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:

Mohith Jain

1BM22CS162

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
September-January 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(22CS6PCSEO) laboratory has been carried out by Mohith Jain (1BM22CS162) during the 5th Semester Sep 2024 - Jan 2025.

Signature of the Faculty Incharge:

Latha N R
Asst. Professor
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

1. Hotel Management System

SRS-Software Requirements Specification

1. Introduction

1.1 Purpose of this Document
This document outlines the software requirements for the Hotel Management System. Provides a clear understanding of the functionalities, design constraints, & overall goals of the software.

1.2 Scope of this document
This document details the objectives, features, and user requirements for the Hotel Management System addressing advanced development costs and timeline.

1.3 Overview
The Hotel Management System will provide a solution for managing reservations, check-ins, billing, and customer service. This system will ensure greater efficiency and improve customer satisfaction.

2. General Description

- The objectives of the system are to simplify reservation management, enhance customer experience, and manage check-in/check-out processes.
- The characteristics of the users: front desk staff, hotel managers, and customers.
- This system provides features as room booking, payment processing, customer management & reporting tools.
- This system is essential for modern hotels to remain efficient in service delivery.

3. Functional Requirements

1. Room Booking: Allow users to search for available rooms and book online.
2. Check-ins/Check-outs: Facilitate easy check-in & check-out processes.
3. Payment Processing: Handle payments via various methods (e.g., credit card, cash, UPI).
4. Customer Management: Maintain customer records and preferences.
5. Reports: Generate reports on occupancy, revenue & customer feedback.

4. Non-functional Requirements

1. User Interface: Must be simple and adaptable for staff & mobile interfaces via application for customers.

2. API Integration: Integrate with third-party services for payment gateways & other systems.
3. Data: Real-time updates on room availability.

5. Performance Requirements

1. Response Time: The system should respond to user requests within 2 seconds.
2. Availability: 100% uptime for critical services.
3. Scalability: Must handle up to 1000 concurrent users.

6. Design Constraints

1. Must be compatible with existing hotel management software.
2. Should comply with data protection regulations.

7. Non-functional Attributes

1. Security: The data must be protected against unauthorized access.
2. Usability: Intuitive for staff and customers.
3. Maintainability: Scalable and upgradable.

