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:

D V Vedith Varma

1BM22CS339

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

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Analysis and Design(23CS5PCOOM) laboratory has been carried out by D V Vedith Varma (1BM22CS339) during the 5th Semester Oct24-Jan2025.

Signature of the Faculty Incharge:

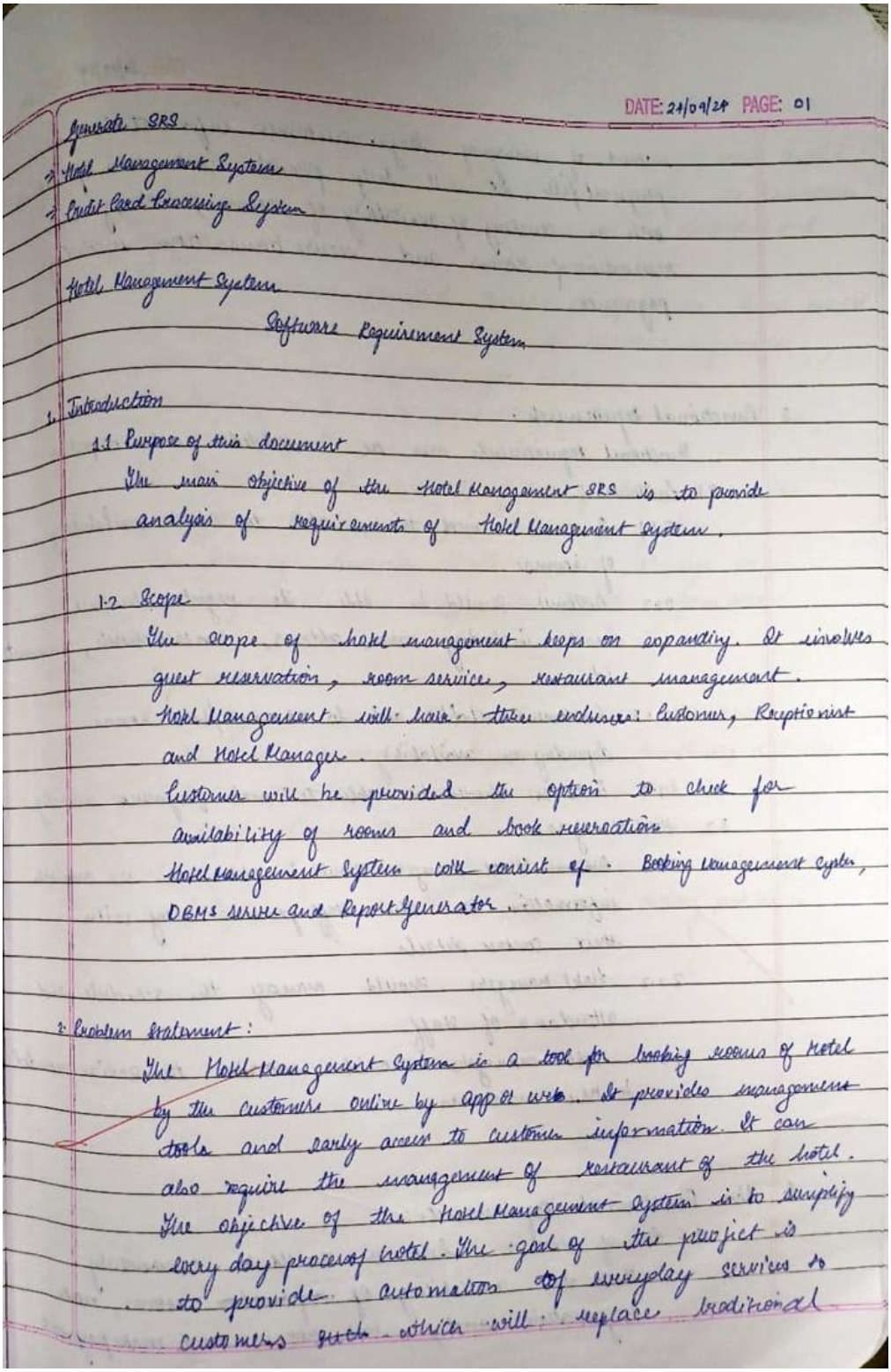
NAME OF THE FACULTY: Prof. Anusha S

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

Table of Contents

1. Hotel Management System

1.1 SRS:



DATE: 21/09/24 PAGE: 01

Generate SRS
→ Hotel Management System
→ Credit Card Processing System

Hotel Management System

Software Requirement System

1. Introduction

1.1 Purpose of this document

The main objective of the Hotel Management SRS is to provide analysis of requirements of Hotel Management system.

1.2 Scope

The scope of hotel management keeps on expanding. It involves guest reservation, room service, restaurant management. Hotel Management will have three endusers: Customer, Receptionist and Hotel Manager.

Customer will be provided the option to check for availability of rooms and book reservation

Hotel Management System will consist of: Booking Management system, DBMS server and Report Generator.

2. Problem Statement:

The Hotel Management System is a tool for booking rooms of Hotel by the customers online by app or web. It provides management tools and early access to customer information. It can also require the management of restaurant of the hotel. The objective of the Hotel Management system is to simplify every day process of hotel. The goal of the project is to provide automation of everyday services to customers such which will replace traditional

method of managing large customer information in physical file. It will help provide automation of services such as checking of availability of rooms, booking and reservation of rooms and secure transaction involving payments.

3. Functional Requirements:

Functional Requirements can be divided with respect to:

3.1 Customers:

3.1.1 Customer should to be able to check availability of rooms

3.1.2 Customer should be able to register his/her details including name, address, contact details, government id

3.1.3 Customer should be able to reserve/book rooms depending on availability

3.1.4 Customer should be able to make payment securely

3.2 Hotel Managers:

3.2.1 ~~Customer~~ Hotel Managers should be able to see available information on the staff present along with their contact details

3.2.2 Hotel managers should manage the schedule and attendance of staff

3.2.3 Hotel managers should monitor the services provided to the customers.

4. Non-Functional Requirements:

4.1 Ease of access: Customer should be conveniently check the availability of ~~information~~ rooms, make registration and, book rooms and make payment

4.2 Reliable : The information regarding availability of rooms should be consistent and there should be no conflict in reservation of rooms. Payments and transactions should be recorded and reflected.

4.3 Speed: The services provided should be fast as. There should be no delay in validation of customer registration, reservation, and payment for room.

4.4 Accuracy : The data provided to customers should be accurate in all cases.

4.5 Robustness : In case of failure of transaction the system should maintain consistency of data.

4.6 Security : Customer data should always be confidential especially the contact information

5. Domain Requirements :

5.1 Security : All external communication between database server and client must be encrypted. Payment process should use HTTPS over Secure protocol to secure payment transaction

5.2 Safety : Database should be backed up every hour

5.3 Capacity Requirement : Not more 10,000 members to be registered System must handle at least 20 transaction during peak hours

1.2 Class Diagram:

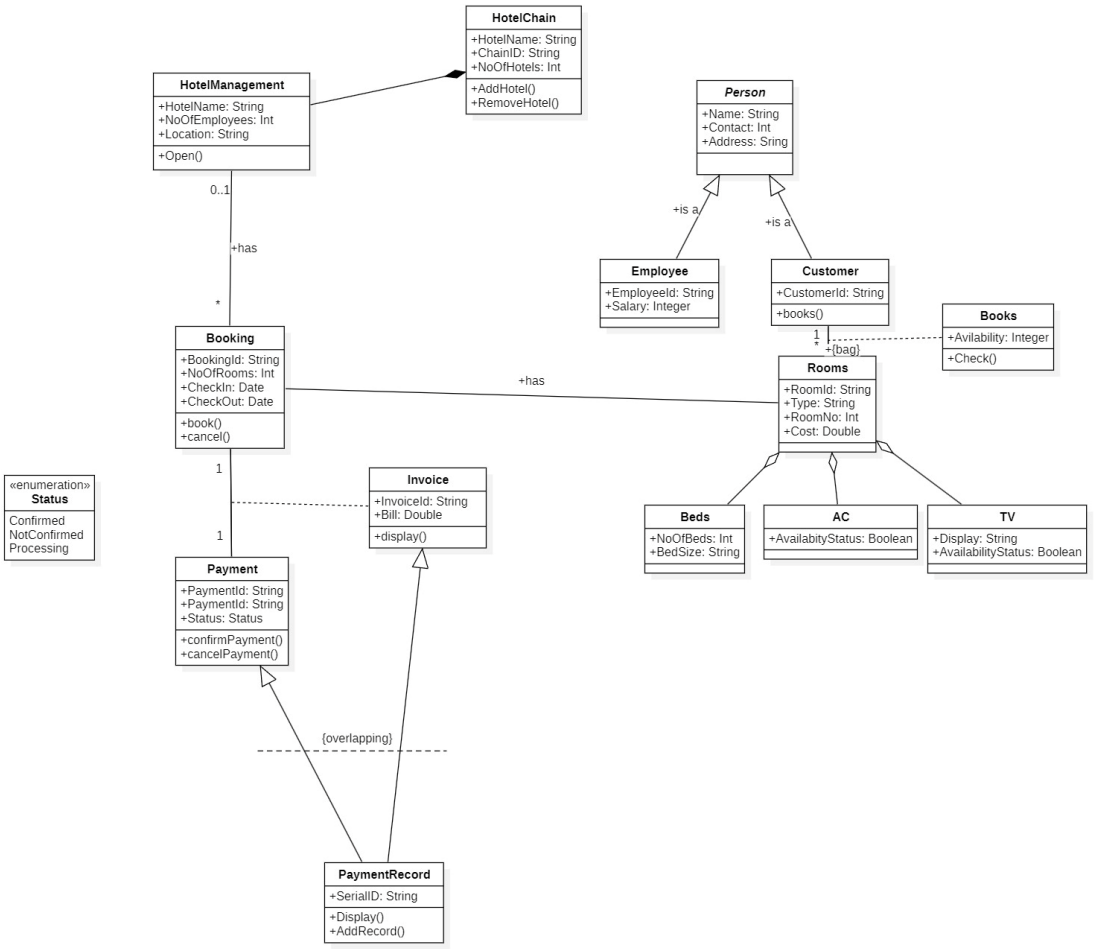


Fig 1.1: Class Diagram

Description:

The class diagram illustrates the structure of the system, highlighting key entities and their relationships.

- **Classes:**
    - **Booking Class**: Manages details of reservations such as booking ID, dates, room type, and status.

- **PersonAccount Class**: Abstract class serving as a base for employees and customers.
    - **Customer Class**: Inherits from PersonAccount and includes attributes like customer ID, name, and contact details.
    - **Employee Class**: Inherits from PersonAccount and includes attributes like employee ID, role, and shift details.
- **Room Class**: Represents the rooms in the hotel, with attributes such as room number, type, availability status, and price.
- **Payment Class**: Manages payment details, including payment ID, method, status, and amount.

- **Relationships:**
    - The **Booking Class** is associated with both **Room Class** and **Customer Class**, signifying that customers make bookings for rooms.
    - The **Payment Class** is related to the **Booking Class** to process transactions for bookings.
    - **Employee Class** is involved in managing bookings and customer queries.

1.3 State Diagram:

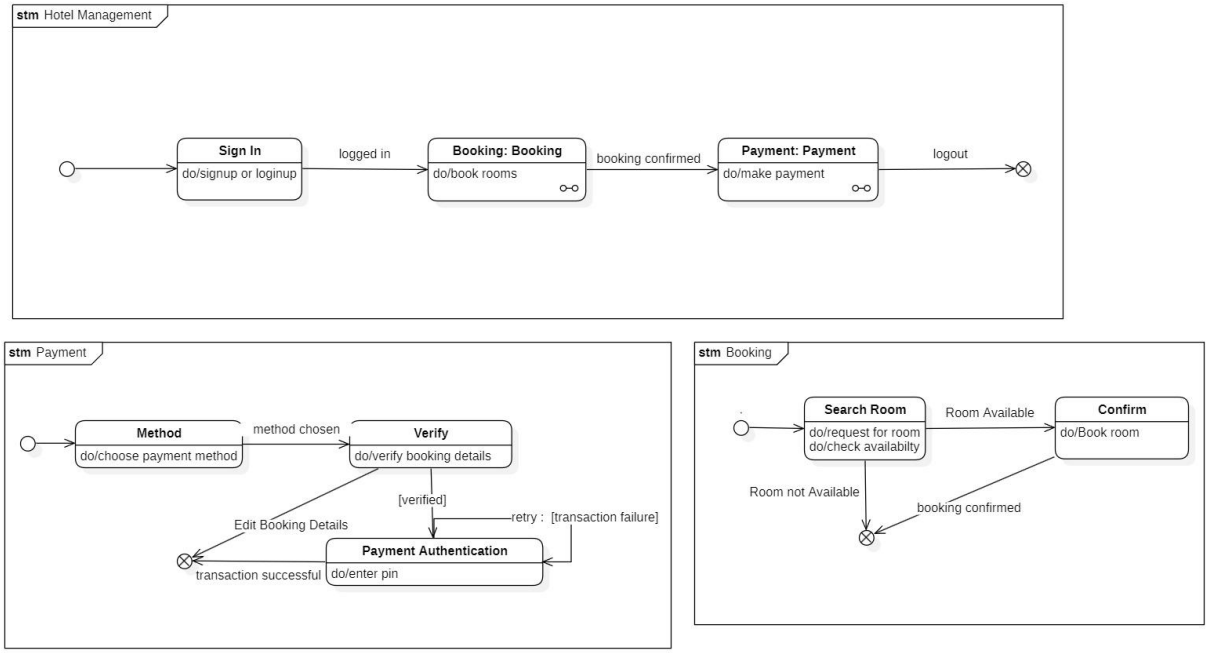


Fig 1.2: State Model Diagram

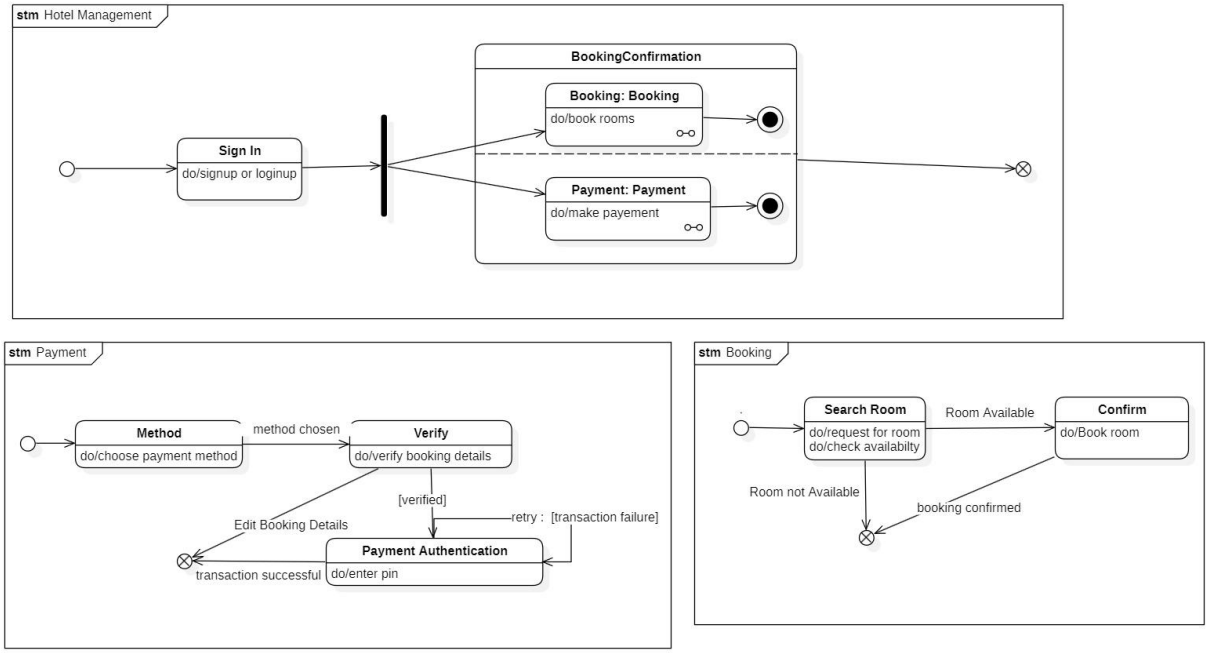


Fig 1.2: Advanced State Model Diagram

Description:

The state diagram shows the various states and transitions involved in the hotel management system.

- **States:**
    - **Sign-In State**: Represents when a user logs into the system (either customer or employee).
    - **Booking State**: Manages the reservation process, including selecting room type, dates, and confirming availability. Sub-states include:
        - Room Selection
        - Availability Check
    - **Payment State**: Handles payment processing for bookings. Sub-states include:
        - Payment Method Selection
        - Payment Confirmation
    - **Concurrent States:**
        - Booking and Payment processes are concurrent, allowing simultaneous management of reservations and transactions.

1.4 Use Case Diagram:

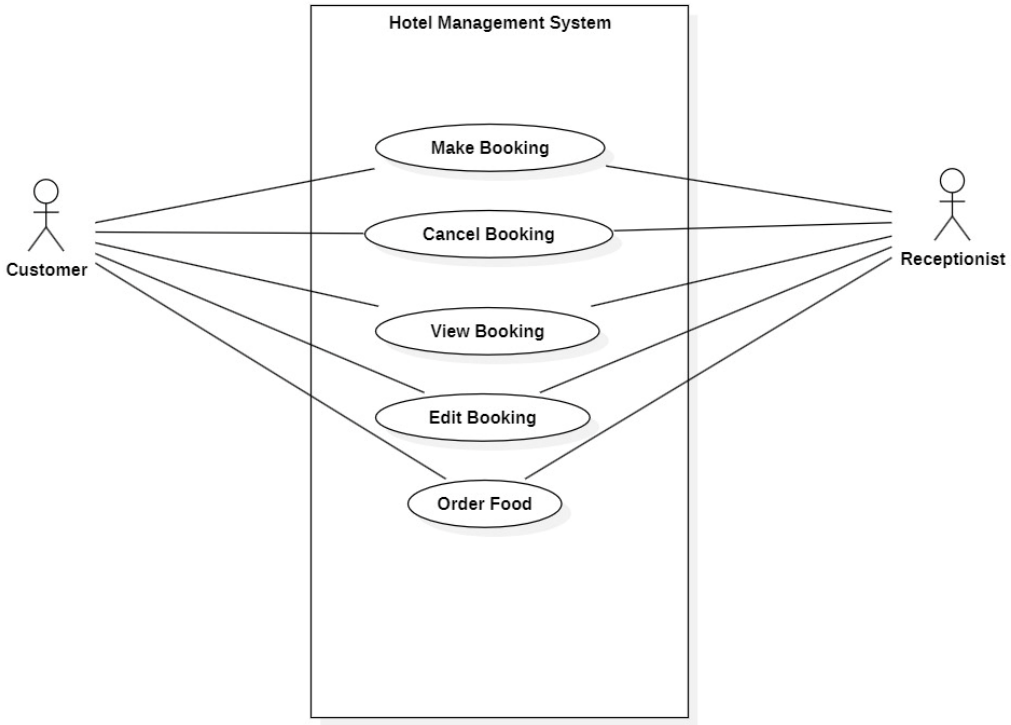


Fig 1.3 : Use Case Diagram

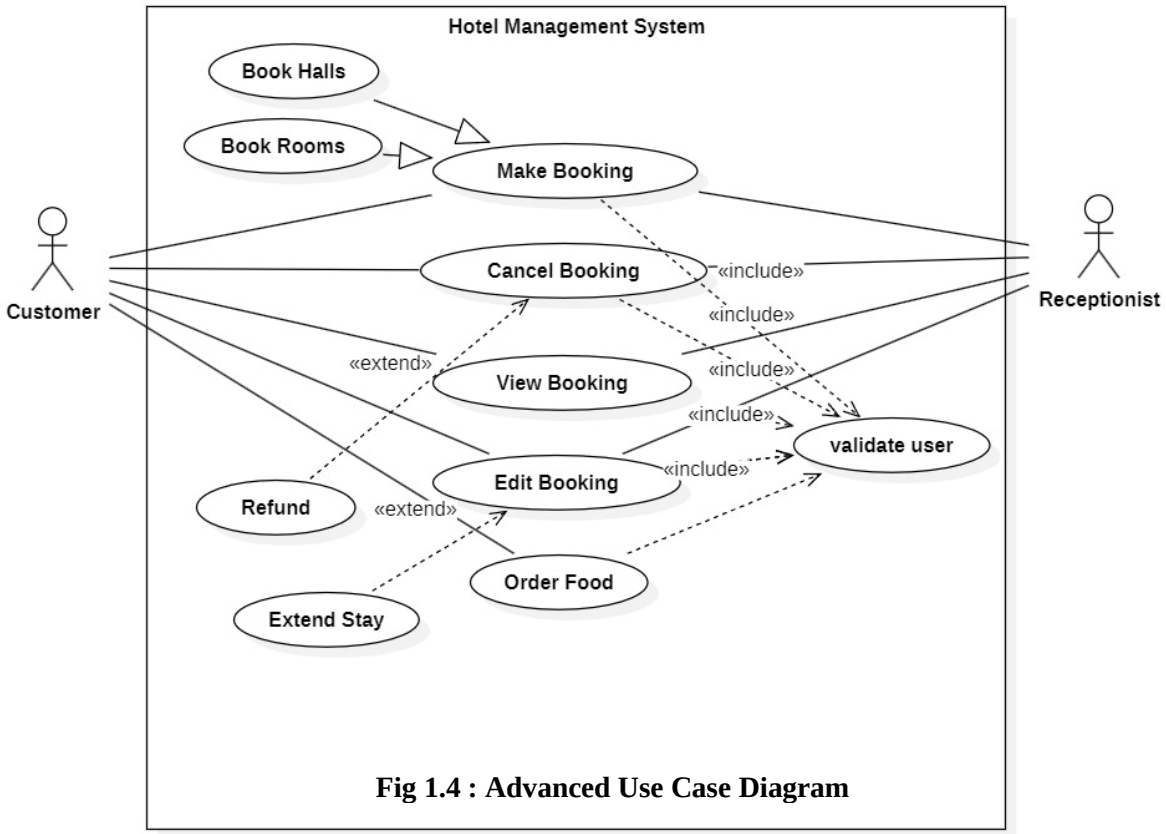


Fig 1.4 : Advanced Use Case Diagram

Description:

The use case diagram defines interactions between actors and system functionalities.

