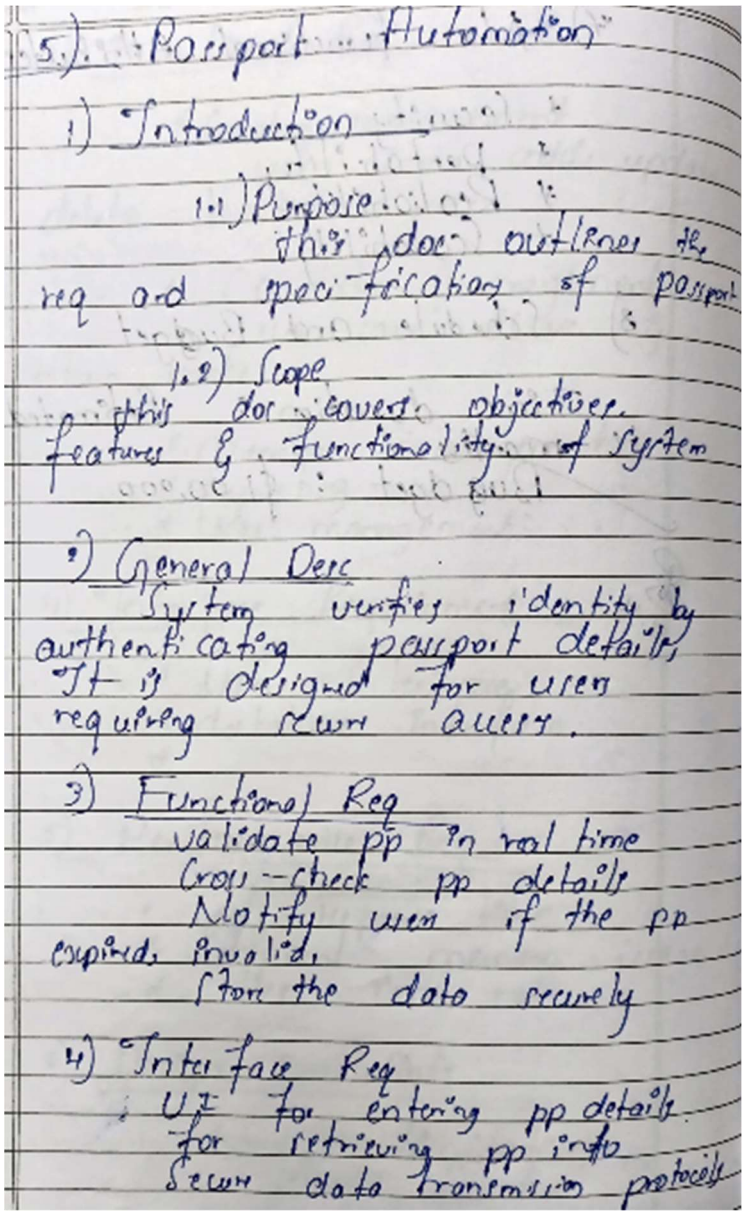
Fig 4.7.1

The diagram showcases a stock management process in a system with four key components: Admin, Inventory Database, Stock Maintenance System, and Supplier. It begins with the Admin logging into the system to manage stock. This triggers the Stock Maintenance System to check the stock levels in the Inventory Database. If the stock is sufficient, the system updates and stores the relevant details in the database, ensuring the Admin has the latest information. However, if the stock is found to be insufficient, the Stock Maintenance System communicates with the Supplier. This involves querying the Supplier to check availability and waiting for the supplies to be provided through the supplier's network. Once the Supplier fulfills the request and the stock is delivered, the system updates the inventory and notifies the Admin that the required stock is now available. This automated process ensures seamless coordination between different components for efficient inventory tracking and timely replenishment.

5. Passport Management System

5.1 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.

5.2 SRS-Software Requirements Specification


5.) Passport Automation

1) Introduction

1.1) Purpose
this doc outlines the req and specifications of passport

1.2) Scope
this doc covers objectives features & functionality of system

2) General Desc
System verifies identity by authenticating passport details. It is designed for users requiring secure access.