8. Schedule & Budget

- Time Estimation: 6 months to 8 months
- Budget: $10,000

Class Diagram

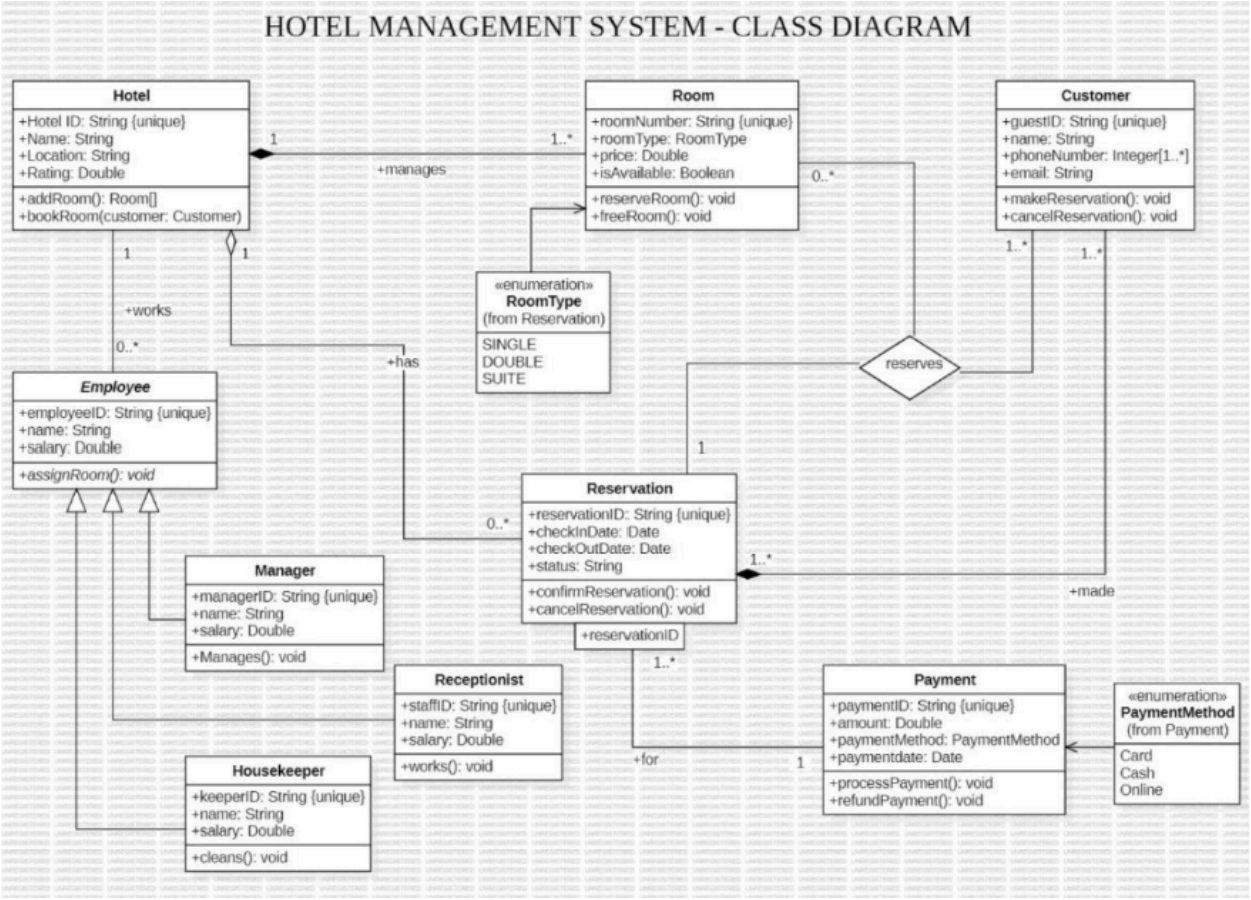

Fig 1.1: Hotel Management System Class Diagram

1. Hotel:
    - Attributes: hotelID, name, location, rating.
    - Methods: addRoom(), bookRoom().
    - Relationships: Manages multiple Room and Employee instances.
2. Room:
    - Attributes: roomNumber, roomType (SINGLE, DOUBLE, SUITE), price, isAvailable.
    - Methods: reserveRoom(), freeRoom().
    - Relationships: Associated with a Hotel and linked to Reservation.
3. Customer:
    - Attributes: guestID, name, phoneNumber, email.
    - Methods: makeReservation(), cancelReservation().

- Relationships: Creates multiple Reservations.

4. Reservation:
    - Attributes: reservationID, checkInDate, checkOutDate, status.
    - Methods: confirmReservation(), cancelReservation().
    - Relationships: Links Customer, Room, and Payment.
5. Payment:
    - Attributes: paymentID, amount, paymentMethod (Card, Cash, Online), paymentDate.
    - Methods: processPayment(), refundPayment().
    - Relationships: Linked to Reservation.
6. Employee:
    - Attributes: employeeID, name, salary.
    - Subtypes: Manager, Receptionist, Housekeeper.
    - Methods: Vary by role (e.g., assignRoom(), cleans()).

Relationships

- Customers reserve rooms and make payments.
- Employees work for a hotel and handle roles like management, reception, and housekeeping.
- Reservations associate rooms, customers, and payments.