- **Actors:**
    - **Customer**: Can make a booking, cancel a booking, view existing bookings, edit bookings, and order food.
    - **Receptionist**: Can assist customers by managing bookings, updating information, and processing food orders.
- **Use Cases:**
    - **Make Booking**: Allows a customer to reserve a room.
    - **Cancel Booking**: Enables cancellation of an existing booking.
    - **View Booking**: Provides booking details to the customer or receptionist.
    - **Edit Booking**: Allows modifications to an existing booking.
    - **Order Food**: Lets the customer request room service or meals during their stay.

1.5 Sequence Diagram:

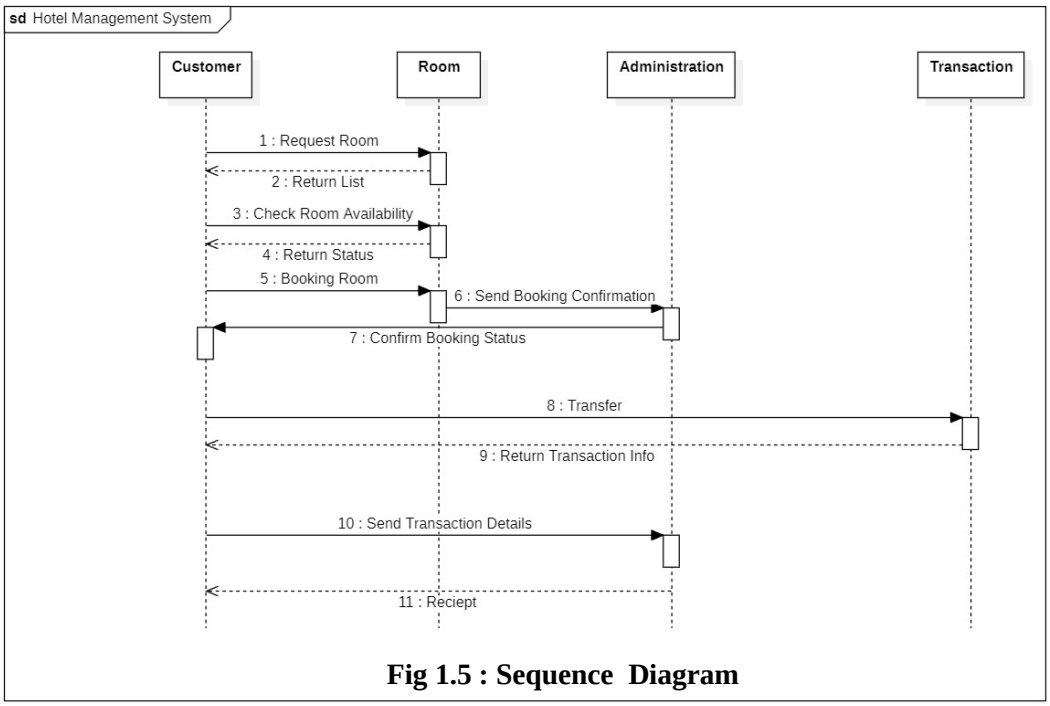


Fig 1.5 : Sequence Diagram

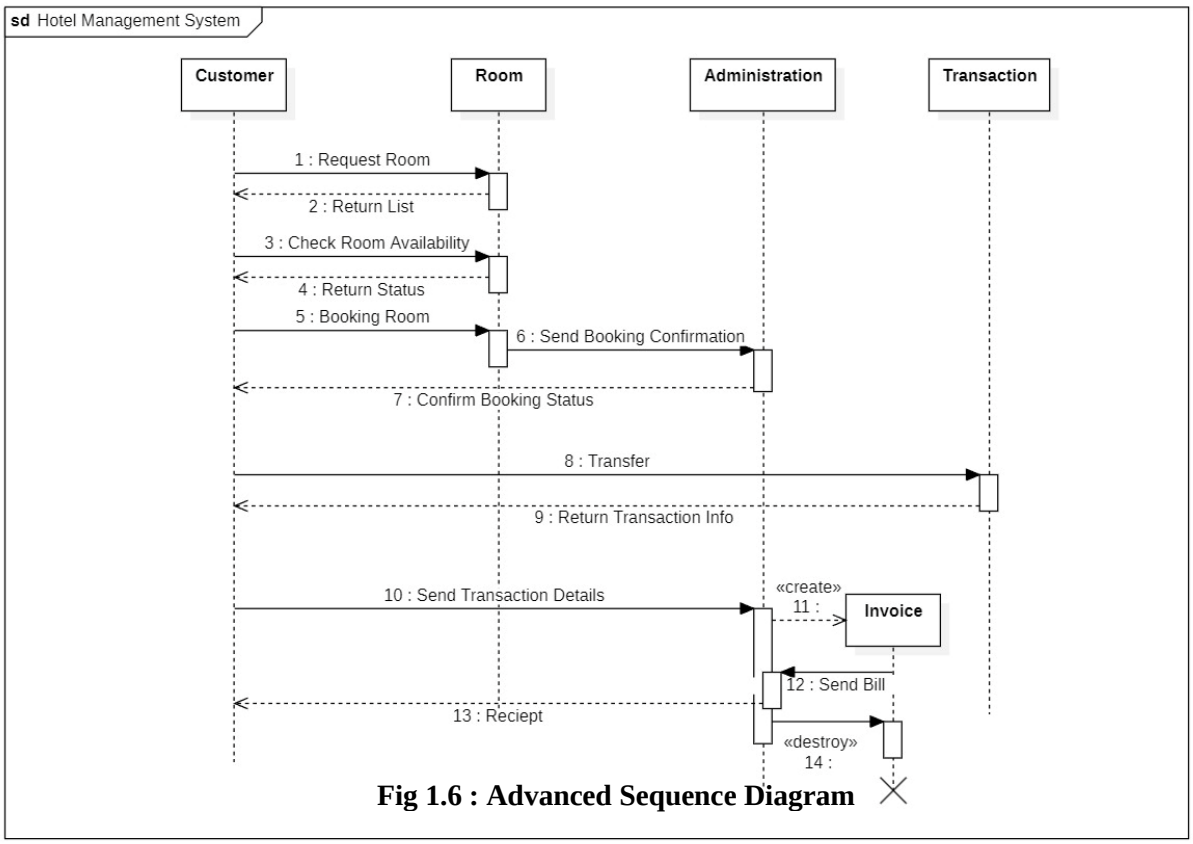


Fig 1.6 : Advanced Sequence Diagram

Description:

The sequence diagram details the interactions between objects to perform a booking operation.

- **Objects:**
    - **Customer**: Initiates the booking process.
    - **Room**: Provides room availability and details.
    - **Administration**: Validates the booking and room availability.
    - **Transaction**: Processes the payment for the booking.
    - **Invoice (Transient Object)**: Created after payment confirmation, representing the receipt of the transaction.
- **Flow:**

1. Customer requests a room booking.
2. System checks room availability via the Room object.
3. Administration validates the details.
4. Payment is processed via the Transaction object.
5. Invoice is generated and sent to the Customer.

1.6 Activity Diagram:

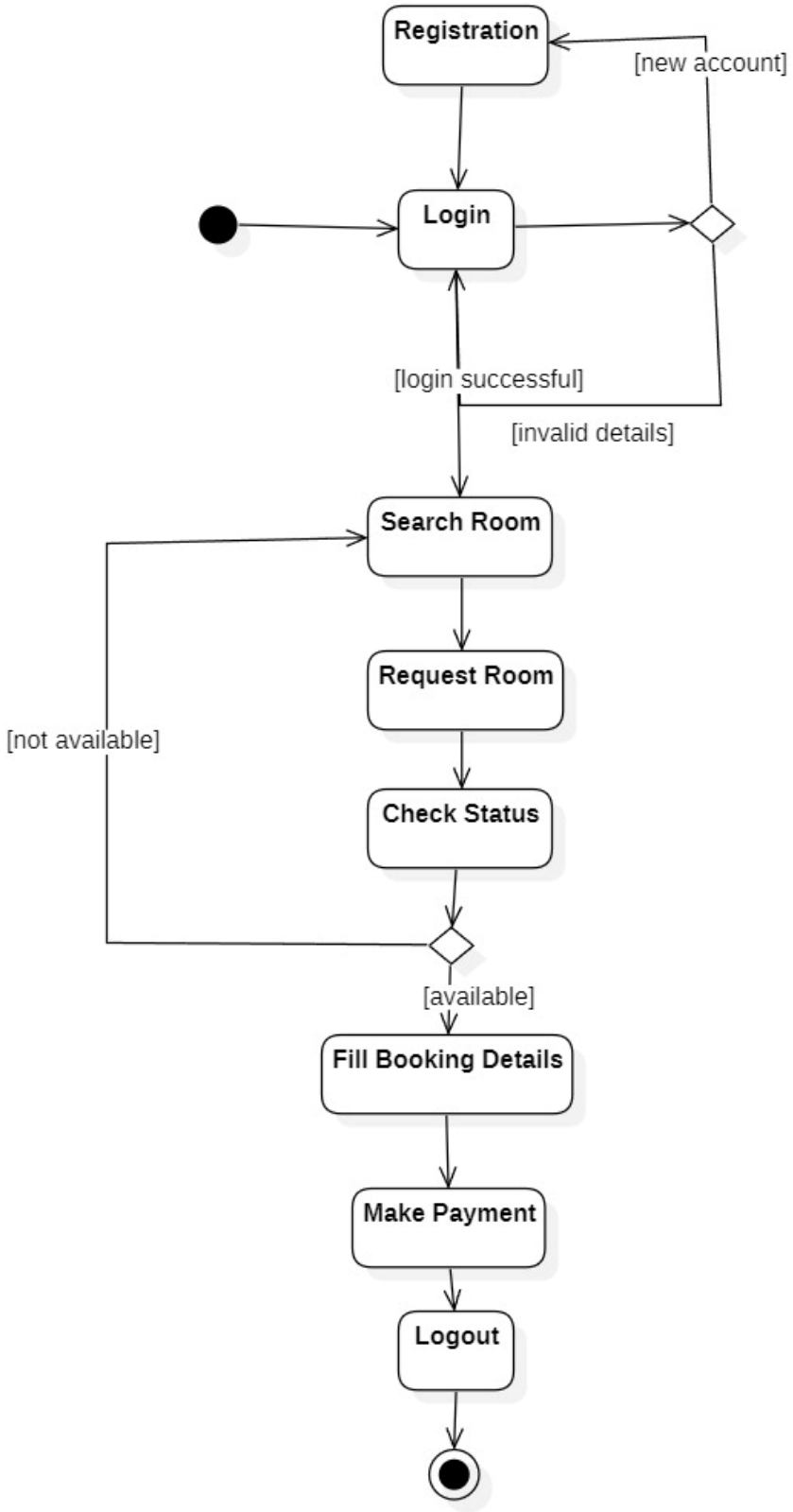


Fig 1.7 : Activity Diagram

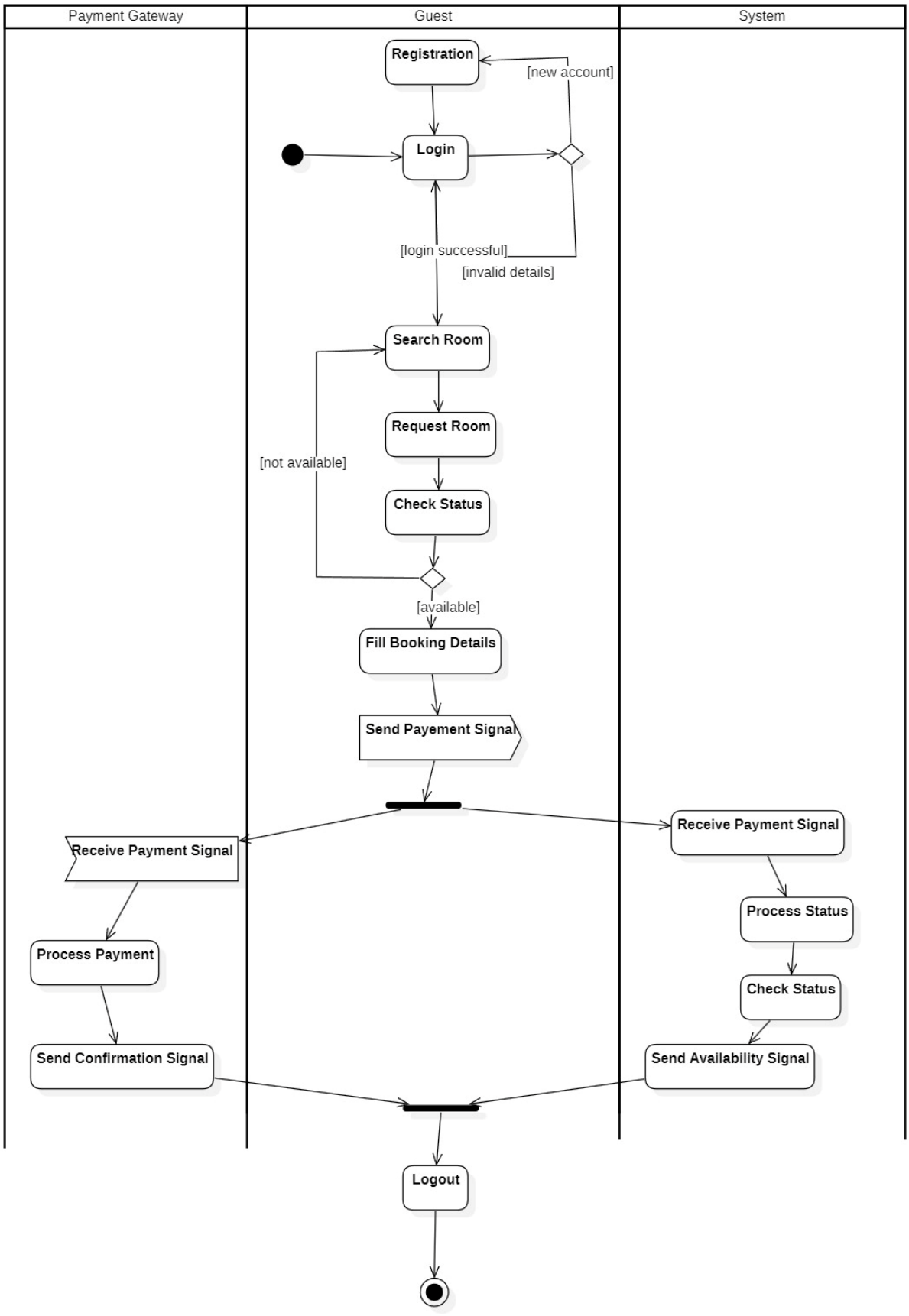


Fig 1.8 : Advanced Activity Diagram

Description:

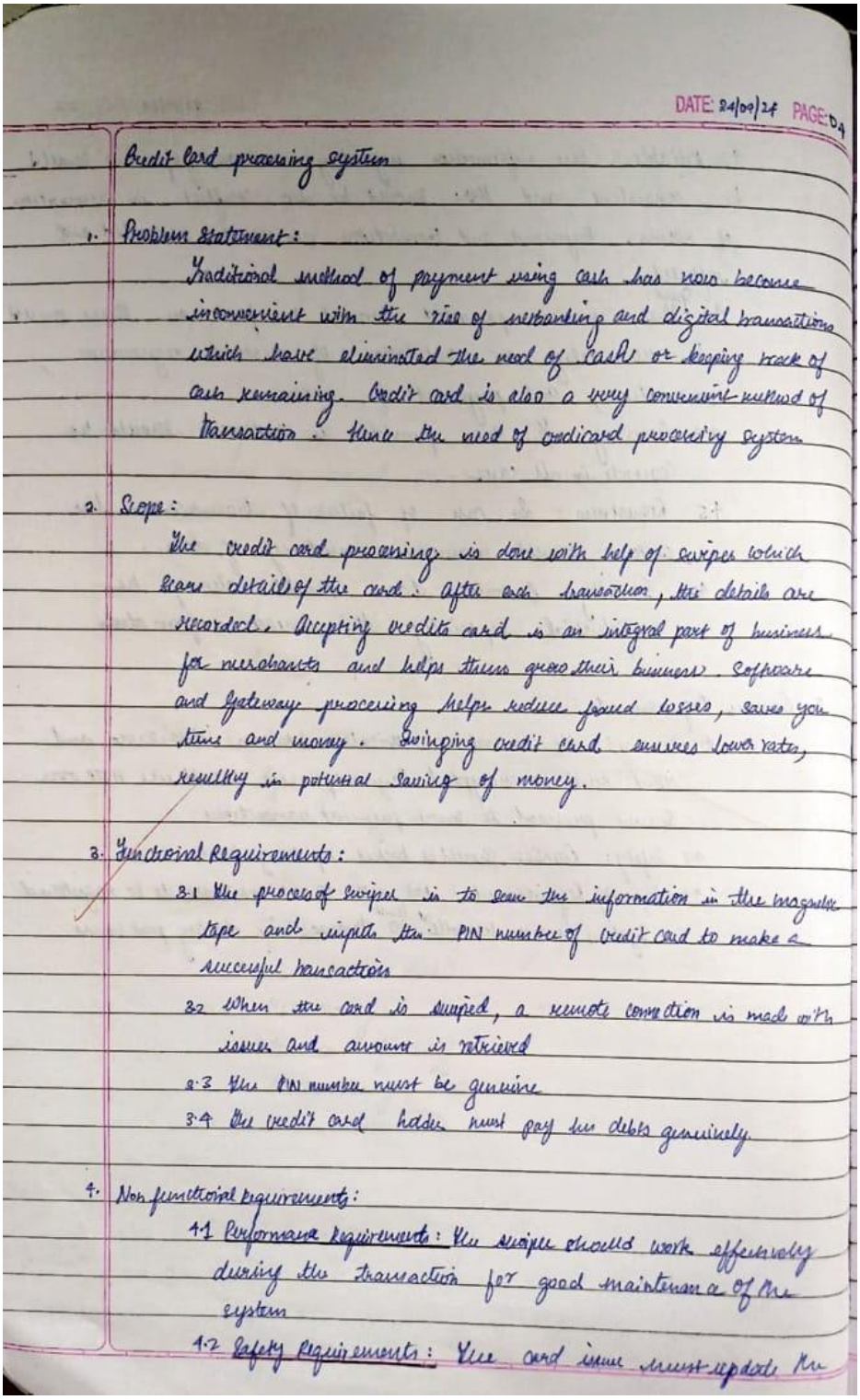
The activity diagram represents the flow of activities for making a payment, divided into swimlanes for clarity.

- **Swimlanes:**
    - **Payment Gateway**: Processes the payment details, including authentication and transaction completion.
    - **Guest**: Performs actions like selecting the room, confirming booking details, and initiating payment.
    - **System**: Validates input, updates booking records, and generates invoices.
- **Flow:**

1. Guest selects room and initiates payment.
2. System validates the booking and redirects to the Payment Gateway.
3. Payment Gateway processes the payment and confirms success.
4. System updates the booking status and generates an invoice for the guest.

2. Credit Card Processing System

2.1 SRS:



DATE: 24/09/24 PAGE: D4

Credit Card processing system

1. Problem Statement:

Traditional method of payment using cash has now become inconvenient with the rise of netbanking and digital transactions which have eliminated the need of cash or keeping track of cash remaining. Credit card is also a very convenient method of transaction. Hence the need of credicard processing system

2. Scope:

The credit card processing is done with help of swipes which scans details of the card. After each transaction, the details are recorded. Accepting credits card is an integral part of business for merchants and helps them grow their business. Software and Gateway processing helps reduce fraud losses, saves you time and money. Swinging credit card ensures lower rates, resulting in potential saving of money.

3. Functional Requirements:

3.1 The process of swipes is to scan the information in the magnetic tape and inputs the PIN number of credit card to make a successful transactions

3.2 When the card is swiped, a remote connection is made with issuer and amount is retrieved

3.3 The PIN number must be genuine

3.4 The credit card holder must pay the debts genuinely

4. Non functional requirements:

1.1 Performance requirements: The swipes should work effectively during the transaction for good maintenance of the system

1.2 Safety Requirements: The card issue must update the

record of the card holders to prevent fraudulent action and other errors that may occur during transaction.

5. Domain Requirements:

5.1 Credit card payments must be procured in compliance with Payment Card Industry Data Security Standard.

5.2 It is not permissible to transmit or obtain credit card information by email, Wireless Devices, PDAs, Instant messaging chat Application or other insecure methods unless approved by IT.

01/10/24

2.2 Class Diagram:

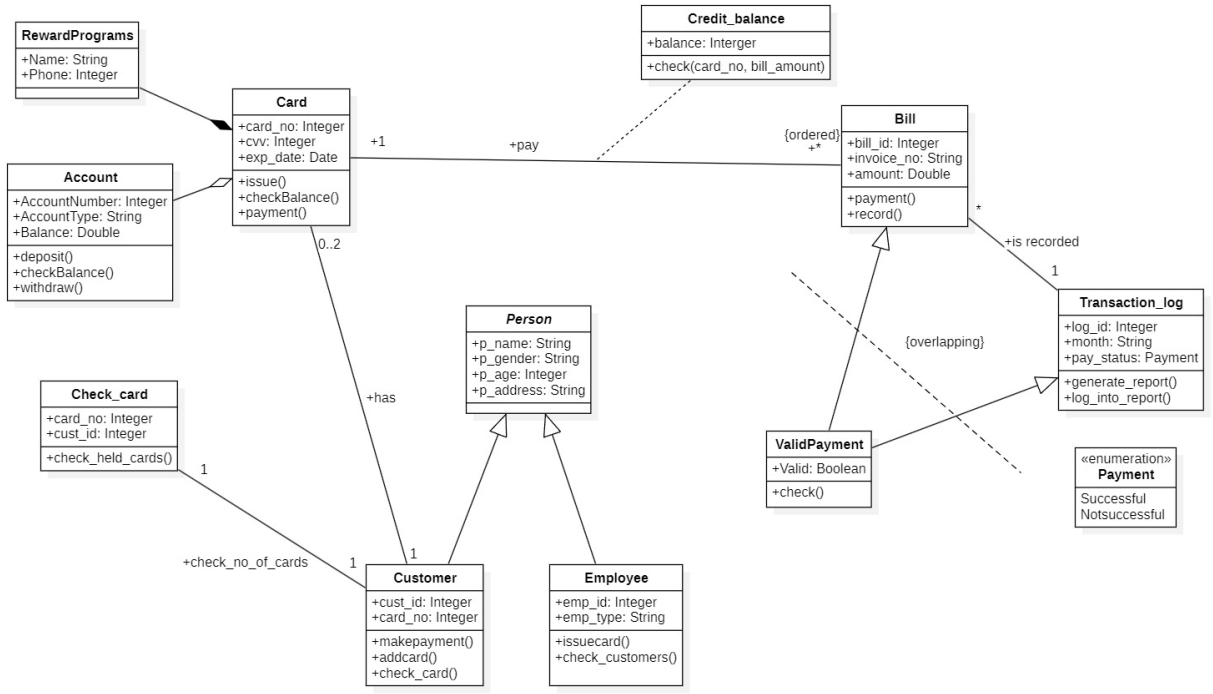


Fig 2.1: Class Diagram

Description:

The class diagram represents the static structure of the credit card processing system, highlighting the main entities and their relationships.