3) Functional Req
validate pp in real time
Cross-check pp details
Notify user if the pp expired, invalid,
store the data securely

4) Interface Req
UI for entering pp details
for retrieving pp info
Secure data transmission protocols

Fig 5.2.1

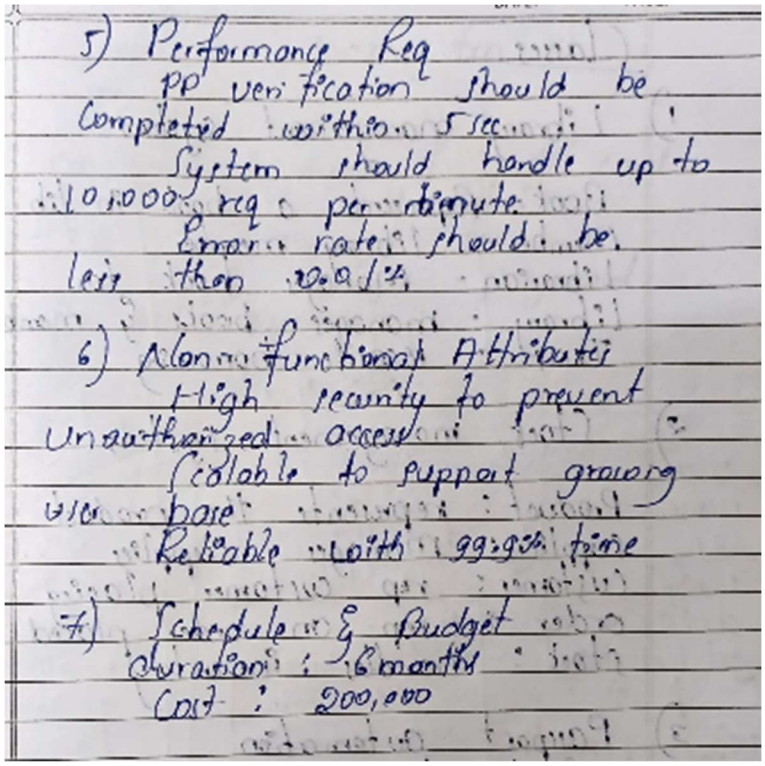


Fig 5.2.2

5.3 Class Diagram

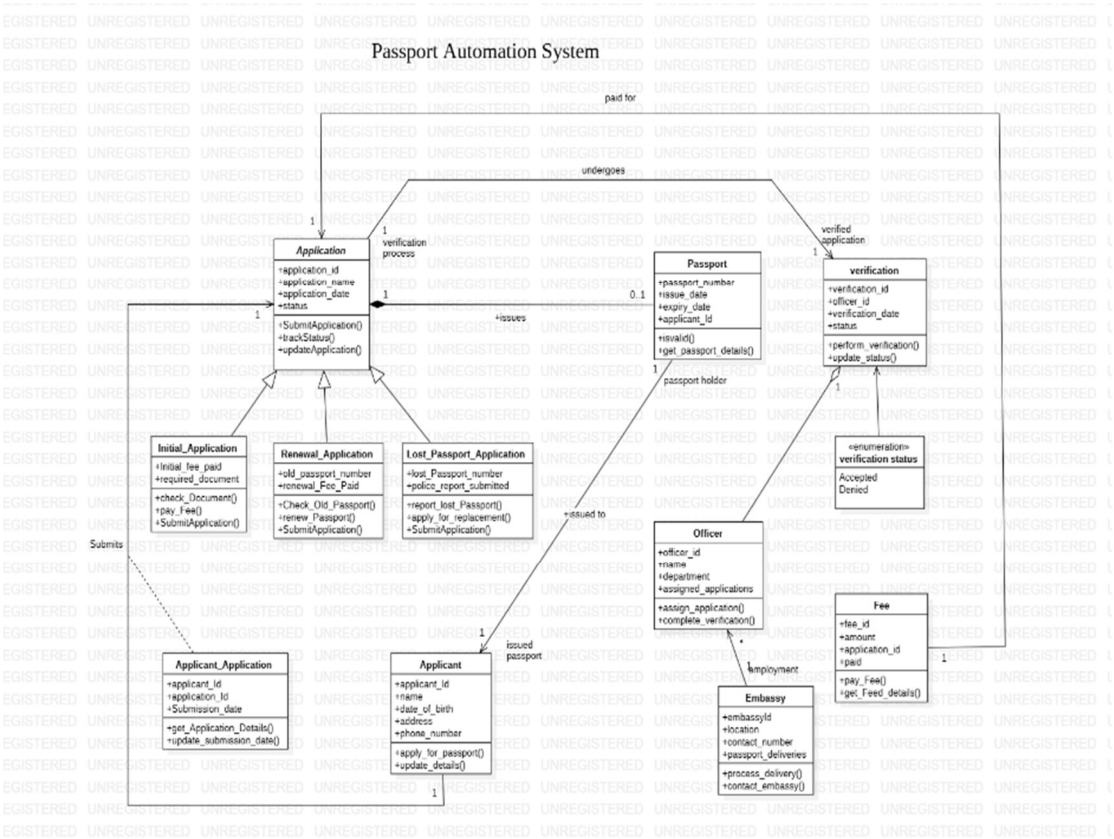


Fig 5.3.1

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.