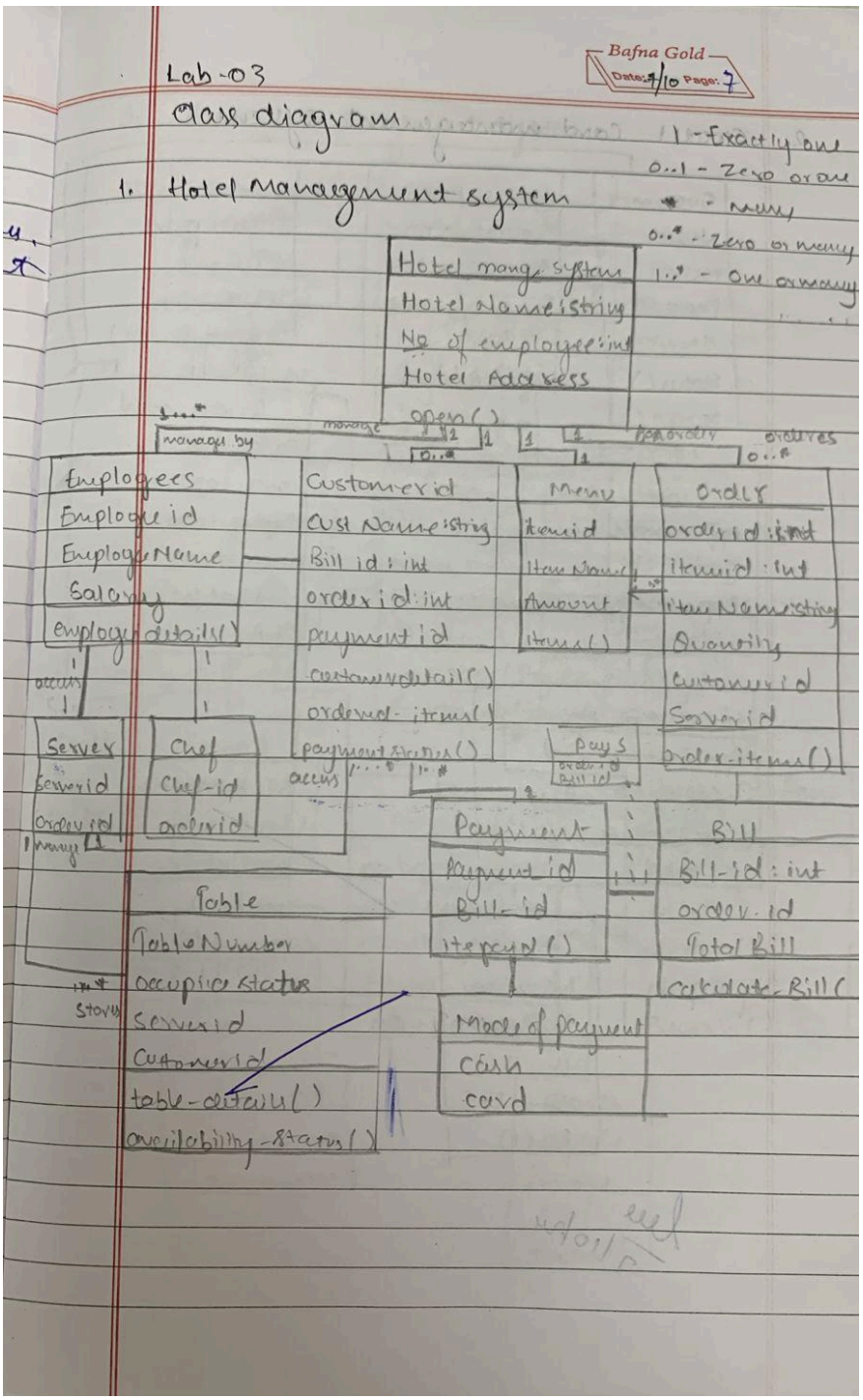

Fig 1.2: Hotel Management System Class Diagram Observation

State Diagram

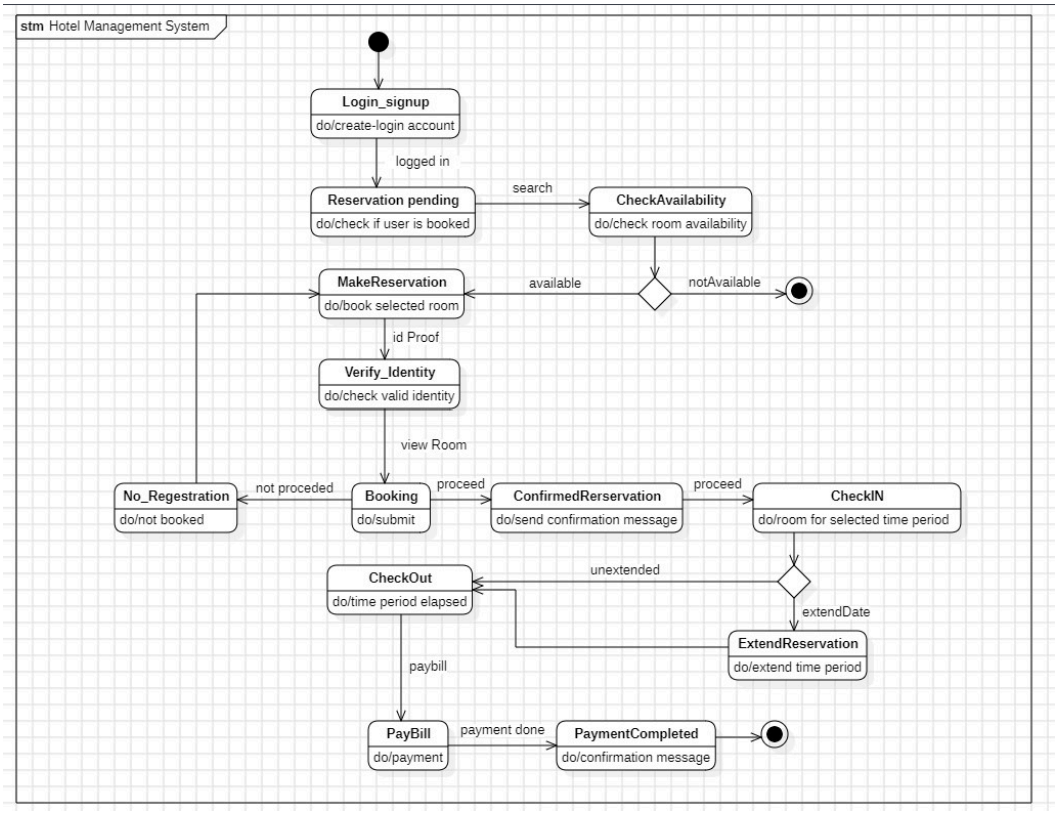

Fig 1.3: Hotel Management System Simple State Diagram

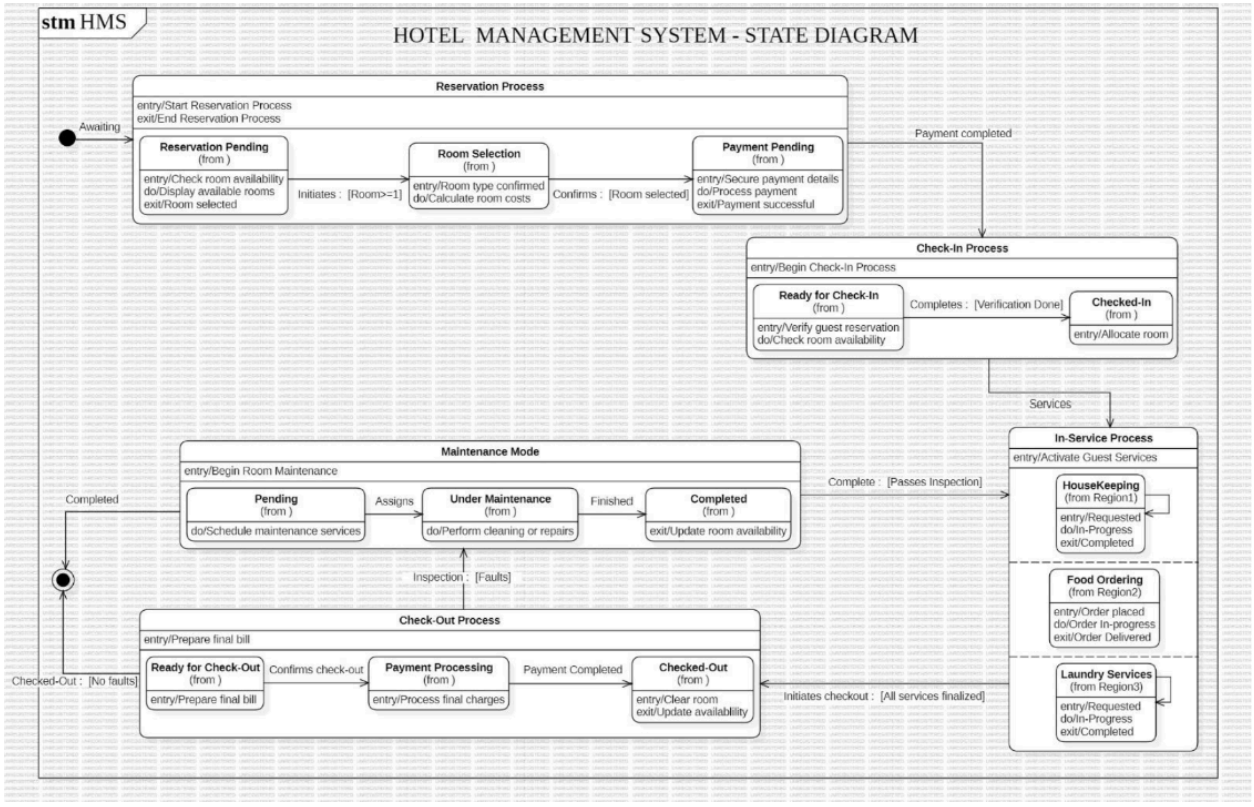

Fig 1.4: Hotel Management System Advanced State Diagram

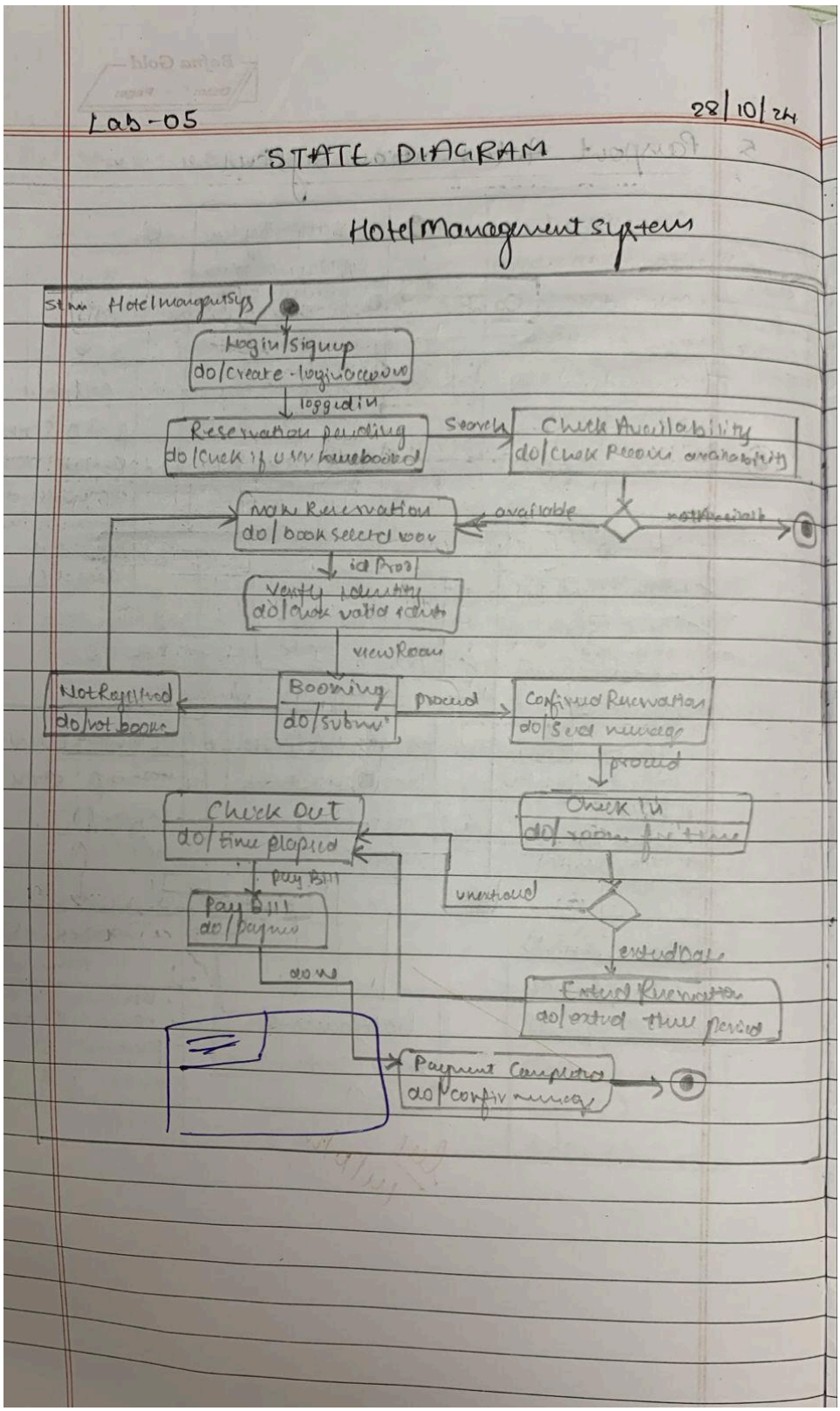

Fig 1.5: Hotel Management System Advanced State Diagram Observation

1. Reservation Process:
    - Reservation Pending: The system begins by checking room availability and displaying options.
    - Room Selection: Customers confirm a room type, and costs are calculated.
    - Payment Pending: Payment details are secured and processed. The process ends upon successful payment.
2. Check-In Process:
    - Ready for Check-In: The system verifies the guest's reservation and room availability.
    - Checked-In: Rooms are allocated after successful verification.
3. In-Service Process:
    - Includes guest services such as:
        - Housekeeping: Cleaning requests are managed and completed.
        - Food Ordering: Orders are placed, processed, and delivered.
        - Laundry Services: Laundry requests are handled and completed.
4. Maintenance Mode:
    - Pending: Maintenance tasks are scheduled.
    - Under Maintenance: Repairs or cleaning are performed.
    - Completed: Room availability is updated once maintenance is finished.
5. Check-Out Process:
    - Ready for Check-Out: Final billing is prepared.
    - Payment Processing: Final charges are processed.
    - Checked-Out: The system clears the room and updates its availability.

Key Features:

- Entry/Exit Conditions: States specify entry, exit, and ongoing actions (e.g., checking availability, processing payments).
- Transitions: Arrows represent transitions triggered by events (e.g., payment completion, verification done).
- Concurrent States: Multiple services (housekeeping, food ordering, laundry) can occur simultaneously.