- **Classes:**
    - **Card Class**: Represents a credit card with attributes like card number, expiration date, CVV, and status (active/inactive).
    - **Account Class**: Represents the customer's account associated with the credit card, with attributes like account ID, balance, credit limit, and holder details.
    - **Bill Class**: Manages billing details, including billing ID, due date, amount due, and payment status.
    - **Transaction Class**: Tracks individual transactions with attributes like transaction ID, timestamp, amount, merchant details, and status (approved/declined).
- **Relationships:**

- **Card Class** is associated with the **Account Class** since each card is tied to a specific account.
- **Transaction Class** is linked to both **Card Class** and **Account Class**, representing purchases made with the card and their effect on the account.
- **Bill Class** is associated with the **Account Class** to manage periodic payments and outstanding balances.

2.3 State Diagram

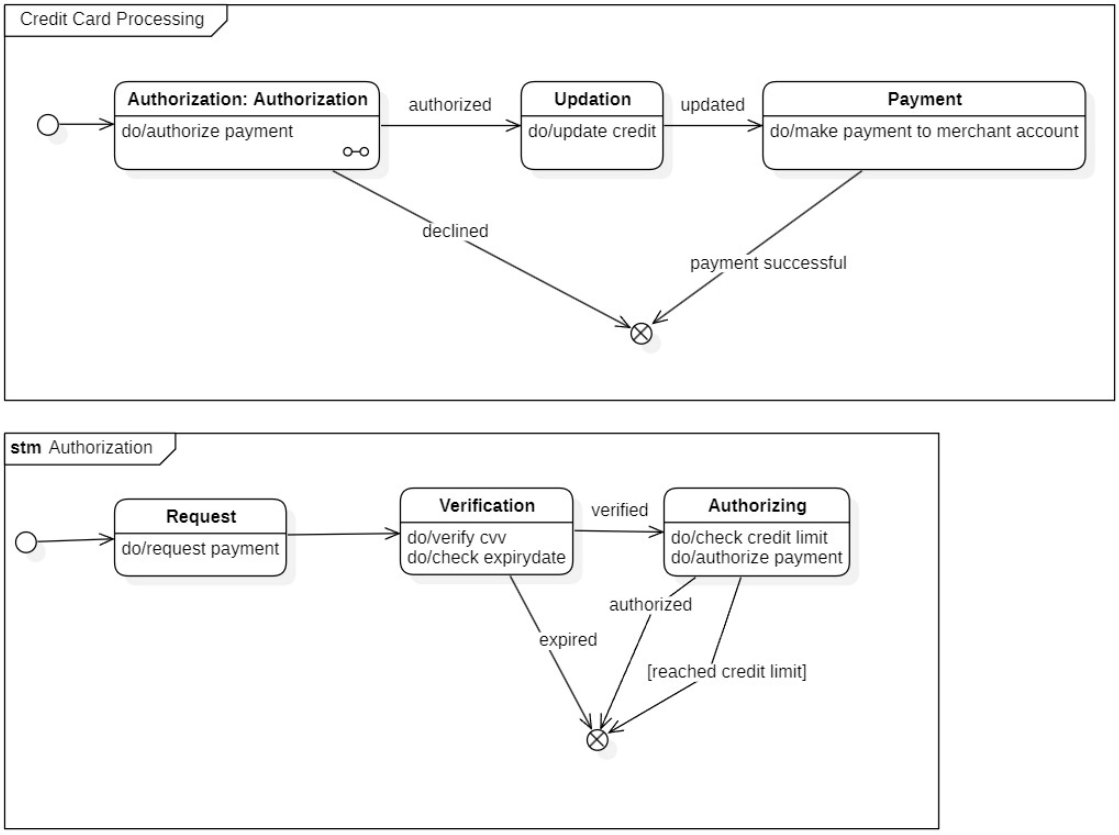


Fig 2.2: State Diagram

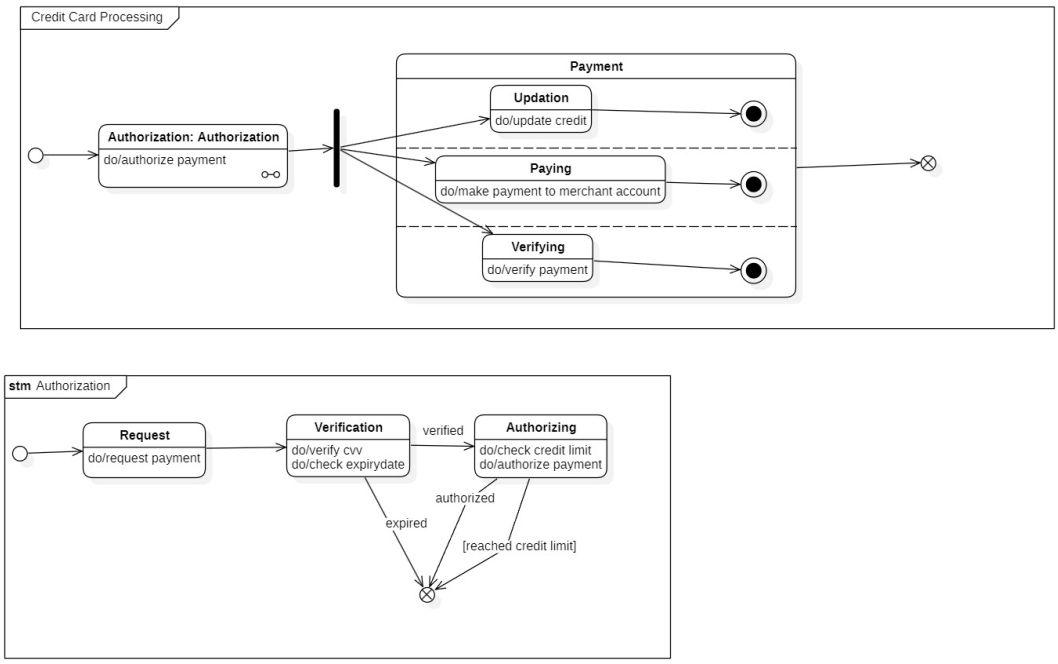


Fig 2.3: Advanced State Diagram

Description:

The state diagram showcases the dynamic behavior of the system, focusing on the transitions between different states.

- **States:**
    - **Authorization State**: Validates card details, checks account status, and approves or declines the transaction.
        - Sub-states:
            - Card Validation
            - Fraud Detection
            - Approval/Rejection
    - **Payment State**: Handles the payment process, involving concurrent sub-states:
        - **Updation of Credit**: Updates the available credit after a successful transaction.
        - **Paying**: Deducts the amount from the account and marks the transaction as paid.
        - **Verifying**: Confirms that the payment is correctly processed and logged.

2.4 Use Case Diagram:

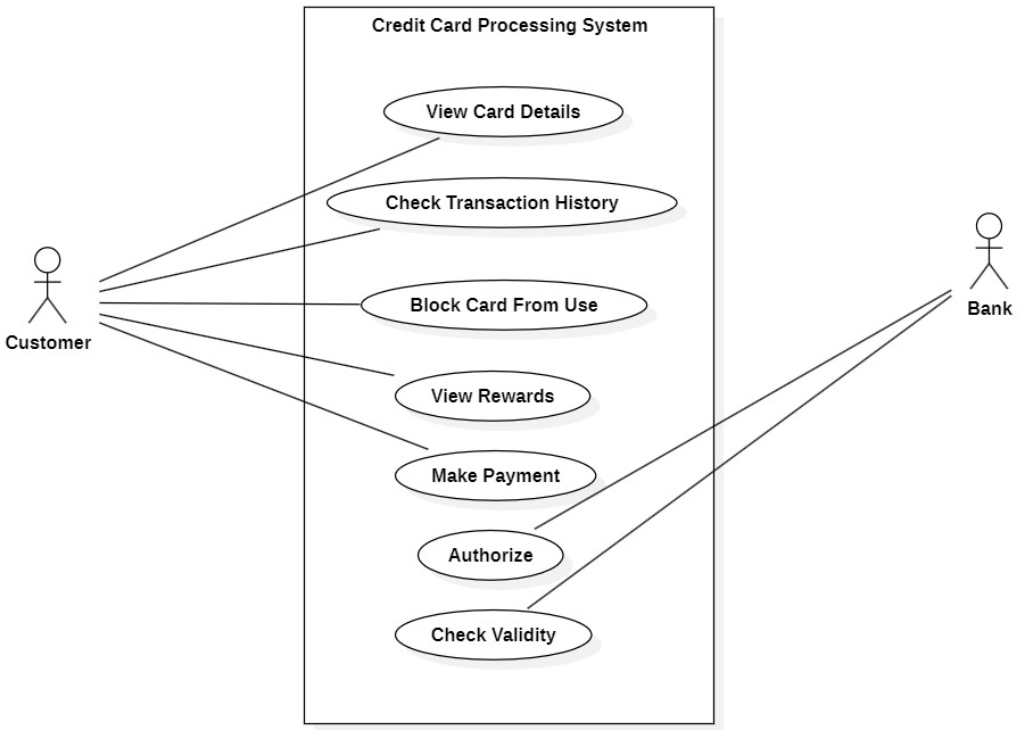


Fig 2.3: Use Case Diagram

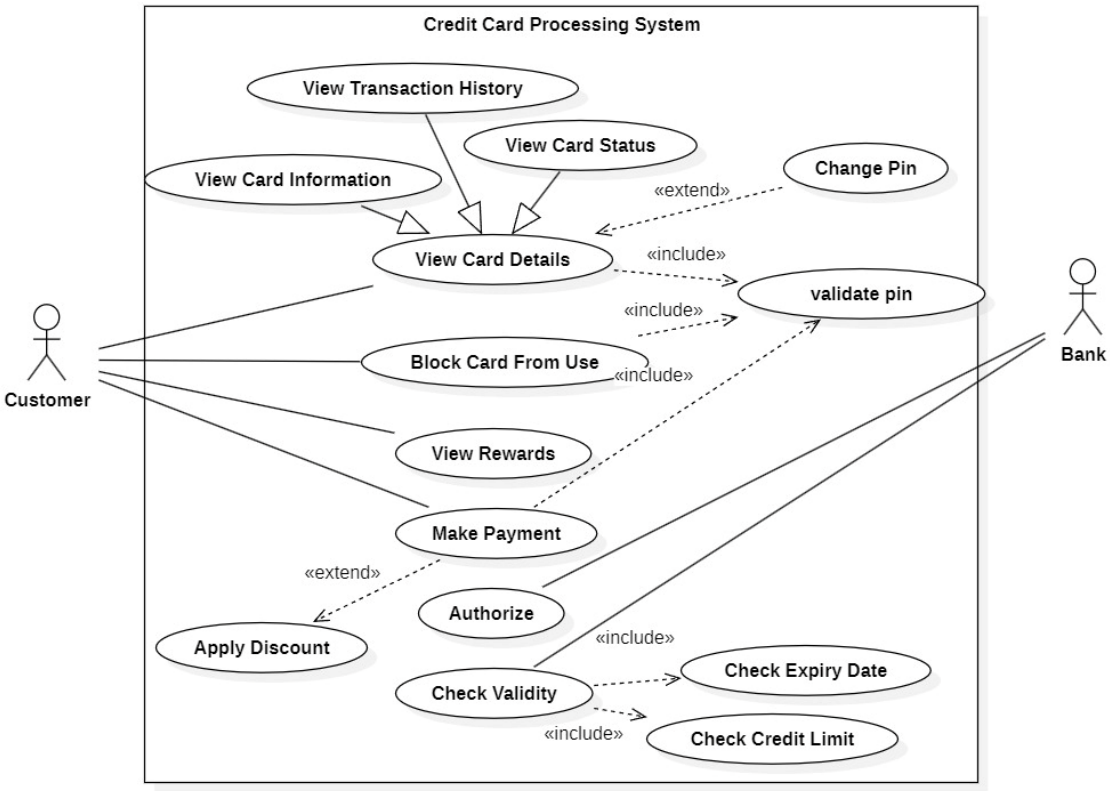


Fig 2.4: Advanced Use Case Diagram

Description:

The use case diagram identifies the interactions between actors and the system functionalities.

- **Actors:**
    - **Customer**: Manages card details, transactions, and rewards.
    - **Merchant**: Initiates transactions and payments.
    - **System**: Authorizes payments and manages account information.
- **Use Cases:**
    - **View Card Details**: Customers can check their credit card information.
    - **Check Transaction History**: Provides a log of past transactions for review.
    - **Block Card from Use**: Allows customers to block a lost or stolen card.
    - **View Rewards**: Displays accrued rewards or cashback for the customer.
    - **Authorize**: Validates a transaction initiated by the merchant.
    - **Make Payment**: Processes bill payments for the credit card.

2.5 Sequence Diagram:

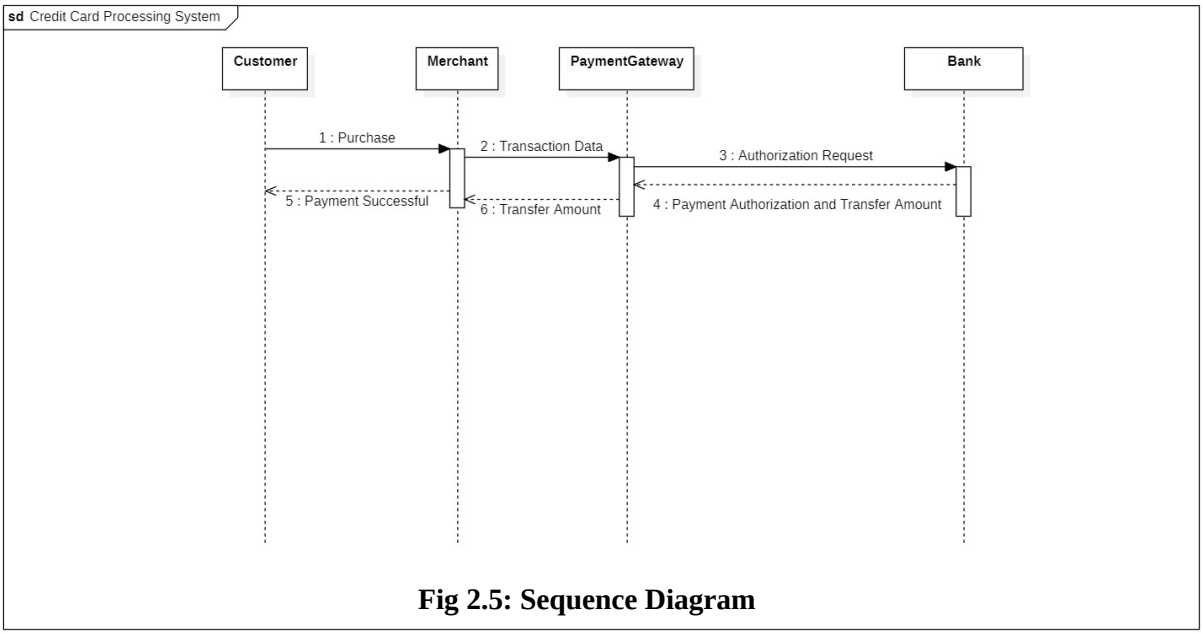


Fig 2.5: Sequence Diagram

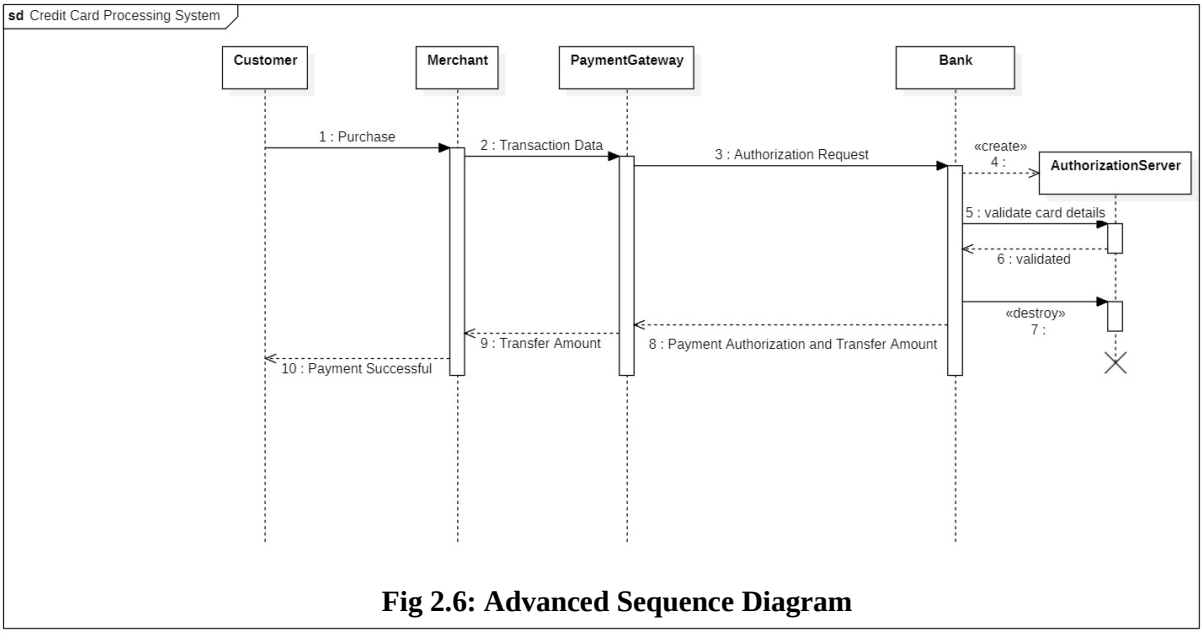


Fig 2.6: Advanced Sequence Diagram

Description:

The sequence diagram represents the interaction flow during a transaction.

- **Objects:**
    - **Customer**: Initiates the transaction by providing card details.

- **Merchant**: Sends the payment request to the system.
- **Payment Gateway**: Mediates between the merchant and the bank, ensuring secure processing.
- **Bank**: Validates the card and processes the transaction.
- **Authorization (Transient Object)**: Created temporarily to validate the transaction.

- **Flow:**

1. Customer provides card details to the Merchant.
2. Merchant forwards the details to the Payment Gateway.
3. Payment Gateway requests transaction approval from the Bank.
4. Bank validates card and account details, creating an Authorization object.
5. Authorization object verifies and responds with an approval/decline status.
6. Payment Gateway informs the Merchant, who finalizes the transaction.

2.6 Activity Diagram

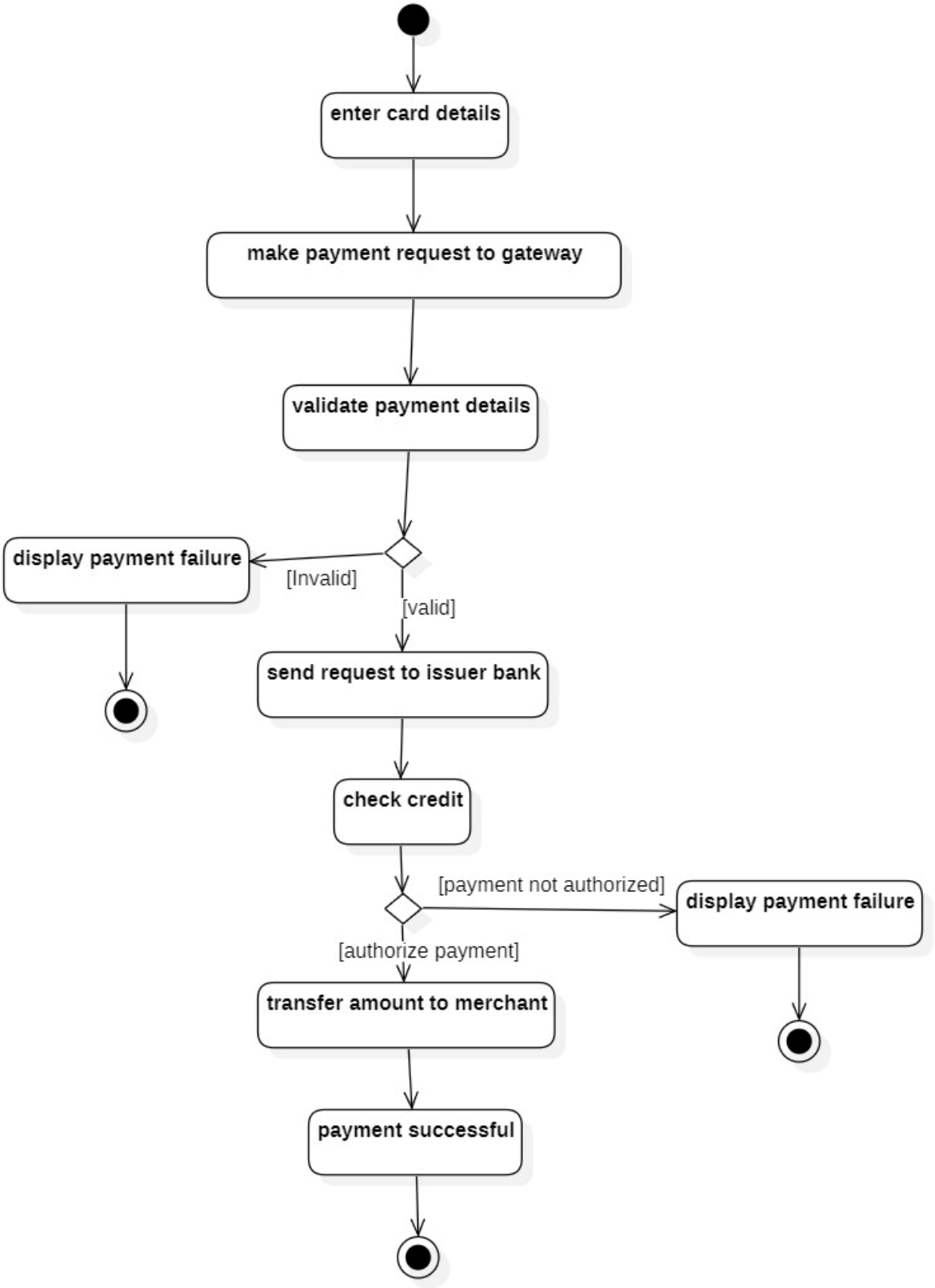


Fig 2.7: Activity Diagram

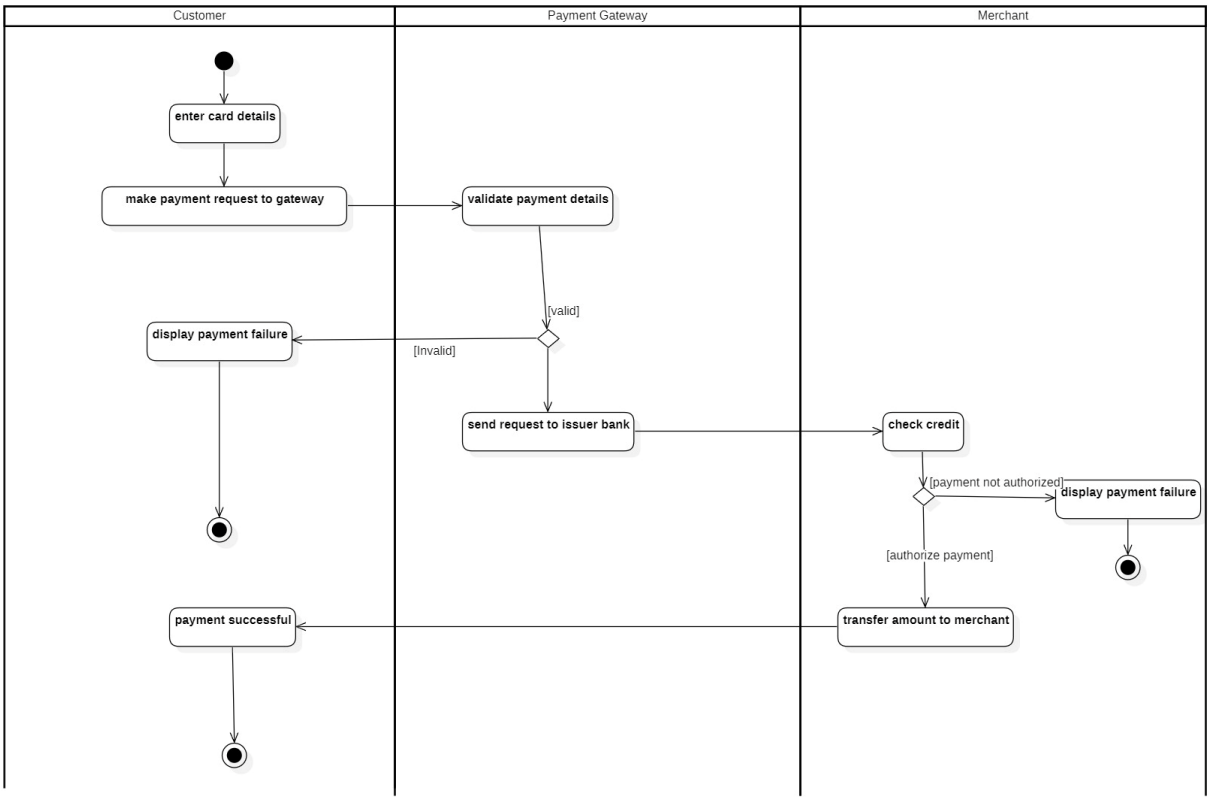


Fig 2.8: Advanced Activity Diagram

Description:

The activity diagram outlines the process flow during a payment, divided into swimlanes for clarity.