5.4 State Diagram

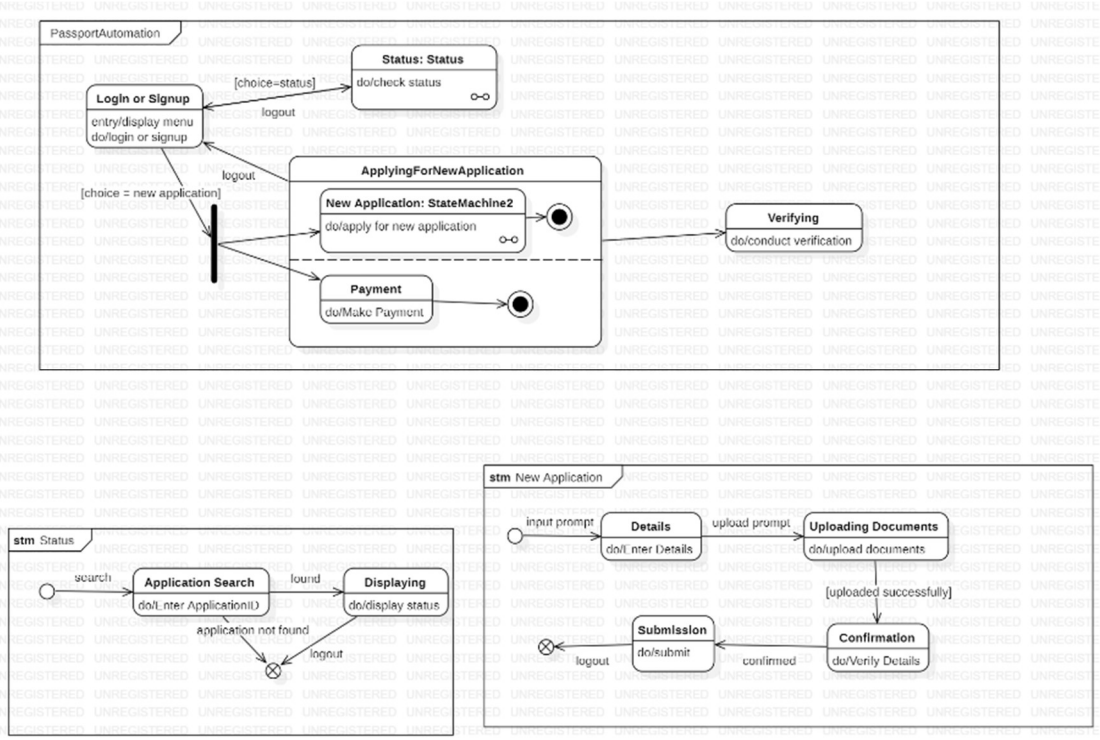


Fig 5.4.1

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.