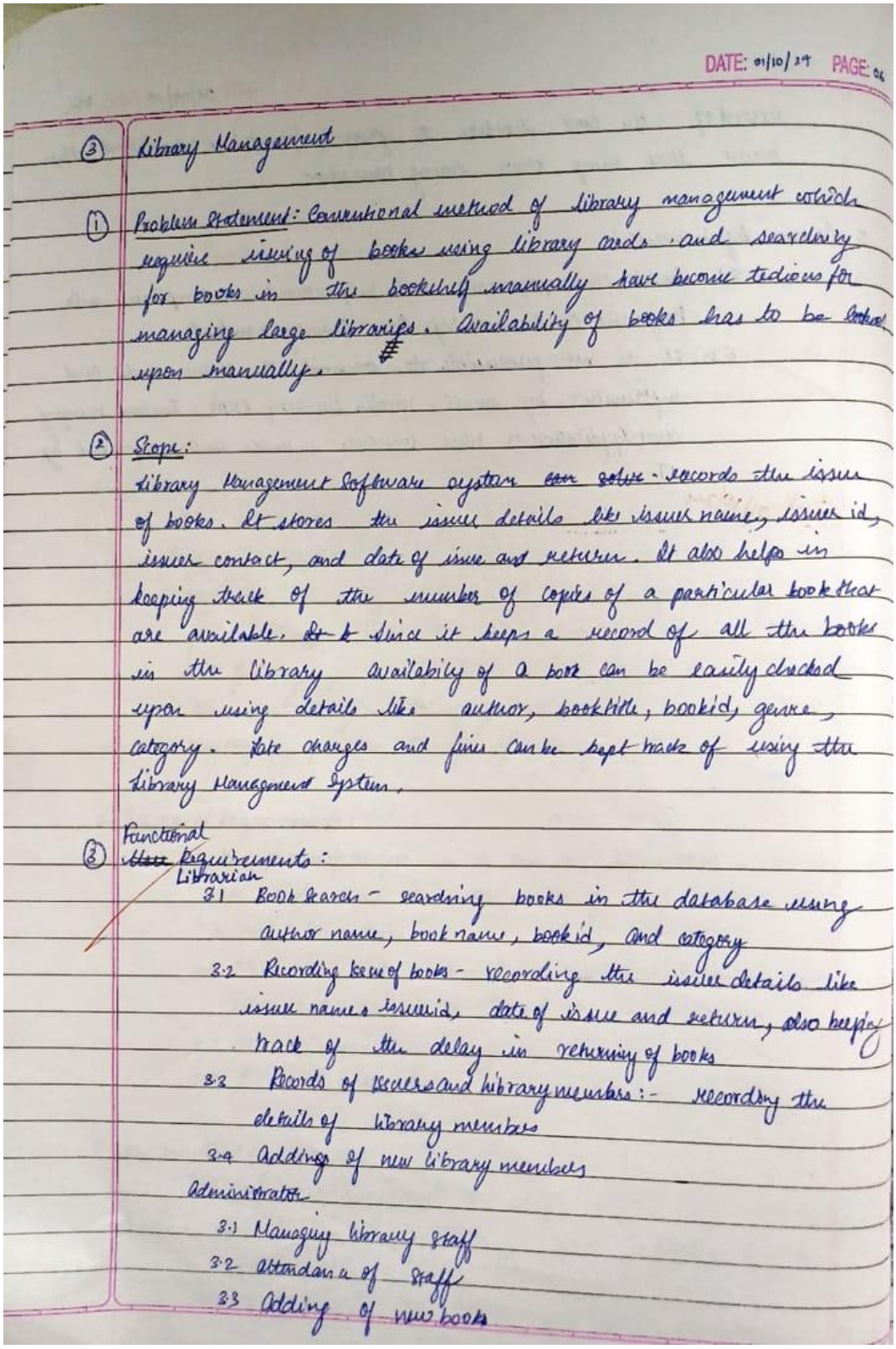
- **Swimlanes:**
    - **Payment Gateway**: Manages secure processing and communication with the bank.
    - **Customer**: Initiates the transaction and confirms payment.
    - **Merchant**: Processes the payment request and completes the transaction.
- **Flow:**

1. Customer initiates payment by providing card details to the Merchant.
2. Merchant sends payment details to the Payment Gateway.
3. Payment Gateway validates details and communicates with the Bank.
4. Bank processes the transaction and confirms success/failure to the Payment Gateway.

5. Payment Gateway updates the Merchant, who completes the process and informs the Customer.

3. Library Management System

3.1 SRS:



DATE: 01/10/24 PAGE: 04

(3) Library Management

(1) Problem Statement: Conventional method of library management which require issuing of books using library cards and searching for books in the bookshelf manually have become tedious for managing large libraries. Availability of books has to be looked upon manually.

(2) Scope:
Library Management Software system ~~can solve~~ records the issue of books. It stores the issue details like issuer name, issuer id, issuer contact, and date of issue and return. It also helps in keeping track of the number of copies of a particular book that are available. ~~It~~ Since it keeps a record of all the books in the library availability of a book can be easily checked upon using details like author, booktitle, bookid, genre, category. Late charges and fines can be kept track of using the library Management System.

(3) ~~Non~~ Functional Requirements:
Librarian
3.1 Book search - searching books in the database using author name, book name, book id, and category
3.2 Recording issue of books - recording the issuer details like issuer name, issuer id, date of issue and return, also keeping track of the delay in returning of books
3.2 Records of issuers and library members :- recording the details of library members
3.4 Adding of new library members
Administrator
3.1 Managing library staff
3.2 attendance of staff
3.3 Adding of new books

~~3~~

Non Functional Requirements:

Security: Role based access control for different user types.

Scalable: The LBS should be able to manage large connection of books and users

Usable: The library management system should be convinient to use

~~Performance~~:

Performance Requirements:

The system should handle at least 500 simultaneous updates of books

Interface Requirements:

The interface should be user friendly and easy to use.

Design Constraints:

The system must be designed to allow web usability. That i the system must be designed in such a way that will be easy to use and visible on most of the browser

Preliminary Schedule and Budget

The actual library management and app development cost wi range between $25000 - $35000

01/10/24

3.2 Class Diagram:

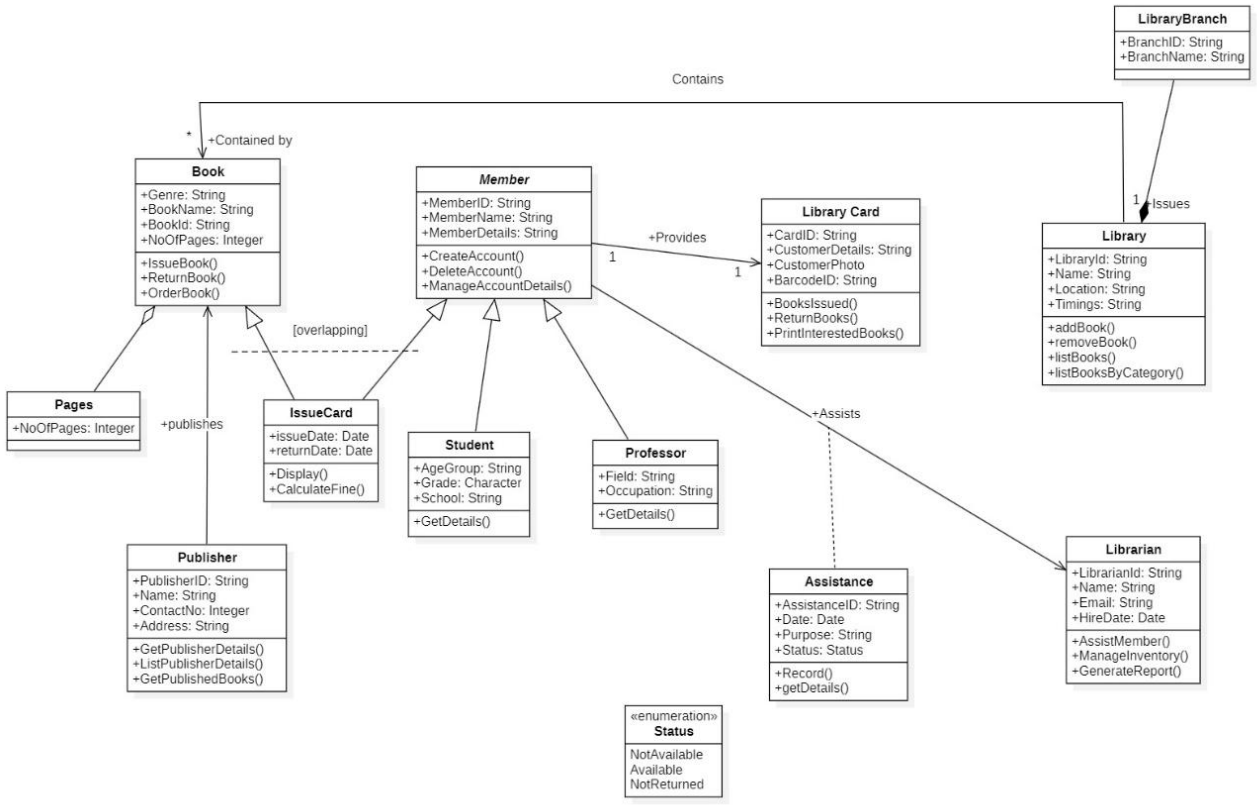


Fig 3.1: Class Diagram

Description:

The class diagram outlines the structure of the library management system, highlighting key entities and their relationships.

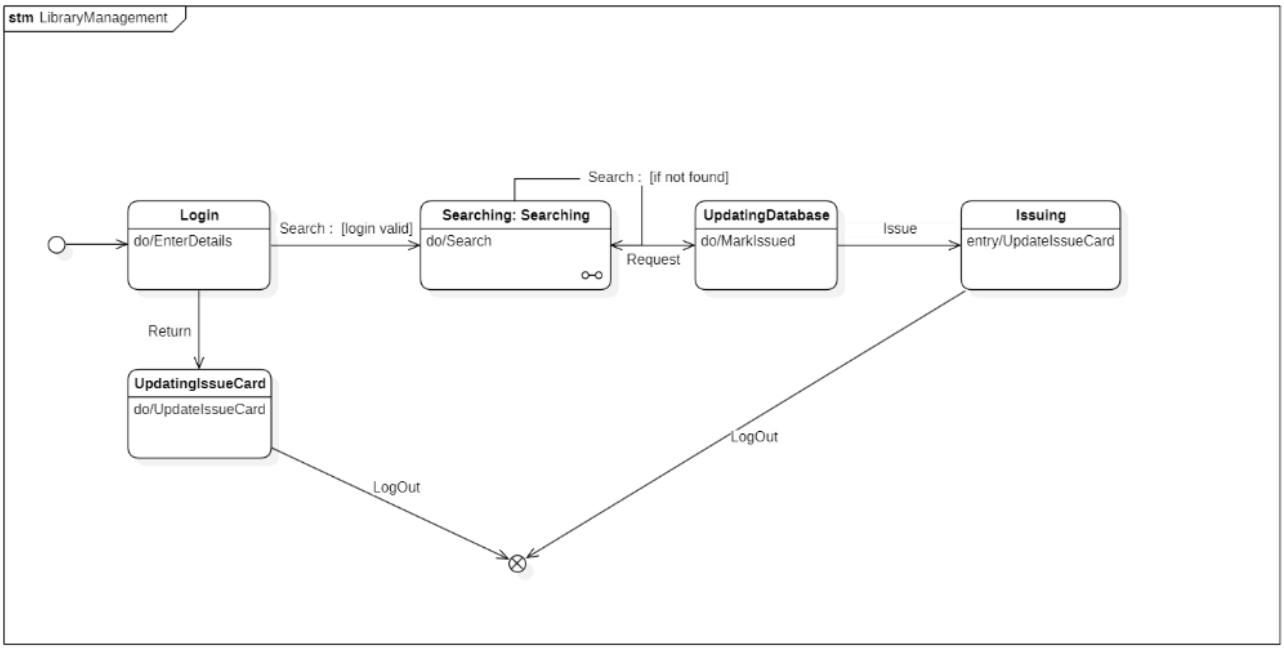
- **Classes:**
    - **Book Class**: Represents a book with attributes like book ID, title, author, genre, and availability status.
    - **Member Class**: Abstract class representing library members, generalized into:
        - **Student Class**: A specific type of member with attributes like student ID and grade.
        - **Professor Class**: A specific type of member with attributes like professor ID and department.
    - **LibraryCard Class**: Represents a unique card assigned to members to track borrowing activity.

- **IssueCard Class**: Represents a temporary overlap of **Book** and **Member**, tracking the details of issued books, including issue date, due date, and status.
- **Library Class**: Manages the overall library system, composed of:
  - **LibraryBranch Class**: Represents individual branches with attributes like branch ID, name, and address.

- **Relationships:**
  - The **Member Class** and **LibraryCard Class** have an association since each member holds a library card.
  - The **IssueCard Class** overlaps the **Book Class** and **Member Class**, representing borrowed books.
  - **Library Class** is composed of multiple **LibraryBranch Classes** for a distributed library system.

‘

3.3 State Diagram:



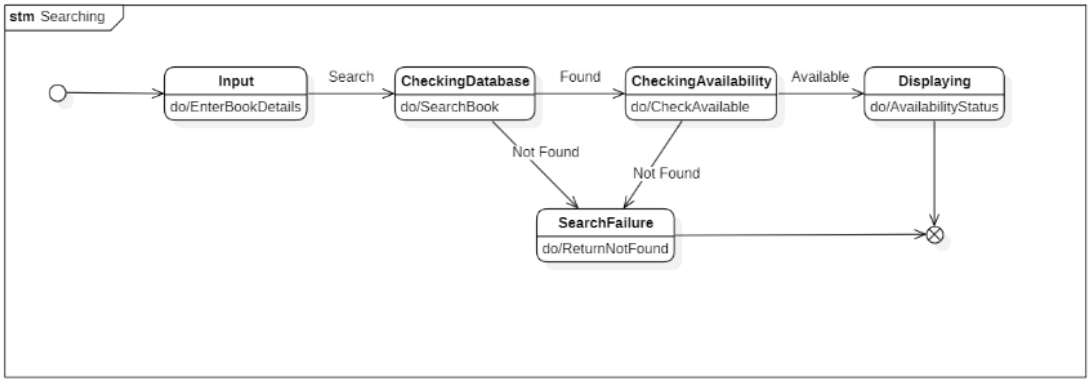


Fig 3.2: State Diagram

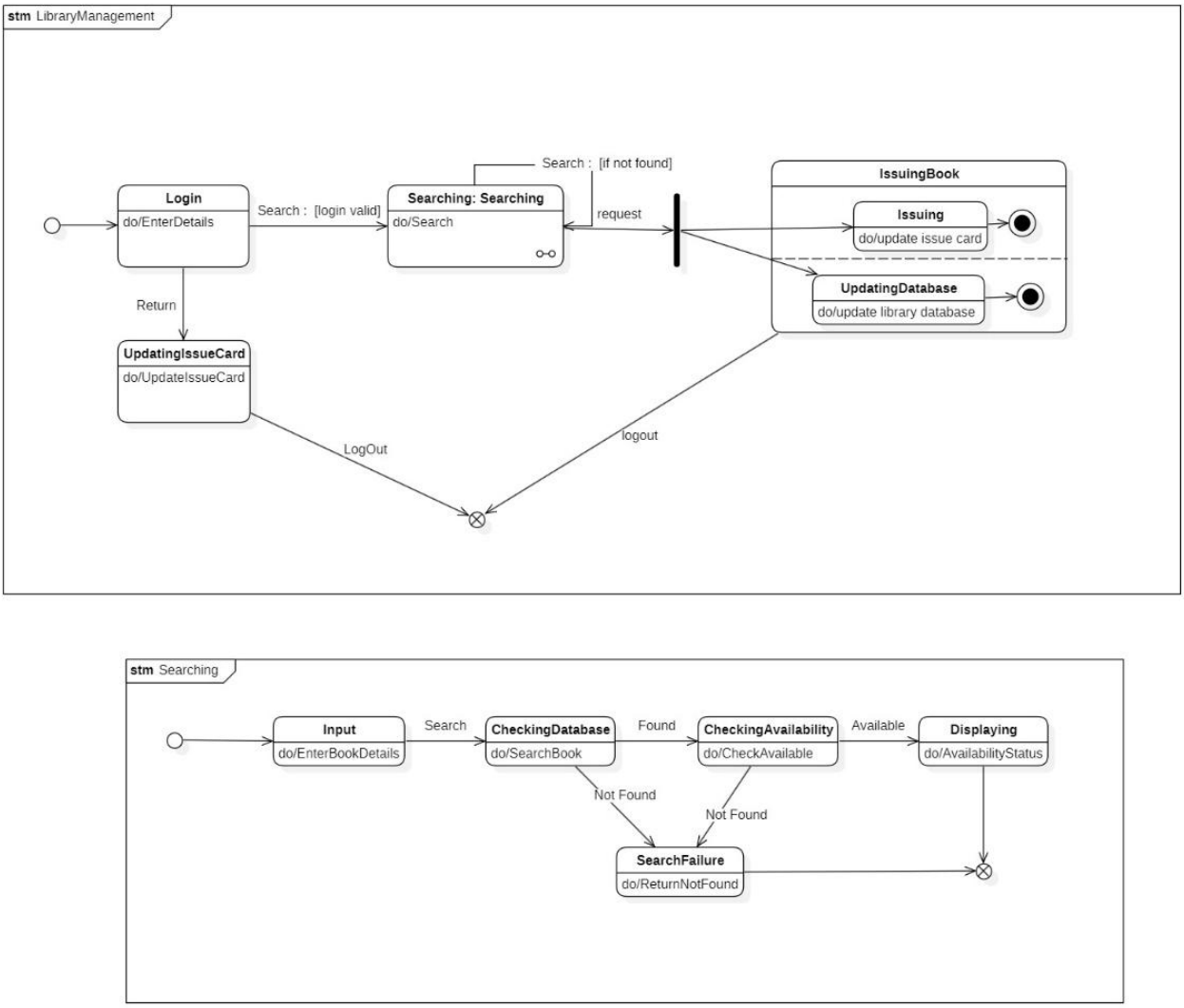


Fig 3.3: Advanced State Diagram

Description:

The state diagram illustrates the transitions between various states in the library management system.

- **States:**
    - **Login State**: Represents when users or librarians log into the system.
    - **Searching State**: Handles searching for books based on criteria like title, author, or genre.
        - Sub-states:
            - Keyword Search

        - Advanced Search (e.g., by publication date or availability)
    - **Updating Database**: Handles the addition, deletion, or modification of records (e.g., book details or member details).
        - Concurrent with **Issuing**.
    - **Issuing State**: Manages the borrowing process, including verifying book availability and member status.
    - **Updating Issue Card**: Updates records in the **IssueCard Class** to track borrowed books.
- **Concurrency:**
    - **Issuing** and **Updating Database** occur simultaneously.