5.5 Use Case Diagram

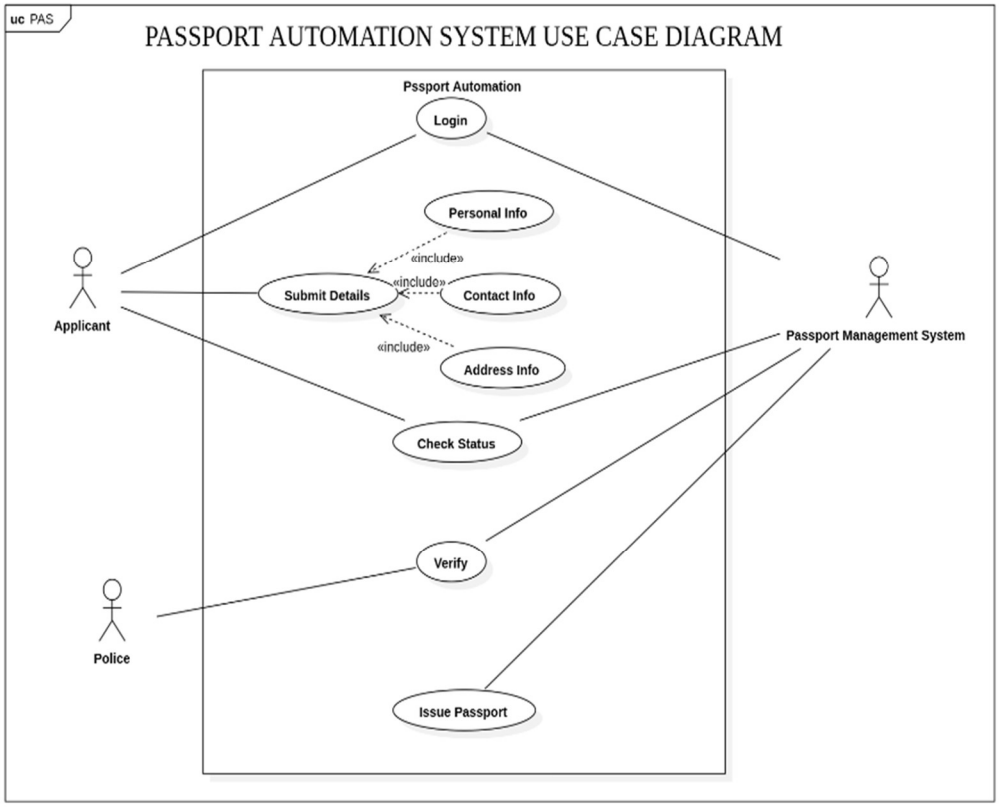


Fig 5.5.1

This use case diagram for a Passport Automation System. It illustrates the interactions between three actors: the Applicant, the Police, and the Passport Management System. The Applicant starts by logging into the system and proceeds to submit details, which include personal information, contact information, and address information. These are linked as separate use cases through the “include” relationship, signifying that they are

essential steps in the process. The Applicant can also check the status of their application. The Police are involved in verifying the submitted details, a process connected to both the Applicant and the Passport Management System. Once the verification is completed, the system facilitates the issuance of the passport. The diagram effectively highlights the functional flow and dependency of activities within the automation system.