3.4 Use Case Diagram:

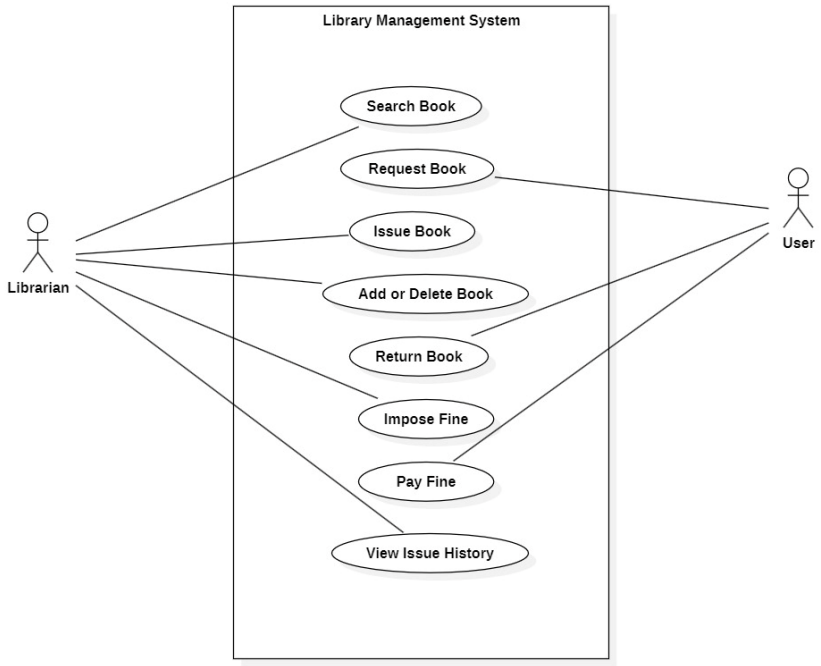


Fig 3.4: Use Case Diagram

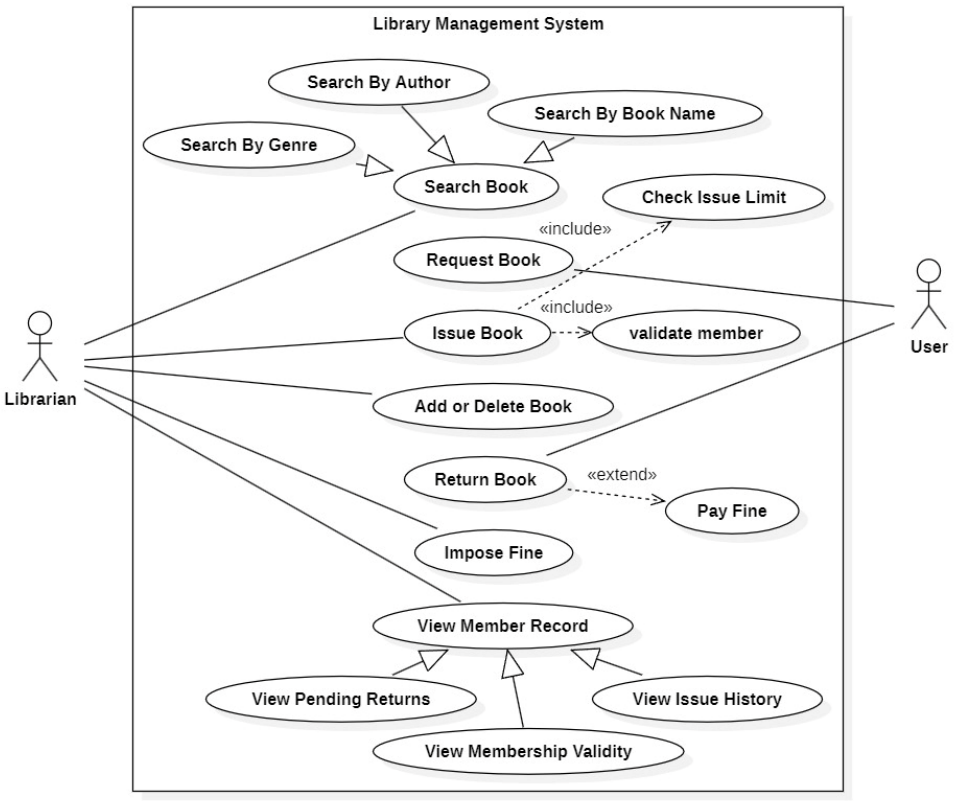


Fig 3.5: Advanced Use Case Diagram

Description:

The use case diagram defines the interactions between actors and system functionalities.

- **Actors:**
    - **Librarian**: Manages library operations such as adding/deleting books and imposing fines.
    - **Member**: Includes both students and professors, who interact with the system for book-related tasks.
- **Use Cases:**
    - **Search Book**: Members search for books in the library catalog.
    - **Request Book**: Members can place a request for unavailable books.
    - **Issue Book**: Books are issued to members after verification.
    - **Add or Delete Book**: Librarians manage the catalog by adding or removing books.
    - **Return Book**: Members return borrowed books, and the system updates the database.
    - **Impose Fine**: Librarians impose fines for overdue returns.
    - **Pay Fine**: Members pay the imposed fines through the system.
    - **View Issue History**: Members can view their borrowing history.

3.5 Sequence Diagram:

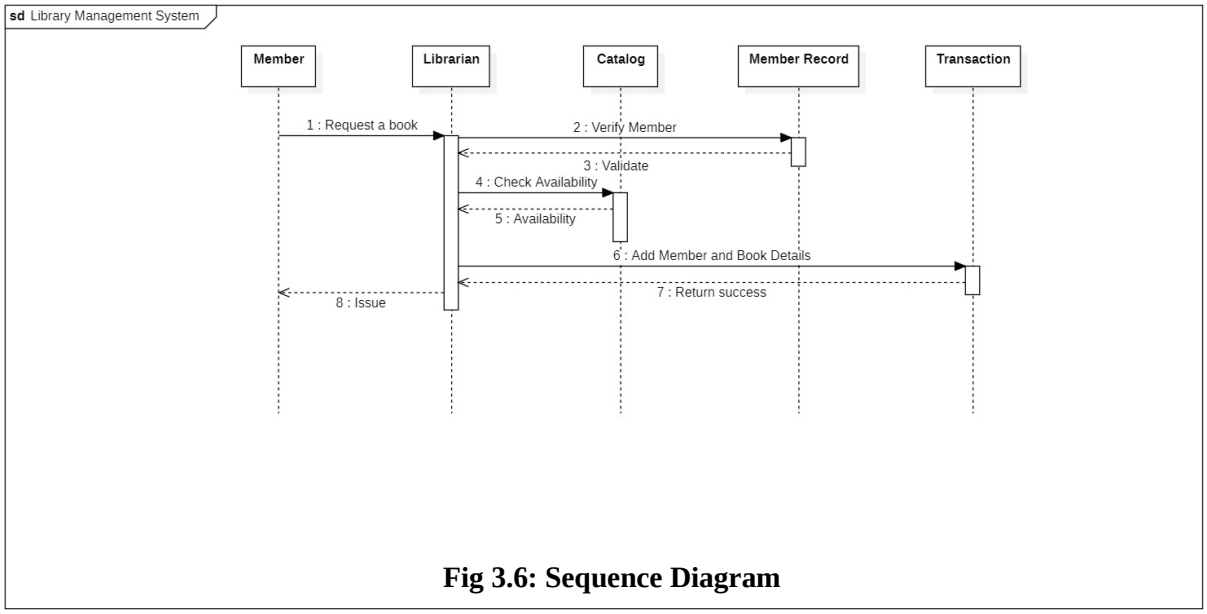


Fig 3.6: Sequence Diagram

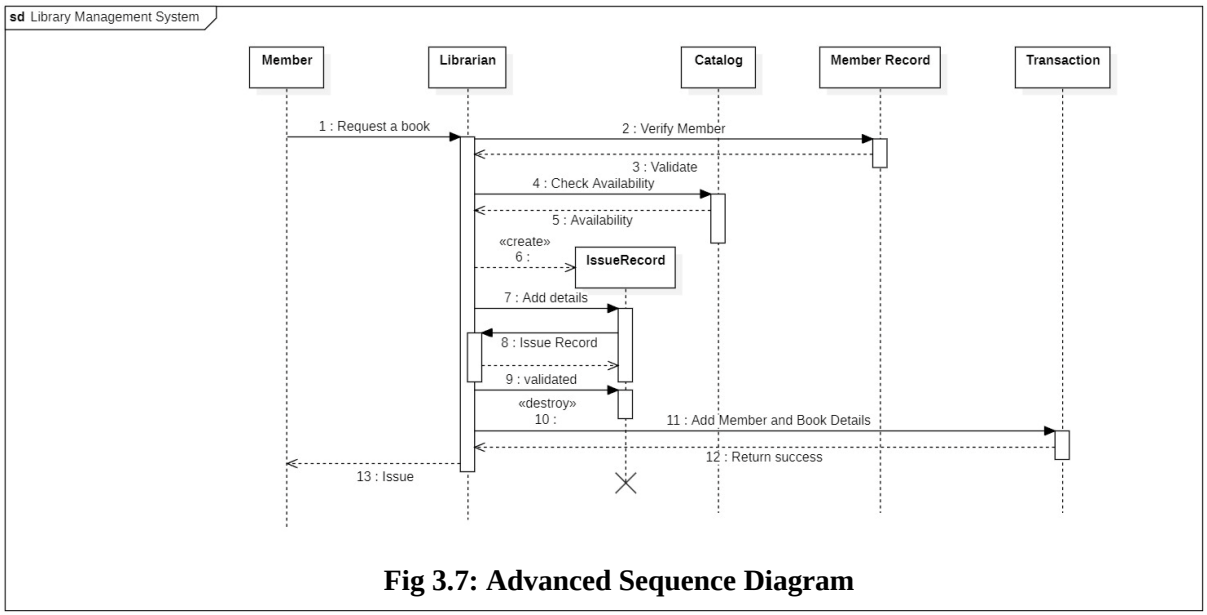


Fig 3.7: Advanced Sequence Diagram

Description:

The sequence diagram illustrates the interaction flow for issuing a book.

- **Objects:**
    - **Member**: Initiates a book issue request.
    - **Librarian**: Verifies the member's credentials and book availability.

- **Catalog**: Checks the availability of the requested book.
- **Member Record**: Updates the member's borrowing details after the book is issued.
- **Transaction**: Represents the book-issuing transaction, tracking details like issue and due dates.

- **Flow:**

1. Member searches for a book and requests to issue it.
2. Librarian checks the Member Record and verifies the book in the Catalog.
3. Catalog confirms book availability.
4. Transaction object records the issue details.
5. Member Record is updated with the new transaction.

3.6 Activity Diagram

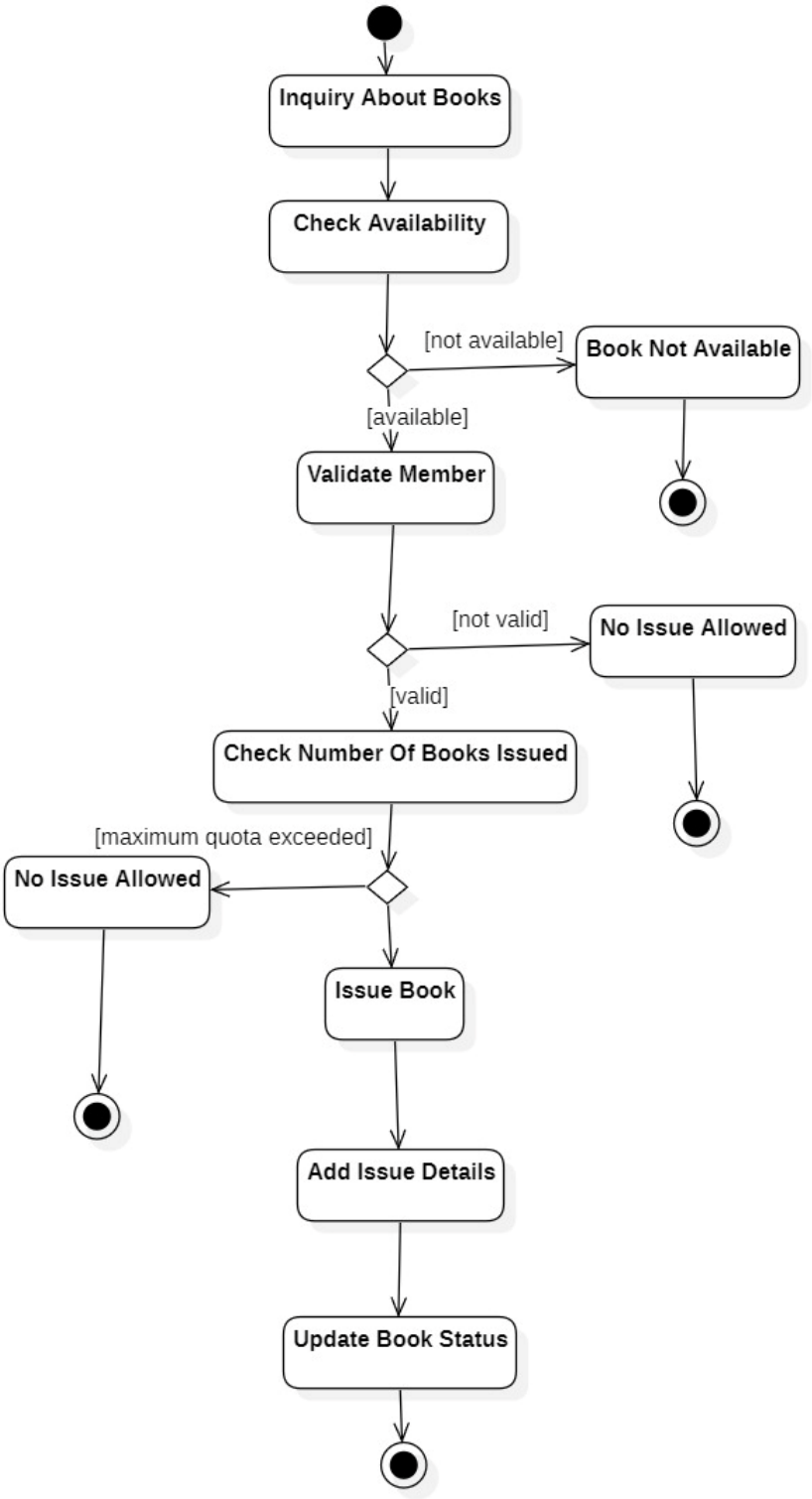


Fig 3.8: Activity Diagram

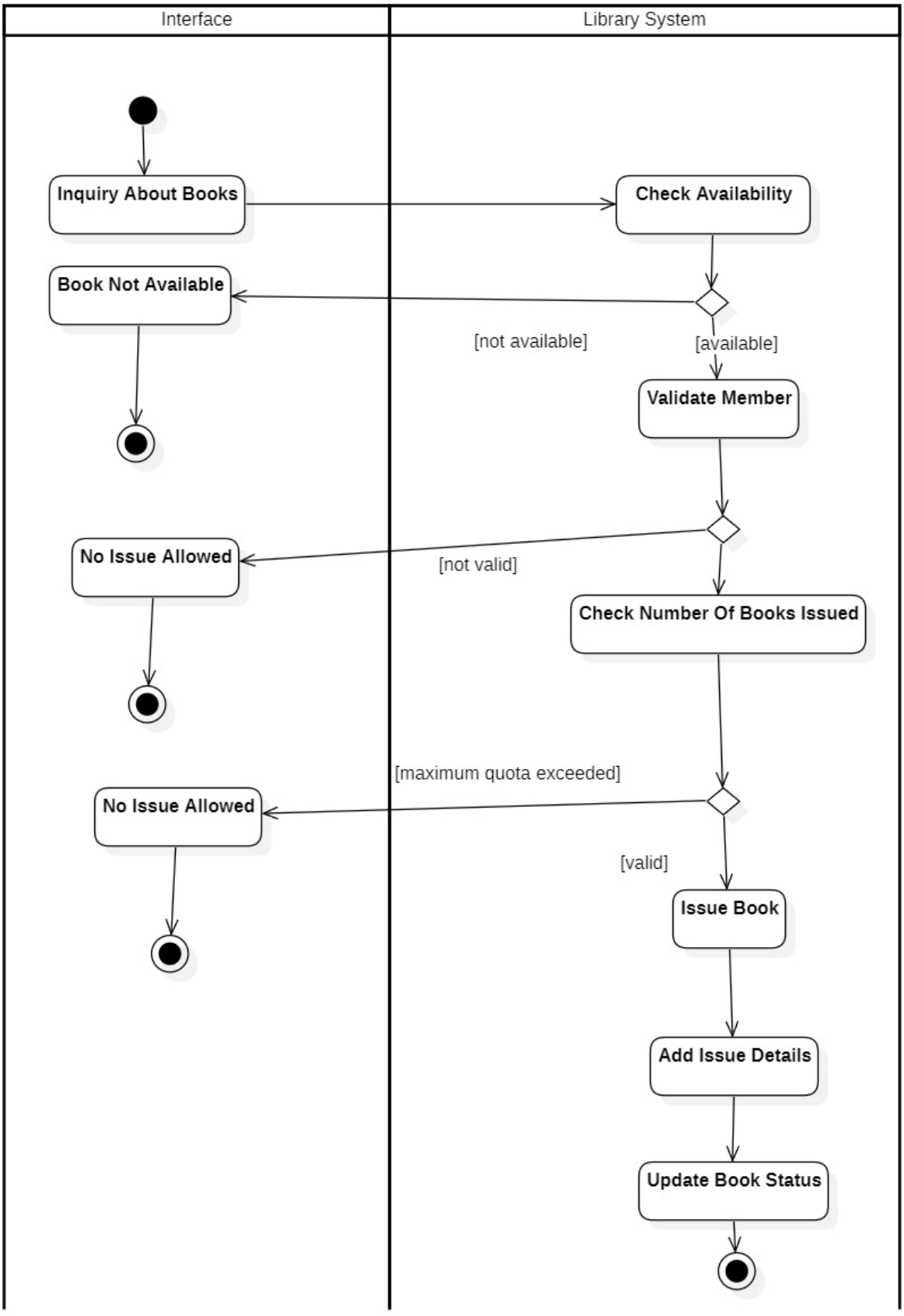


Fig 3.9: Advanced Activity Diagram

Description:

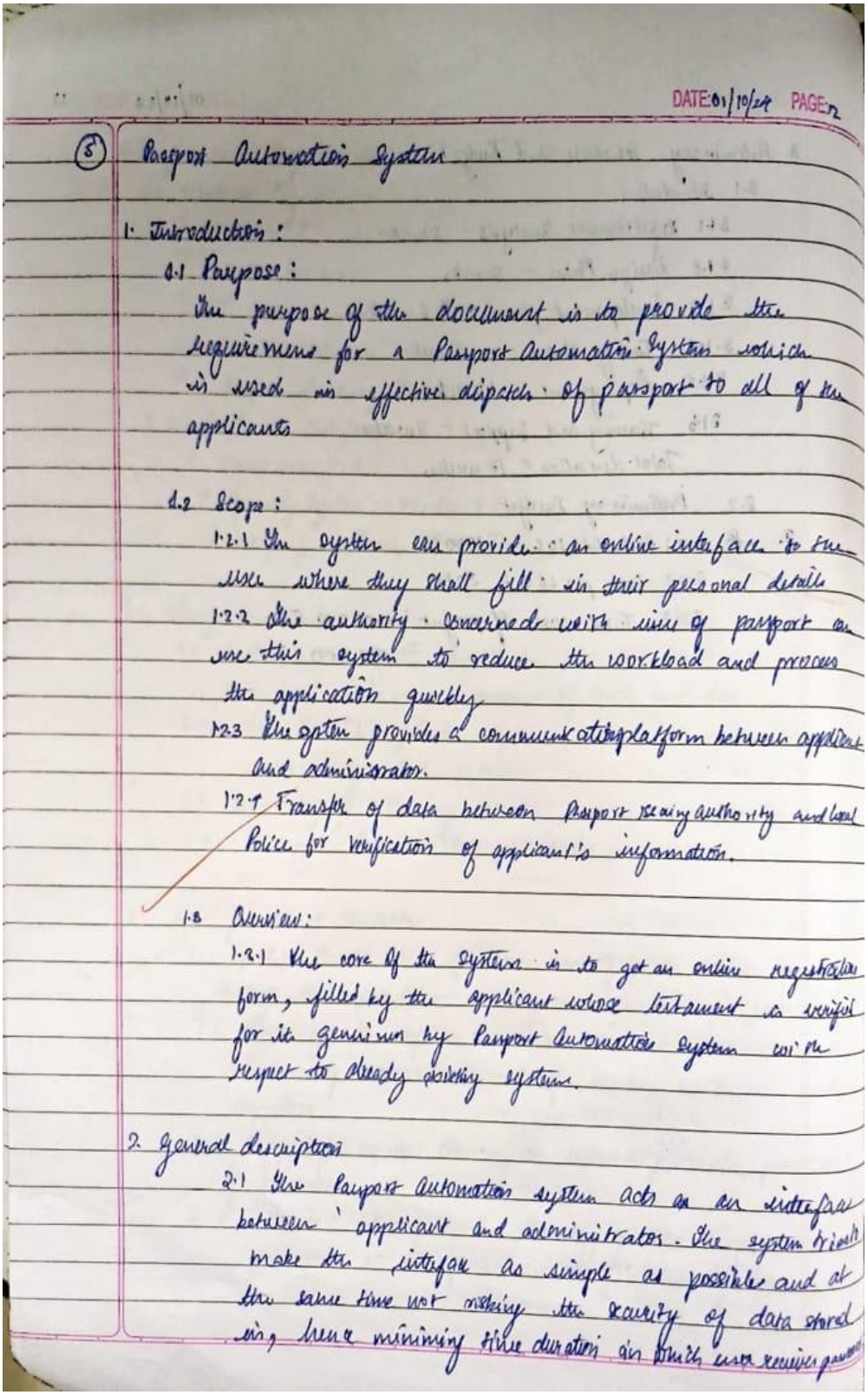
The activity diagram shows the process of issuing a book, divided into swimlanes for clarity.

- **Swimlanes:**
    - **Interface**: Represents the system's user interface where members and librarians interact.
    - **Library Management**: Handles backend operations like book availability checks, member validation, and transaction processing.
- **Flow:**

1. Member logs in and searches for a book via the Interface.
2. Library Management validates the request and checks book availability.
3. If available, the book is issued, and the transaction is recorded.
4. Interface updates the member's borrowing details and confirms the issue.

4. Passport Automation System

4.1 SRS:



DATE: 01/10/24 PAGE: 12

(5) Passport Automation System

1. Introduction:

1.1 Purpose:

The purpose of the document is to provide the requirements for a Passport Automation System which is used in effective dispatch of passport to all of the applicants

1.2 Scope:

1.2.1 The system can provide an online interface to the user where they shall fill in their personal details

1.2.2 The authority concerned with issue of passport can use this system to reduce the workload and process the application quickly

1.2.3 The system provides a communication platform between applicant and administrator.

1.2.4 Transfer of data between Passport issuing authority and local Police for verification of applicant's information.

1.3 Overview:

1.3.1 The core of the system is to get an online registration form, filled by the applicant whose testament is verified for its genuinity by Passport Automation System with respect to already existing systems.