5.6 Sequence Diagram

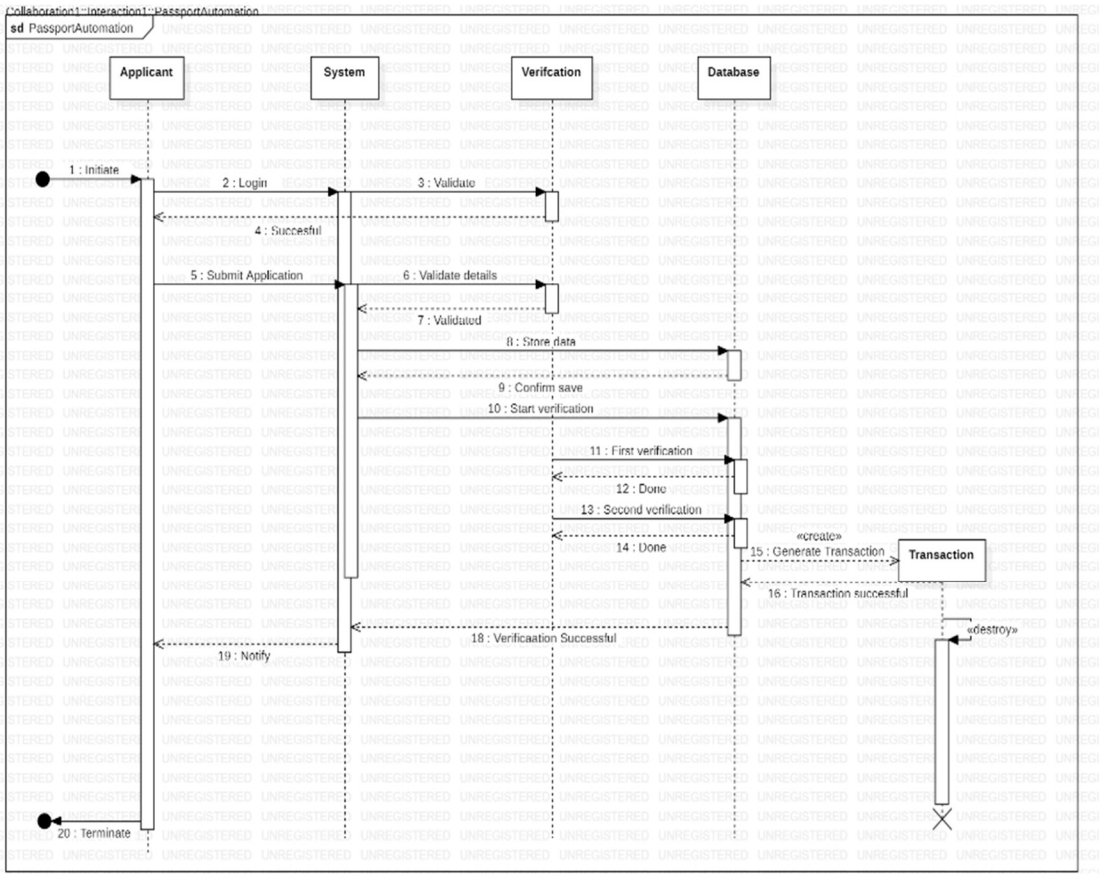


Fig 5.6.1

The sequence diagram illustrates the process flow in a Passport Automation System, involving four main entities: the Applicant, Passport Administration System, Officer, and System Database. The process begins with the Applicant accessing the portal (Step 1), which then displays the portal interface (Step 2). The Applicant initializes the application (Step 3), uploads the required documents (Step 5), and provides fee payment information (Step 6). The system stores the Applicant's details in the database

(Step 4) and updates the application database after submission (Step 8). Once the Applicant submits the application (Step 7), they receive an acknowledgment of submission (Step 9). The application is then assigned to an Officer for review (Step 10), and the Officer verifies the information, updating the application status (Step 11). Finally, the Applicant receives updates regarding the status of their application (Step 12). This diagram effectively captures the sequential interactions and data flow between the entities involved.

5.7 Activity Diagram

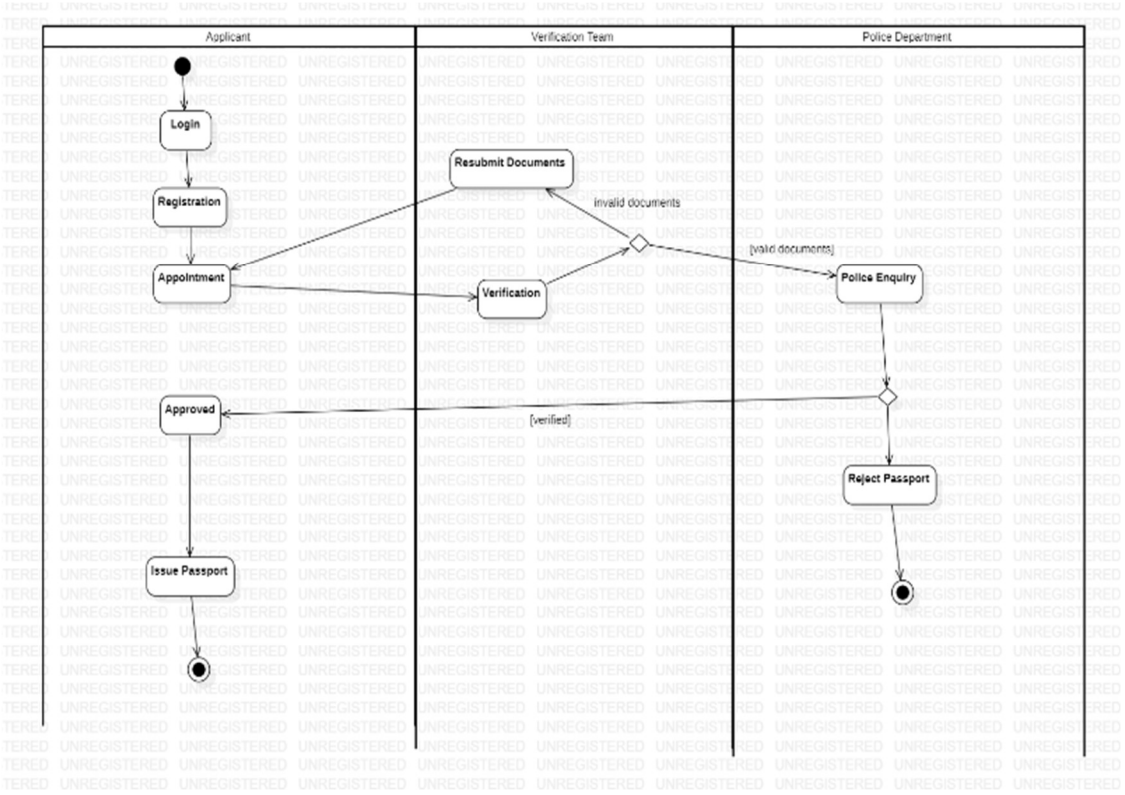


Fig 5.7.1

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.