2. General description

2.1 The Passport Automation system acts as an interface between applicant and administrator. The system tries to make the interface as simple as possible and at the same time not making the security of data stored in, hence minimizing time duration in which user receives passport

2.2 It aims at improving the efficiency in the issue of Passport and reduces the complexities involved in it to maximum extent possible.

3. Functional Requirements:

3.1 Passport Information: This module is helpful for the people to move to other states or countries for their needs.

3.2 Applying For Passport: This module is used for those to apply for the passport and which certificates are needed to submit for the particular types of passport.

3.3 Payment: This module is used to pay fees in different ways.

3.4 Document Uploading: To submit the documents

3.5 Verification Module: To verify all certifications online and change the status.

3.6 Authentication Module: This module is used to check whether user is valid or not.

3.7 Status verification and feedback: To check status of verification process of and delivery of passport.

4. Interface Requirements: The interface should be easy to use and userfriendly. It should have.

→ User Input Form fields
→ Status DashBoard
→ Notifications
→ File Upload
→ File Size and Format Validation
→ Payment Gateway
→ Profile Management
→ FAQ section

5. Performance Requirements:
5.1 Should be able to process 500 applications/hour
5.2 Should support upto 200 users simultaneously.
5.3 Should Response under 2 seconds
5.4 Should be scalable for peak application period.

6. Design Constraints:
6.1 Software Interface:
6.1.1 FrontEnd: Microsoft Visual Basic 6.0
6.1.2 BackEnd: MS Access database
6.2 Hardware Interface
The server is directly connected to client systems.
The client systems should have access to database in server.

7. Non-Functional Attributes:
7.1 Reliability: Should be able to continue operating in event of hardware or software failure as well as Quick restoration
7.2. Performance: Quick response time
7.3 Security: Implementation of strong encryption and secure protocols to protect sensitive information.

8. Preliminary Schedule and Budget:
8.1 Preliminary Schedule:
8.1.1 Planning and Requirement gathering should take 1-2 months
8.1.2 System design should take two months
8.1.3 Development should take 6 months
8.1.4 Testing should take 2 months
8.1.5 Deployment should take 1 month
8.2. Preliminary Budget: Estimated LOC = 50,000 lines
Cost per loc = 10 Rs
Total LOC cost = ₹ 5,00,000

4.2 Class Diagram:

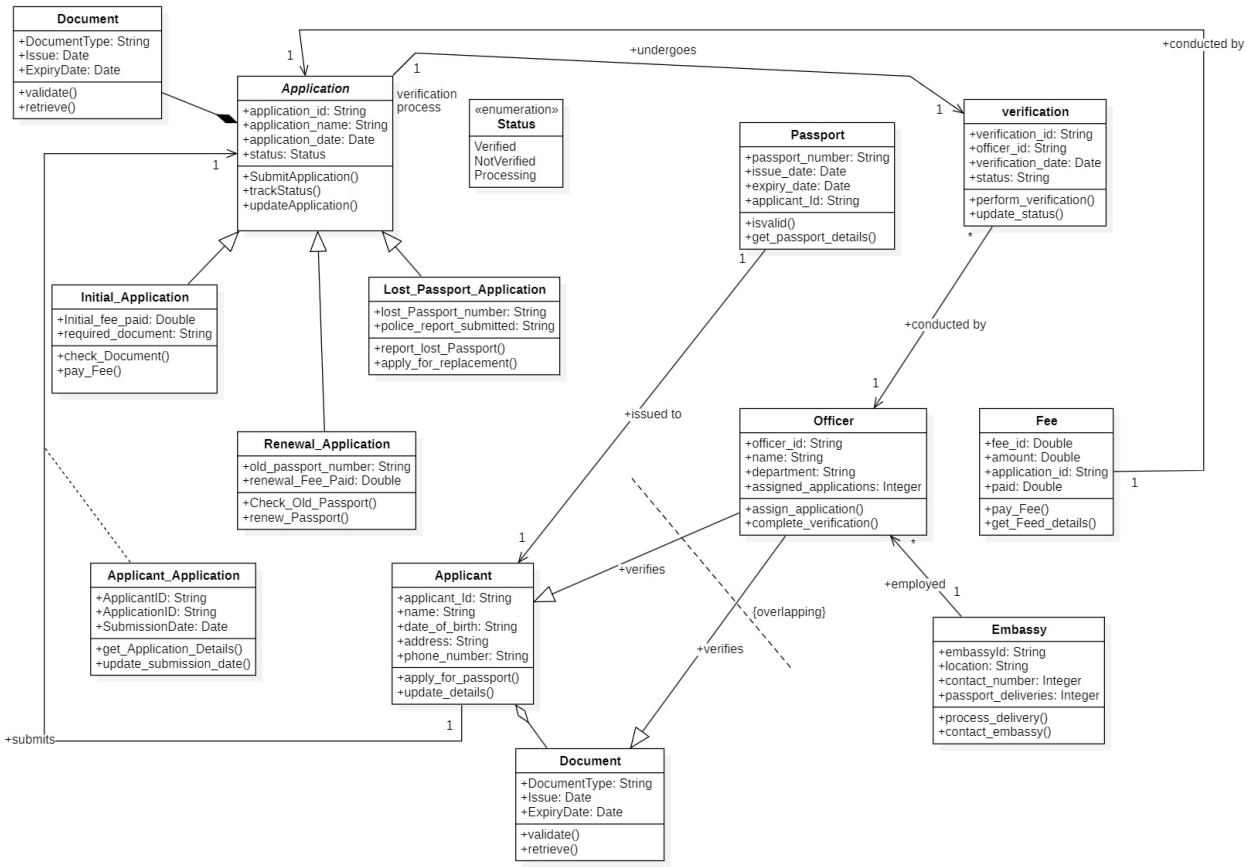


Fig 4.1: Class Diagram

Description:

The class diagram represents the static structure of the system, highlighting main entities and their relationships.

- **Classes:**
    - **Application Class**: Represents the overall process of passport applications, with generalizations:
        - **Initial Application**: For first-time passport applicants, containing attributes like birth certificate details and ID proof.
        - **Renewal Application**: For applicants renewing expired or soon-to-expire passports, with renewal-specific attributes.
        - **Lost Passport Application**: For applicants reporting and replacing lost passports, including police complaint details.

- **Passport Class**: Represents a passport with attributes like passport number, issue date, expiration date, and applicant details.
- **Verification Class**: Manages the verification process, with attributes like verification ID, type (police, regional admin), and status (approved/rejected).

- **Relationships:**
  - **Application Class** is associated with **Passport Class** to represent the issuance process.
  - **Verification Class** is linked to **Application Class** to ensure every application goes through proper checks.

4.3 State Diagram:

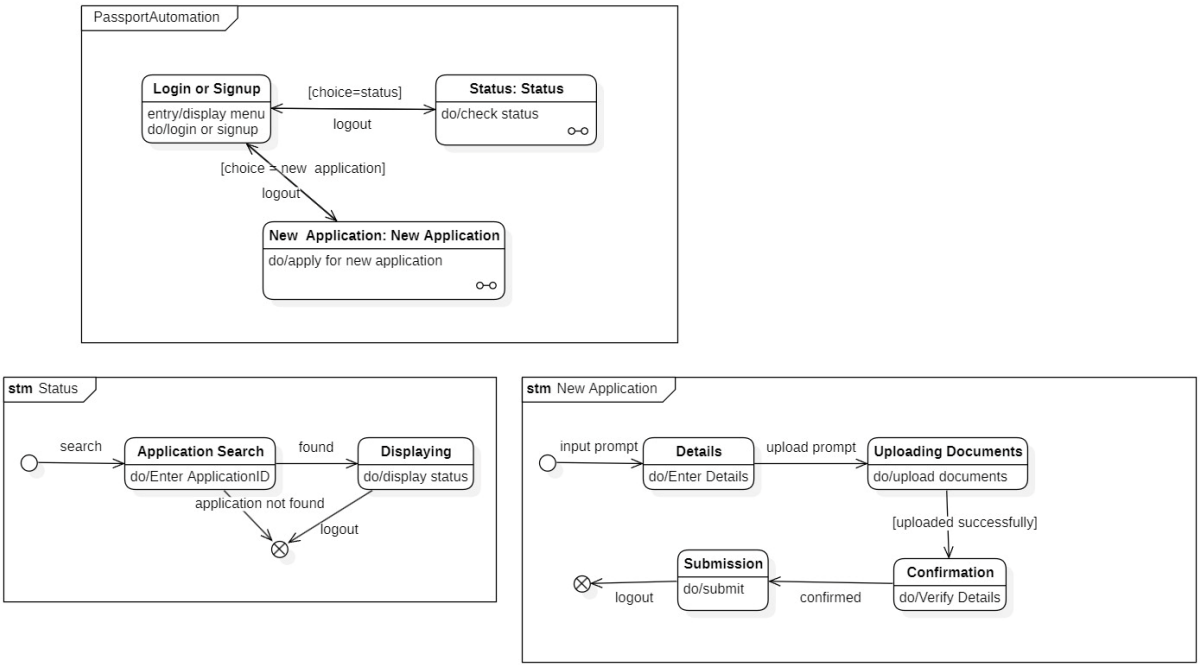


Fig 4.2: State Diagram

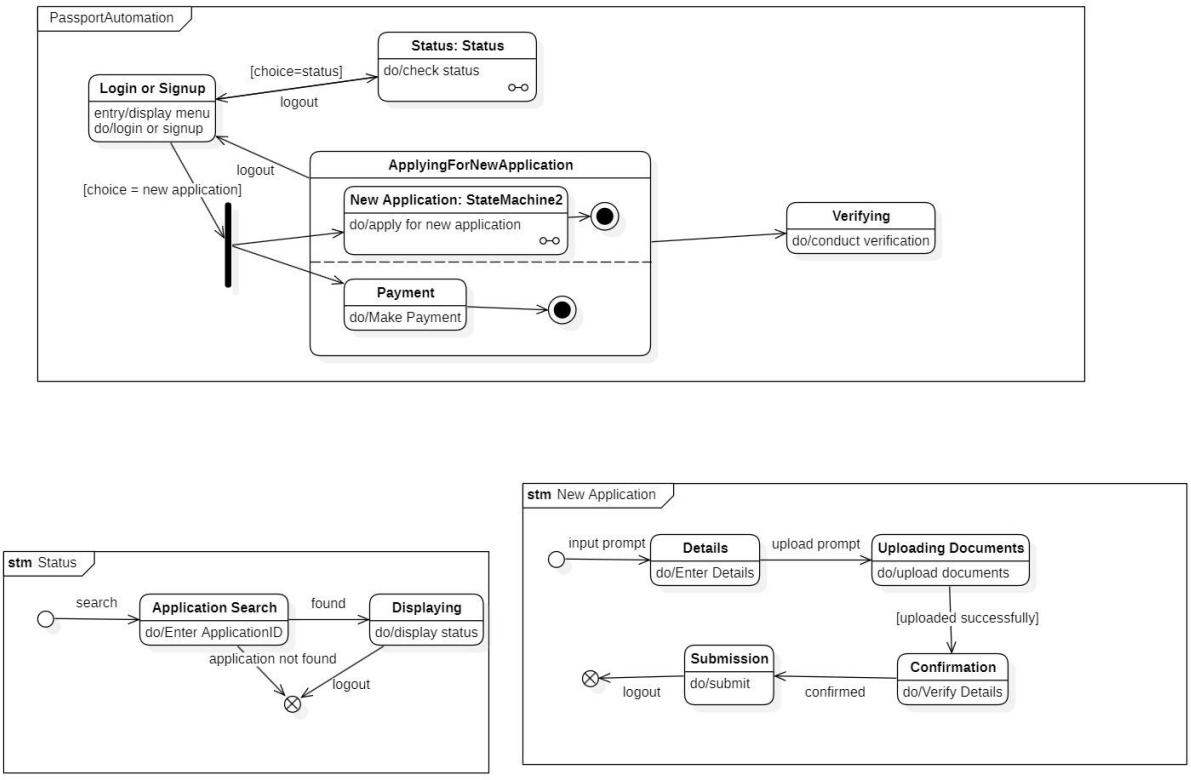


Fig 4.3: Advanced State Diagram

Description:

The state diagram outlines the transitions between different states in the passport automation system.

- **States:**
    - **Login State**: Represents user authentication into the system.
    - **Status State**: Enables applicants to check their application status.
    - **Verifying State**: Manages the verification of applicant details by authorities.
    - **Applying for New Application State**: Handles the process of submitting a new application.
        - **Concurrent States**:
            - **New Application**: Submits applicant details and required documents.
            - **Payment**: Manages the payment of fees for the application.
        - **Submachines**:
            - **New Application**: Includes steps like entering details and uploading documents.
            - **Status**: Tracks the progress of the application.

4.4 Use Case Diagram:

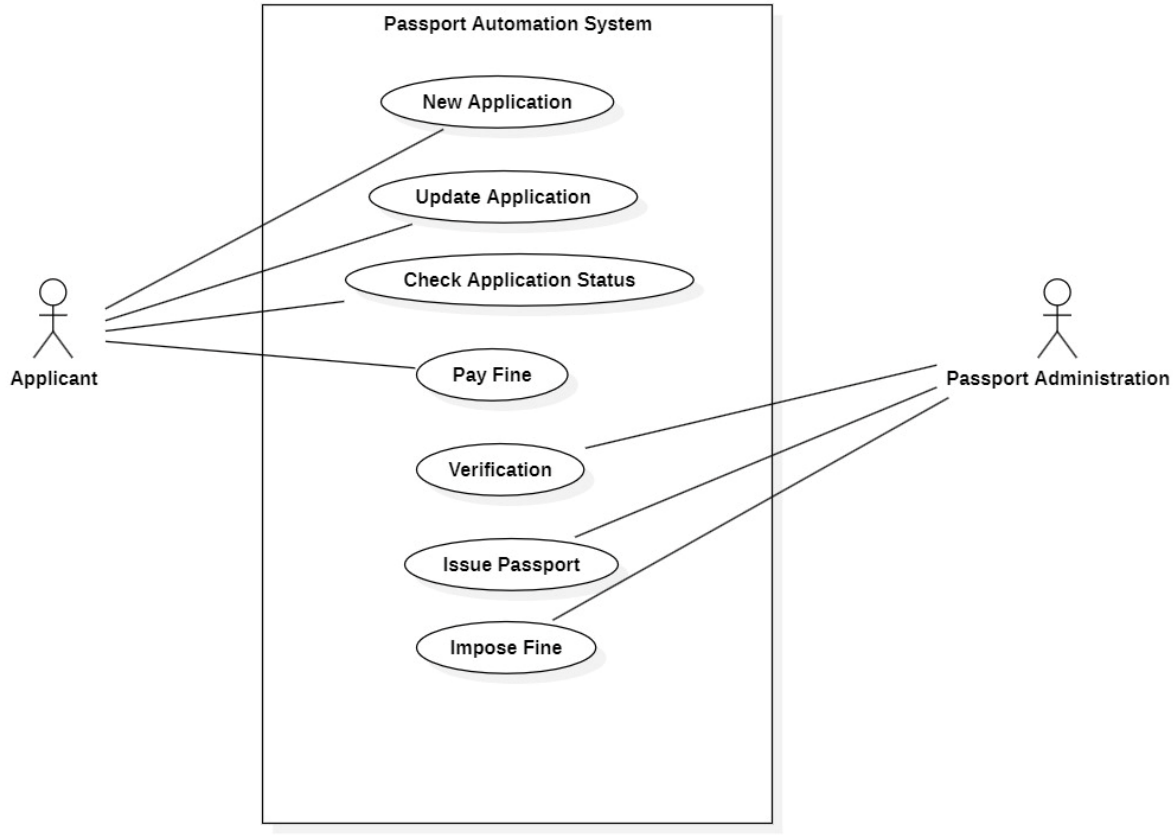


Fig 4.4: Use Case Diagram

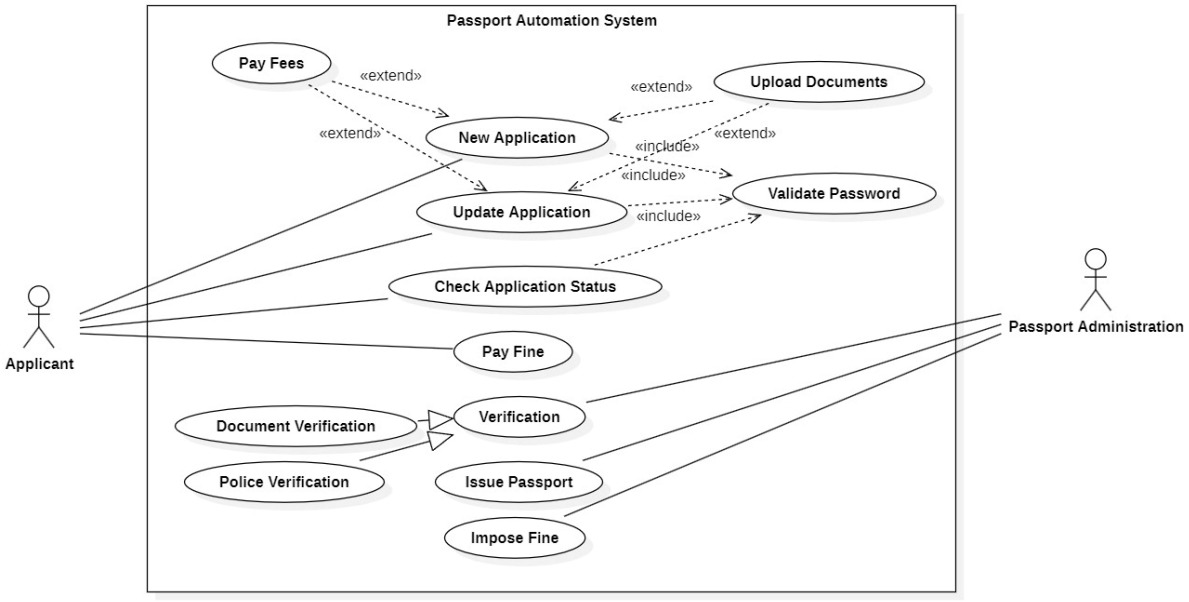


Fig 4.5: Advanced Use Case Diagram

Description:

The use case diagram illustrates interactions between users (actors) and system functionalities.

- **Actors:**
    - **Applicant**: Submits applications, checks status, and pays fines.
    - **Passport Admin**: Processes applications and imposes fines.
- **Use Cases:**
    - **New Application**: Submit a fresh application for a passport.
    - **Update Application**: Make changes to an ongoing or existing application.
    - **Check Application Status**: Track the progress of an application.
    - **Verification**: Authorities verify applicant details.
    - **Pay Fine**: Handles fines for late renewals or errors.
    - **Issue Passport**: Final step where the passport is issued to the applicant.

Impose Fine: Admin imposes penalties for discrepancies or delays.

4.5 Sequence Diagram:

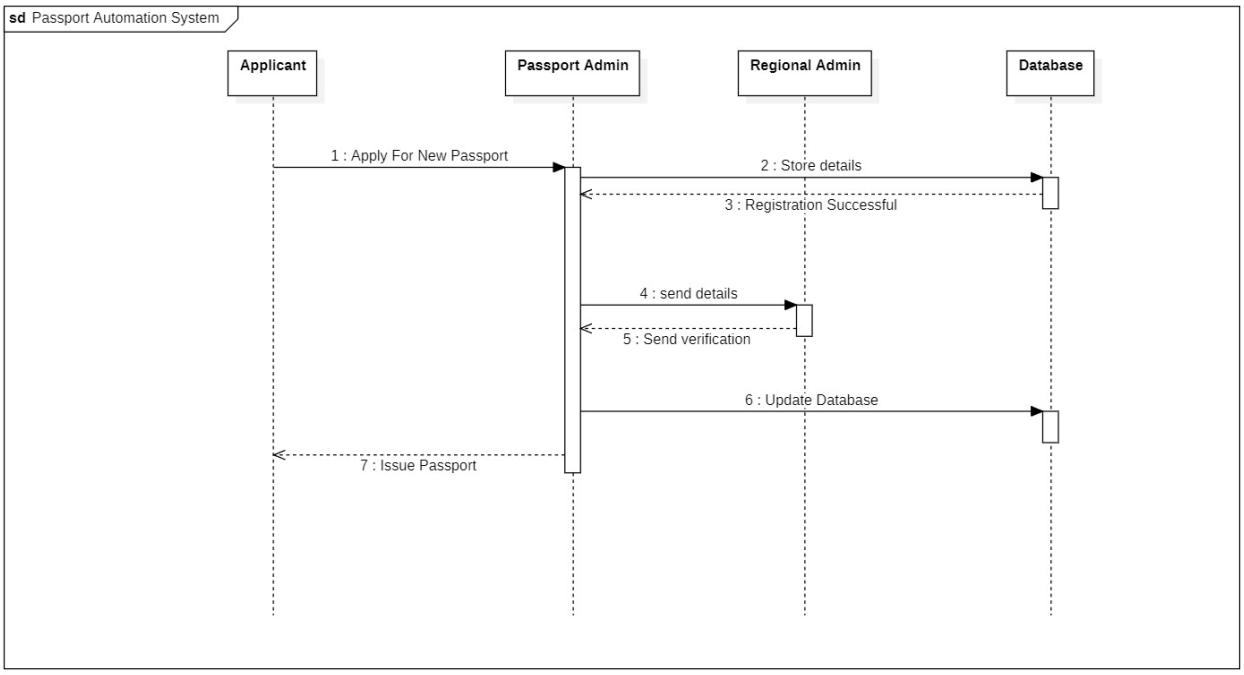


Fig 4.6: Sequence Diagram

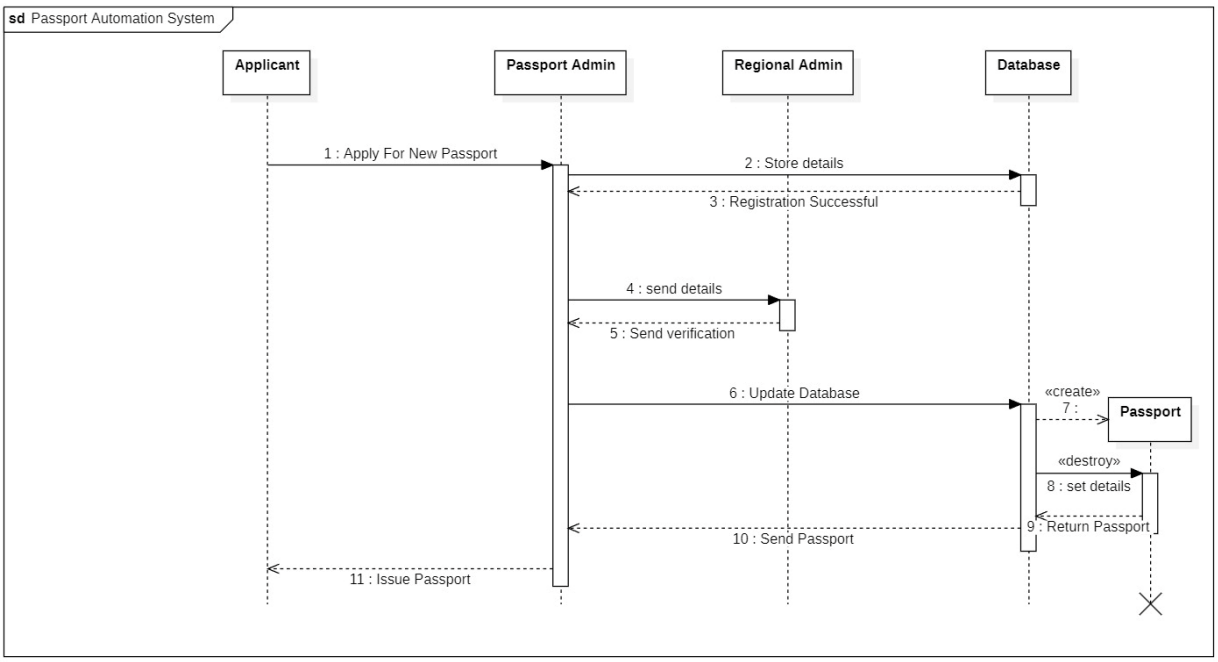


Fig 4.7: Advanced Sequence Diagram

Description:

The sequence diagram showcases the flow of interactions during the application process.

- **Objects:**
    - **Applicant**: Initiates the application and provides details.
    - **Passport Admin**: Processes the application and coordinates verification.
    - **Regional Admin**: Handles escalations or approvals for specific cases.
    - **Database**: Stores and retrieves application and verification information.
    - **Passport (Transient Object)**: Temporarily represents the passport being issued.
- **Flow:**

1. Applicant submits application details to the Passport Admin.
2. Passport Admin stores the application in the Database and forwards it for verification.
3. Regional Admin oversees the verification process and approves/rejects the application.
4. Upon approval, the Database updates the application status and generates the Passport object.
5. Applicant is notified of the issuance.

4.6 Activity Diagram

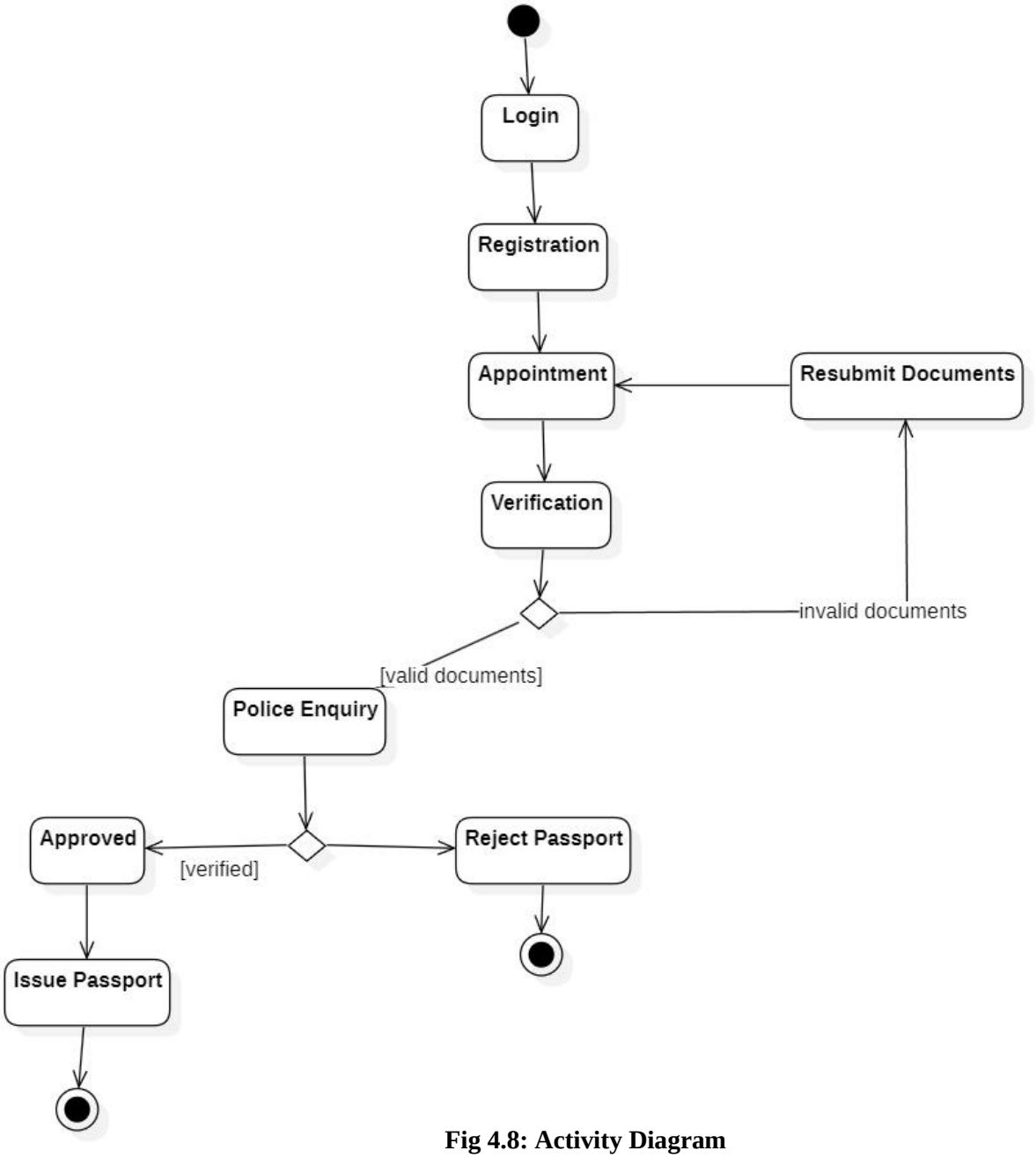


Fig 4.8: Activity Diagram

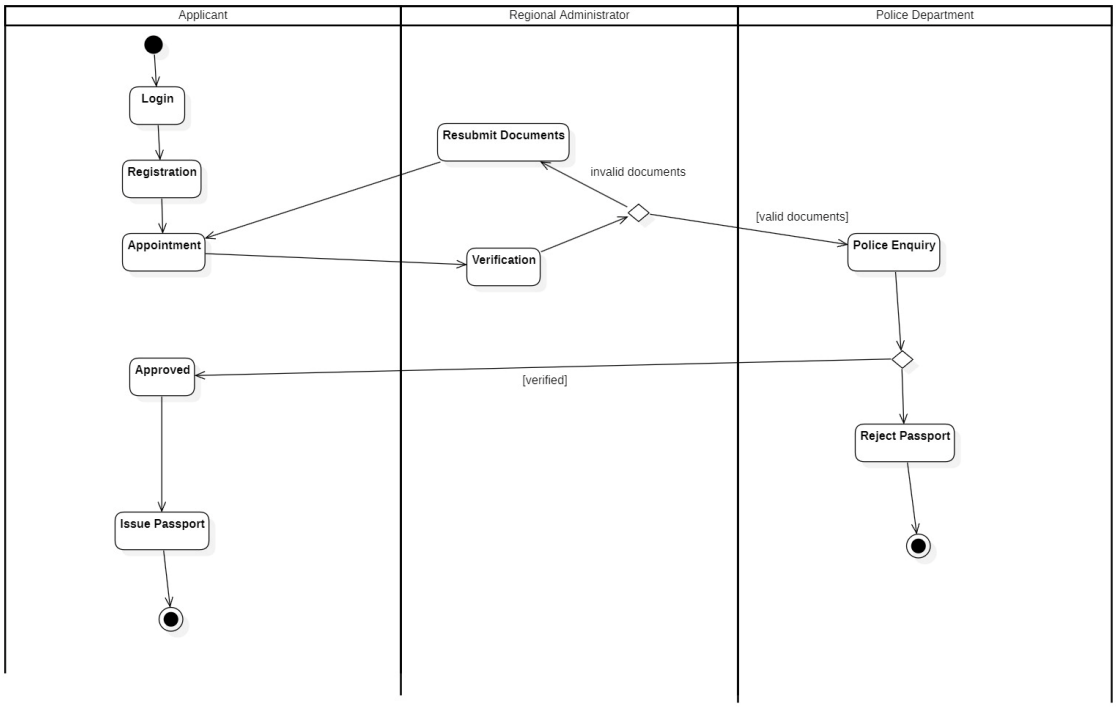


Fig 4.9: Advanced Activity Diagram

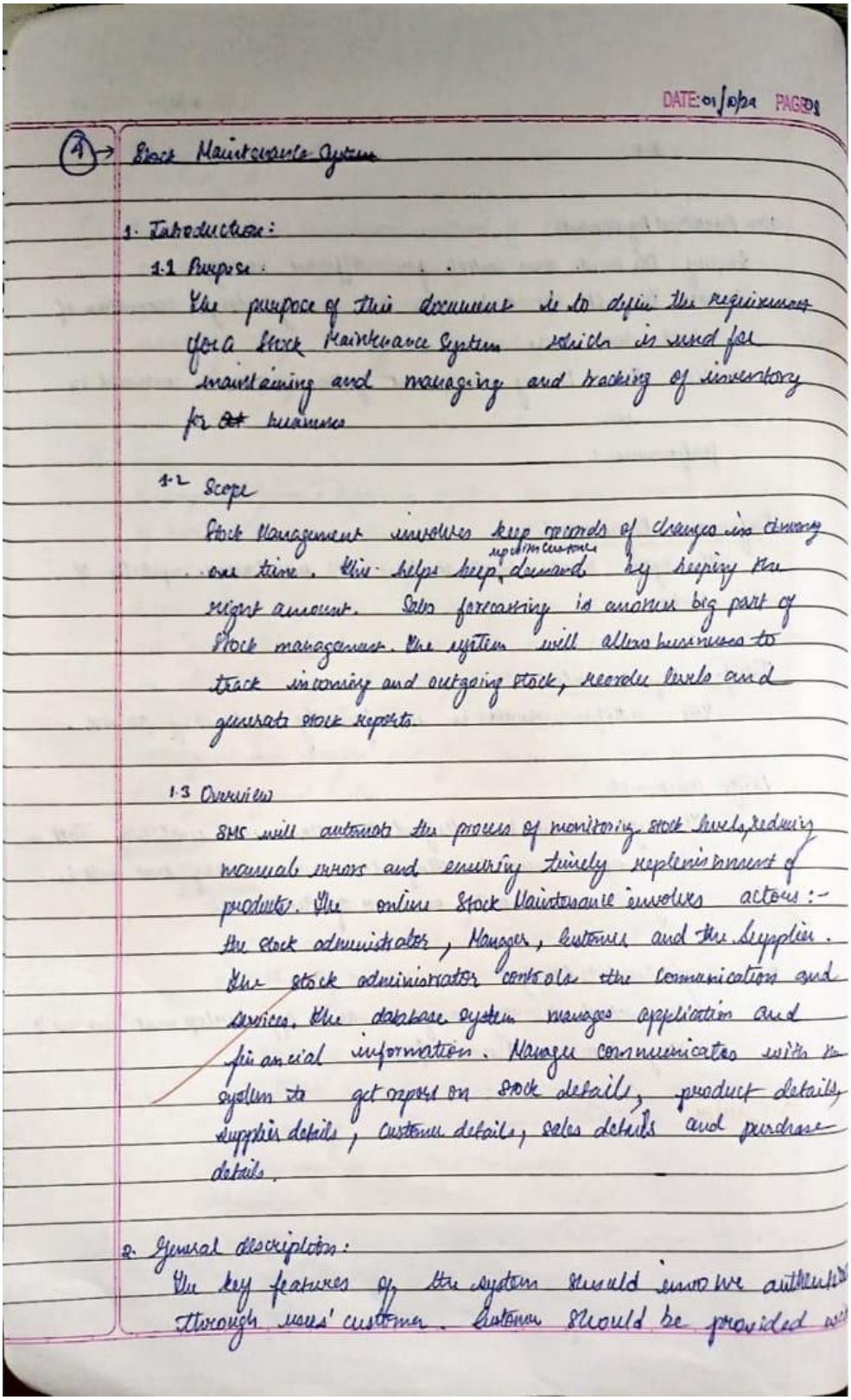
The activity diagram outlines the workflow for processing a passport application, divided into swimlanes for clarity.

- **Swimlanes:**
    - **Applicant**: Submits application, provides documents, and pays fees.
    - **Regional Administrator**: Verifies the application and documents.
    - **Police Department**: Conducts background checks and verification.
- **Flow:**

1. Applicant logs into the system, submits the application, uploads documents, and makes payment.
2. The Regional Administrator verifies the documents and forwards the application to the Police Department.
3. The Police Department conducts a background check and updates the status.
4. The Regional Administrator approves the application and notifies the Applicant.
5. The passport is issued, and the Applicant is informed.

5. Stock Maintenance System

5.1 SRS:



DATE: 01/04/24 PAGE: 01

(4) → Stock Maintenance System

1. Introduction:

1.1 Purpose:

The purpose of this document is to define the requirements for a Stock Maintenance System which is used for maintaining and managing and tracking of inventory for ~~Of~~ business

1.2 Scope

Stock Management involves keep records of changes in during one time. This helps keep up-to-date demands by keeping the right amount. Sales forecasting is another big part of stock management. The system will allow businesses to track incoming and outgoing stock, reorder levels and generate stock reports.

1.3 Overview

SMS will automate the process of monitoring stock levels, reducing manual errors and ensuring timely replenishment of products. The online Stock Maintenance involves actors :- the stock administrator, Manager, Customer and the Supplier. The stock administrator controls the communications and services. The database system manages application and financial information. Manager communicates with the system to get report on stock details, product details, supplier details, customer details, sales details and purchase details.

2. General description:

The key features of the system should ensure authentic through users' customer. Customer should be provided wi

product list, from which he/she can select and buy products. Details such as Manager, Supplier details, product details, Stock details, Sales details as well as purchase order details. The supplier should be provided the feature to receive purchase orders and send invoices. The stock admin should be allowed to manage and update stock, product details and supplier details.

3. Functional Requirements: The functional requirements are as follows:
3.1 The system shall be internet oriented and require an online server.
3.2 The system shall save the product details, customer details, supplier details, purchase details and stock details in remote database.
3.3 The system shall allow to customer to log in and buy the product he sales process.
3.4 The system shall allow manager to view and print product details, customer details, Sales details.
3.5 The system shall allow to send purchase order to Supplier and receive invoices.
3.6 The system shall allow admin to update and manage product details.

4. Interface Requirements: The interface should be user friendly and convenient to use. It should consist of following components
→ Customer log-in
→ Manager log-in
→ Supplier Log-in
→ Stock : Admin-log-in
→ Product Sales Process
→ Product Purchase Process
→ Product Stock Maintenance
→ Views and Print Process

5. Performance Requirement:

5.1 Hardware Requirements:

→ 10GB RAM PC

→ i8th Processor

→ 14" color monitor

→ 120 GB HDD CPU

→ Proper Running Internet

5.2 Software Interfaces:

→ Database: Oracle

→ Operating System: Window 7 & above, Linux, Mac OS

→ language: Java, HTML, Javascript

6: Design Constraints:

6.1 Scalability: System should be scalable & should be able to handle increasing amount of stock and data

6.2 User Accessibility: The system should be accessible to user having technical skill.

6.3 Data Integrity: Ensuring accurate and consistent data is crucial

6.4 Security: Sensitive data must be protected

7 Non functional Attributes

7.1 Speed: Should have fair amount of speed while browsing through the products, selling and purchasing product

7.2 Efficiency: The software should be able to accomodate a minimum of 100000 records of stocks, customers and suppliers.

7.3 Safety Requirements: The system should provide protection of database.

7.4 Robust: Backup and Recovery of data should be provided at the fastest in case of failure of System.

DATE: 01/10/24 PAGE: 17

8 Preliminary Schedule and Budget :

8.1 Schedule :

8.1.1 Requirement Analysis - 2 weeks

8.1.2 Design Phase - 3 weeks

8.1.3 Development phase - 6 weeks

8.1.4 Testing Phase - 3 week

8.1.5 Deployment - 1 week

8.1.6 Training and Support - 2 weeks

Total duration = 17 weeks.

8.2 Preliminary Budget

8.2.1 Estimated LOC : 10000

8.2.2 Cost per LOC : 50 Rs

8.2.3 Total cost of project : 10000 × 50

= 500,000

5.2 Class Diagram:

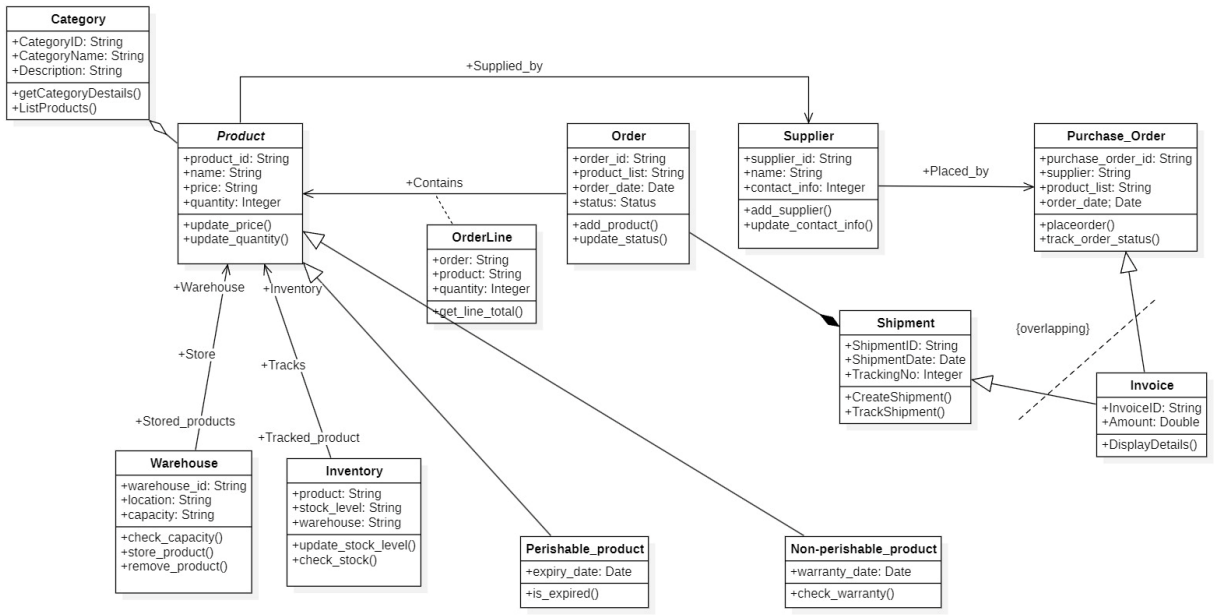


Fig 5.1: Class Diagram

Description:

This class diagram represents the key entities of the stock maintenance system and their relationships.

- **Classes:**
    - **Product Class**: Represents a product with attributes like product ID, name, price, and quantity.
        - **Perishable Product Class**: Subclass of Product with additional attributes like expiration date and storage requirements.
        - **Non-Perishable Product Class**: Subclass of Product without special storage needs.
    - **Category Class**: Represents the category of a product, with attributes like category ID and name.
    - **Warehouse Class**: Represents the storage location of products, with attributes like warehouse ID, location, and capacity.
    - **Inventory Class**: Manages stock levels and tracks product details, associating **Product** with **Warehouse**.

- **Relationships:**
    - **Product** is aggregated with **Category**, **Warehouse**, and **Inventory** to represent its categorization, storage, and stock level management.

5.3 State Diagram:

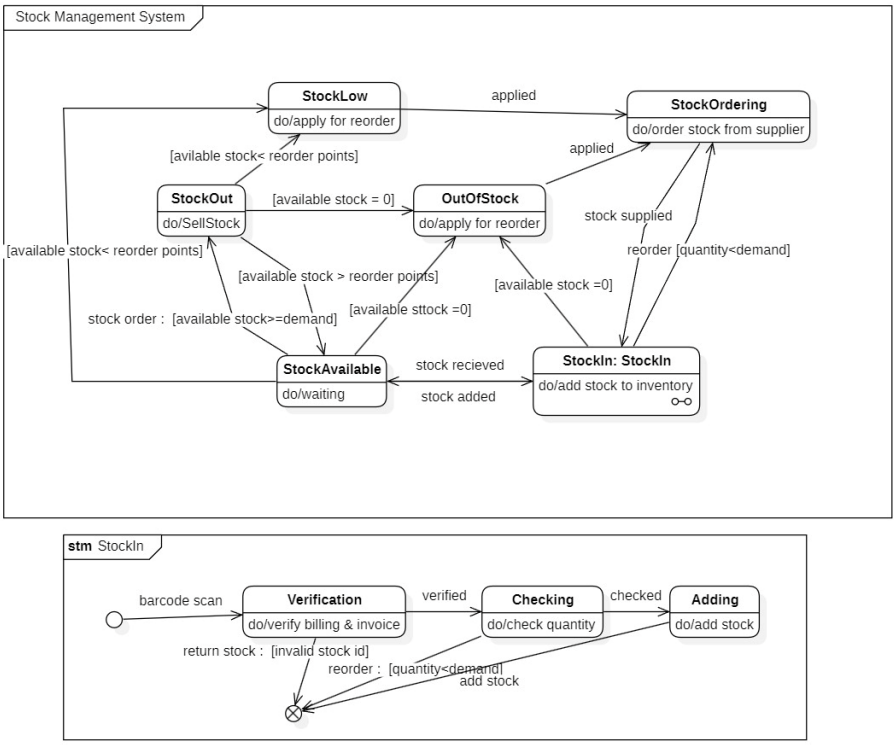


Fig 5.2: State Diagram

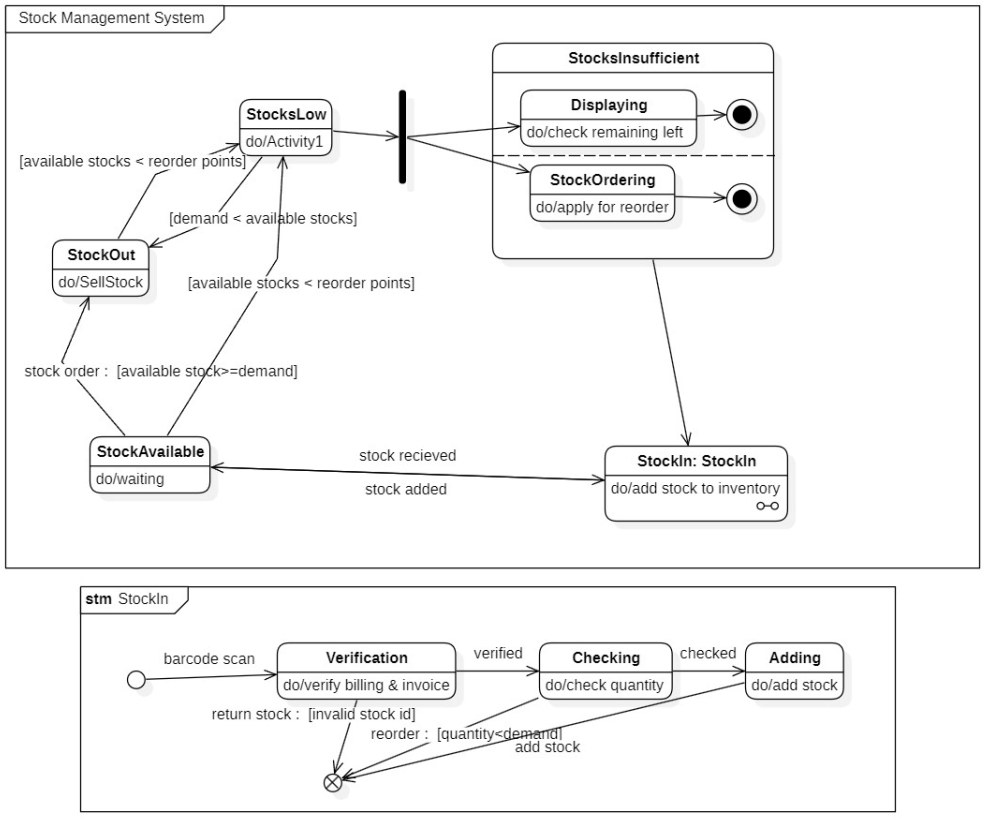


Fig 5.3: Advanced State Diagram

Description:

This state diagram highlights the transitions between various states in the stock lifecycle.

- **States:**
    - **Low Stock**: Indicates that product stock is below the reorder threshold.
    - **Stock Ordering**: Represents the process of ordering new stock.
    - **Stock Out**: Indicates that stock is being dispatched or sold.
    - **Stock In**: Represents the arrival and addition of stock to the inventory.
    - **Out of Stock**: Occurs when the product is unavailable for sale or dispatch.
    - **Stock Available**: Normal state when adequate stock is present in the inventory.

5.4 Use Case Diagram:

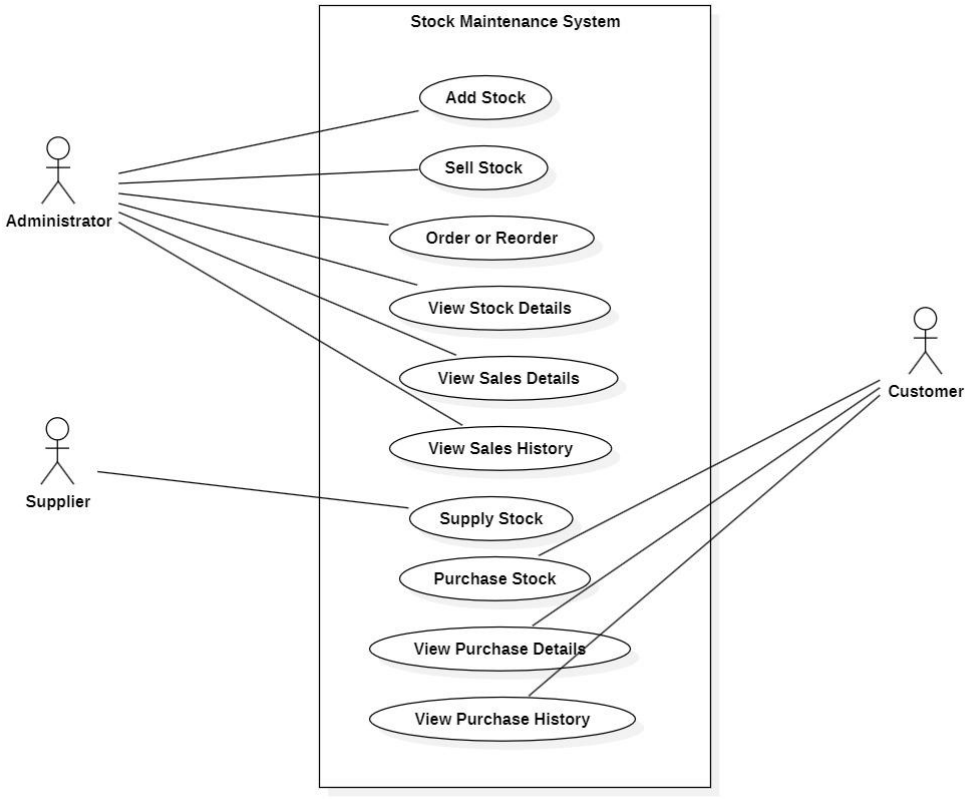


Fig 5.4: Use Case Diagram

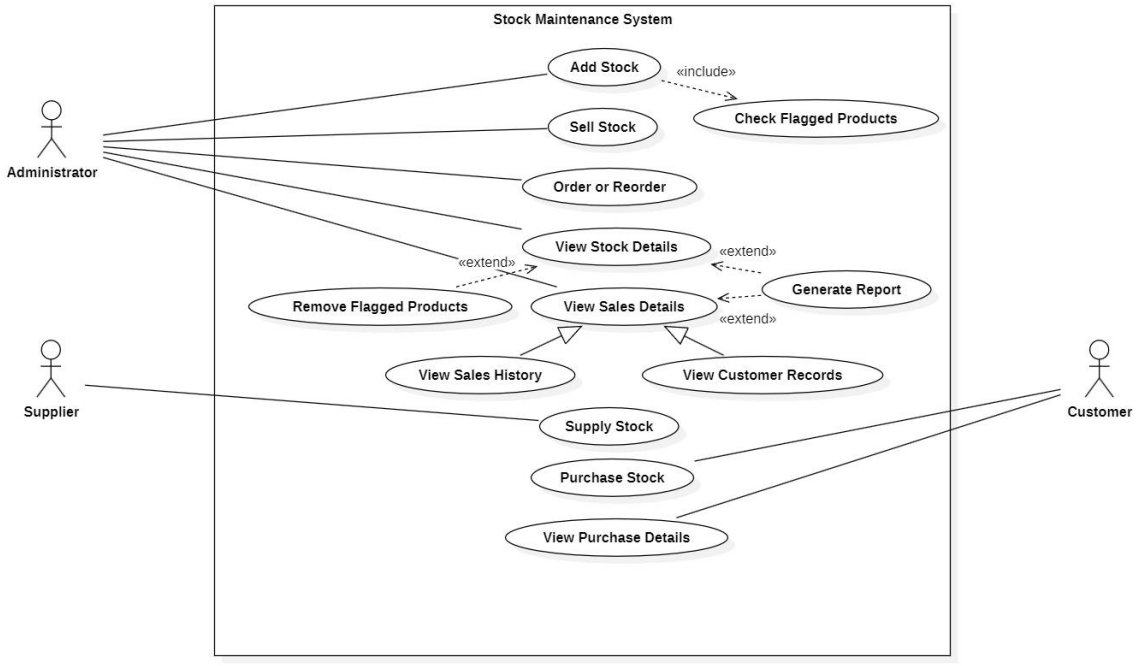


Fig 5.5: Advanced Use Case Diagram

Description:

The use case diagram illustrates the interactions between actors and system functionalities.

- **Actors:**
    - **Supplier**: Supplies stock to the warehouse.
    - **Admin**: Manages inventory and oversees operations like viewing stock and sales.
    - **Customer**: Purchases stock or products.
- **Use Cases:**
    - **Add Stock**: Admin adds new stock to the inventory.
    - **Sell Stock**: Stock is sold to customers.
    - **Order Stock**: Admin places an order for new stock with suppliers.
    - **View Stock Details**: Admin views detailed stock information, including quantities and categories.
    - **View Sales**: Admin tracks sales performance.
    - **Supply Stock**: Supplier delivers stock to the warehouse.
    - **Purchase Stock**: Customer buys products from the inventory.

5.5 Sequence Diagram:

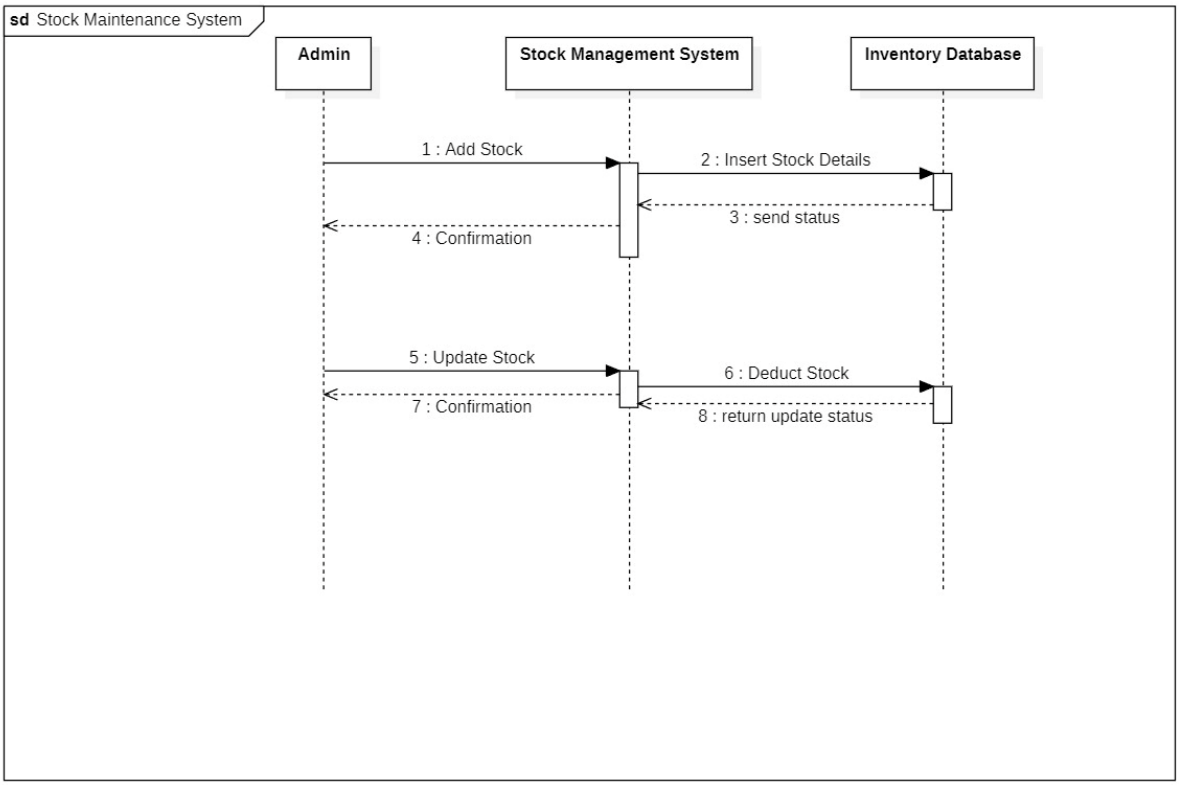


Fig 5.6: Sequence Diagram

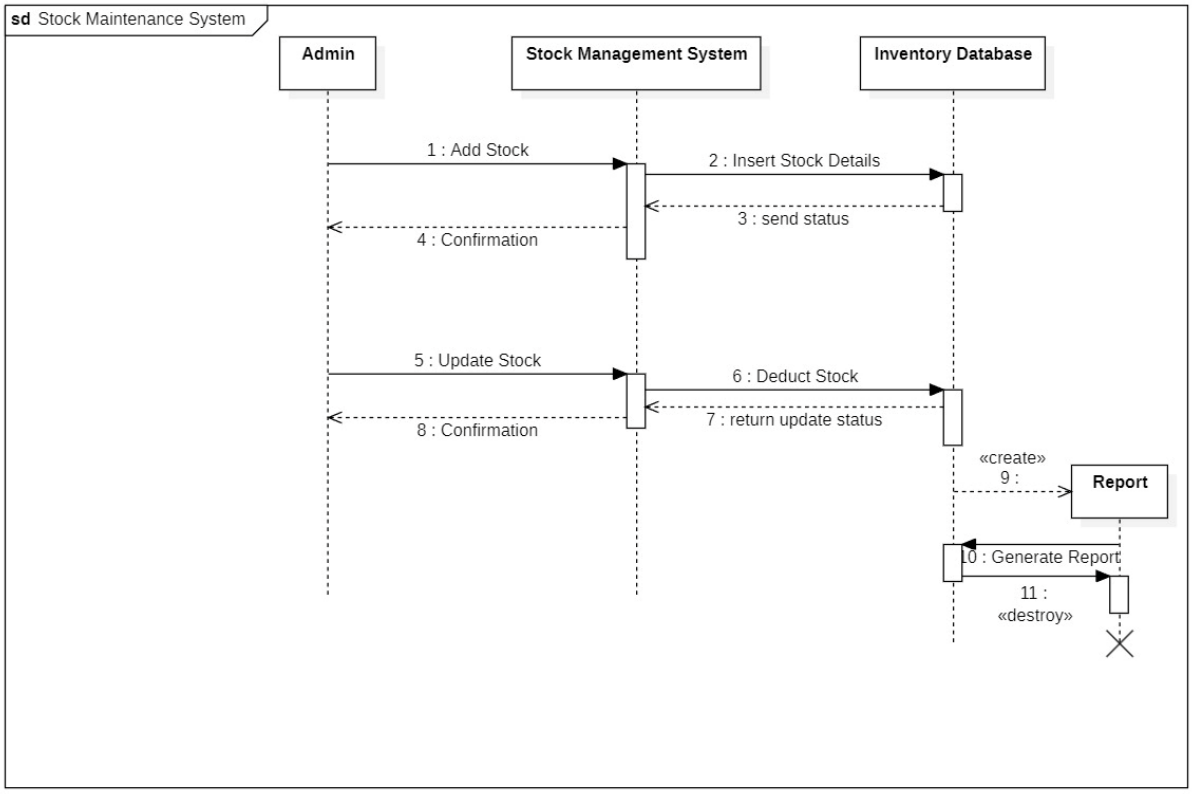


Fig 5.7: Advanced Sequence Diagram

Description:

The sequence diagram illustrates the flow of interactions for generating a stock report.

- **Objects:**
    - **Admin**: Initiates the request for stock management or reporting.
    - **Stock Maintenance System**: Processes requests related to inventory.
    - **Inventory Database**: Stores all stock and transaction details.
    - **Report (Transient Object)**: Temporarily represents the generated stock or sales report.
- **Flow:**

1. Admin requests a stock or sales report via the system.
2. Stock Maintenance System queries the Inventory Database for relevant data.
3. Inventory Database retrieves the requested information.
4. Stock Maintenance System generates a transient **Report** object.
5. Admin views or exports the report.

5.6 Activity Diagram

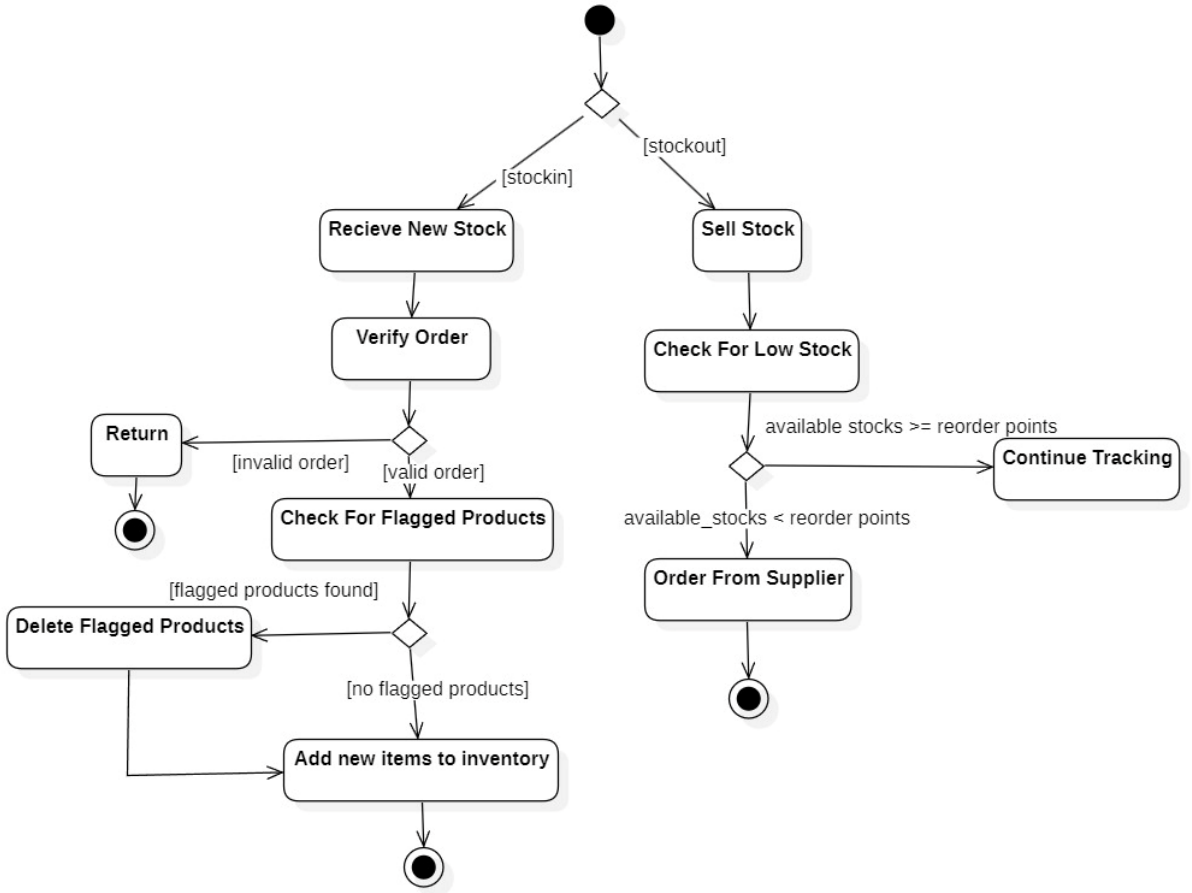


Fig 5.8: Activity Diagram

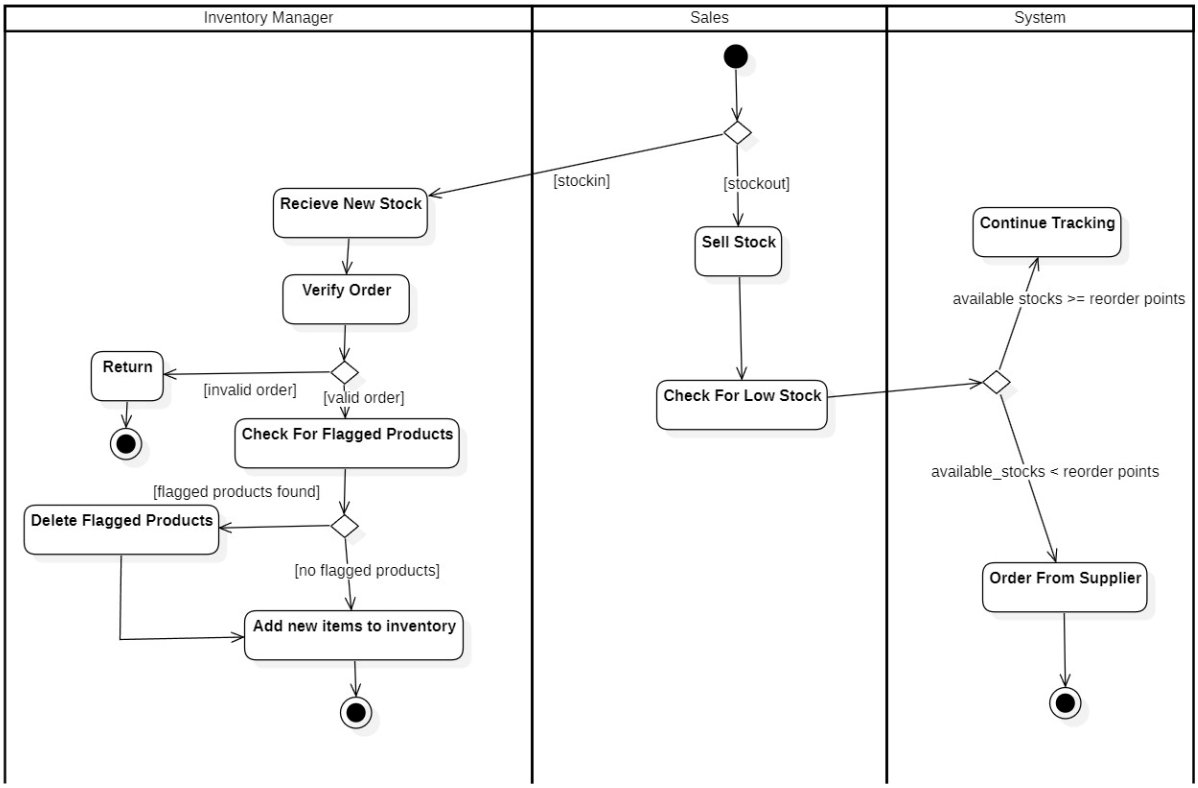


Fig 5.9: Advanced Activity Diagram

Description:

The activity diagram represents the workflow for managing inventory and sales.

- **Swimlanes:**
    - **Inventory Manager**: Handles stock updates, reorders, and inventory monitoring.
    - **Sales**: Processes sales transactions and updates stock levels.
    - **System**: Automates stock monitoring, order placements, and report generation.
- **Flow:**

1. Inventory Manager logs into the system and views stock levels.
2. If stock is low, a reorder is placed, and new stock is added upon arrival.
3. Sales transactions are processed, reducing stock in the inventory.
4. System generates sales and stock reports, which the Inventory Manager reviews.