Use Case Diagram

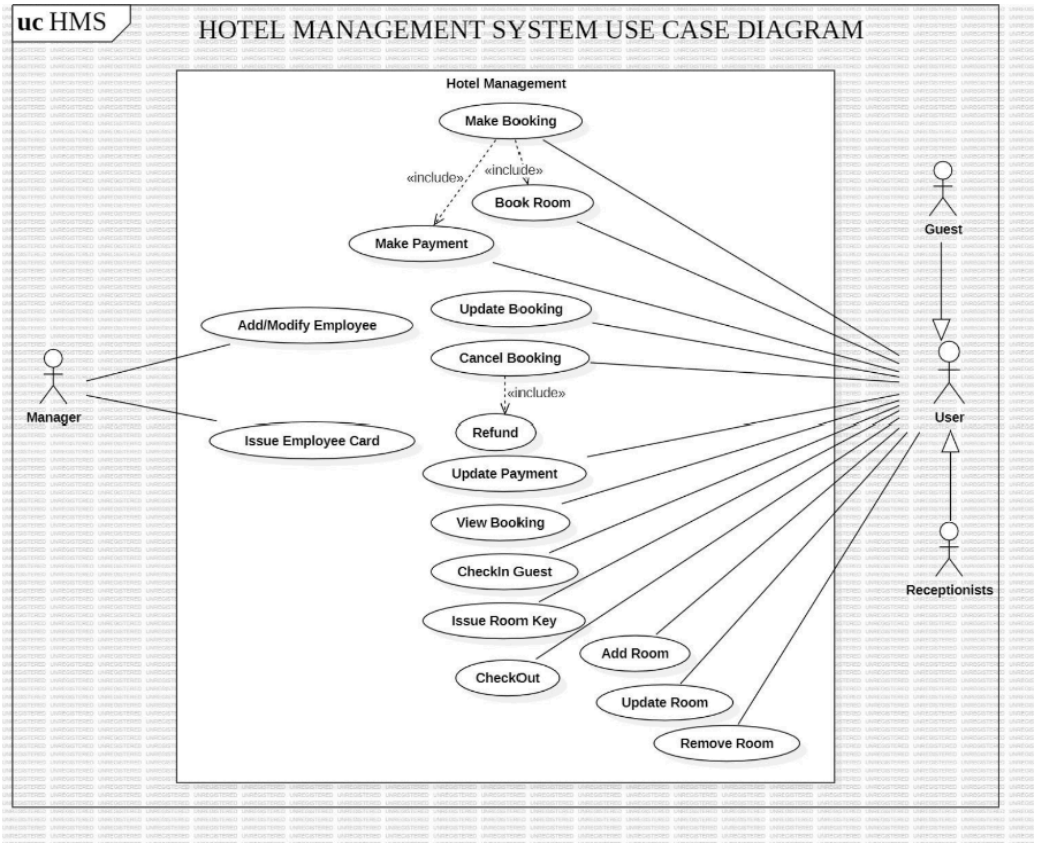

Fig 1.6: Hotel Management System Use Case Diagram

Actors:

1. Manager:
    - Responsible for managing employees and overseeing hotel operations.
2. Receptionists:
    - Handle bookings, check-ins, check-outs, and room assignments.
3. Guest:
    - The end-user who makes bookings, payments, and utilizes services.
4. User:
    - General user role, representing staff and customers interacting with the system.

Use Cases:

1. Hotel Management:
    - Central process encompassing all management functionalities.

2. Make Booking:
    - Includes the process of booking a room and processing payment.
3. Book Room:
    - Part of the booking process where users select a room.
4. Make Payment:
    - Involves processing payments for bookings or other services.
5. Update Booking:
    - Modify details of an existing reservation.
6. Cancel Booking:
    - Cancel an existing booking; includes issuing refunds if applicable.
7. Refund:
    - Manage and process refunds for canceled bookings or other reasons.
8. Update Payment:
    - Adjust or record payment details.
9. View Booking:
    - Allows users to view their reservation details.
10. Check-In Guest:
    - Receptionists manage the check-in process for arriving guests.
11. Issue Room Key:
    - Provide guests with keys or access credentials for their rooms.
12. Check-Out:
    - Manage the guest's departure, including payment settlement and updating room availability.
13. Add/Modify Employee:
    - Manager adds or updates employee records in the system.
14. Issue Employee Card:
    - Manager issues identification cards or access credentials to employees.
15. Add Room:
    - Receptionists or managers add new rooms to the system.
16. Update Room:
    - Modify room details, such as status, price, or features.
17. Remove Room:

- Remove rooms from the system (e.g., during maintenance or renovations).

Relationships:

1. Includes:
   - Denoted by <<include>>, showing dependent use cases.
   - Example: "Make Booking" includes "Book Room" and "Make Payment".
2. Direct Associations:
   - Lines connect actors to the use cases they interact with.
   - Example: Guests interact with "Make Booking," "Cancel Booking," and "View Booking."

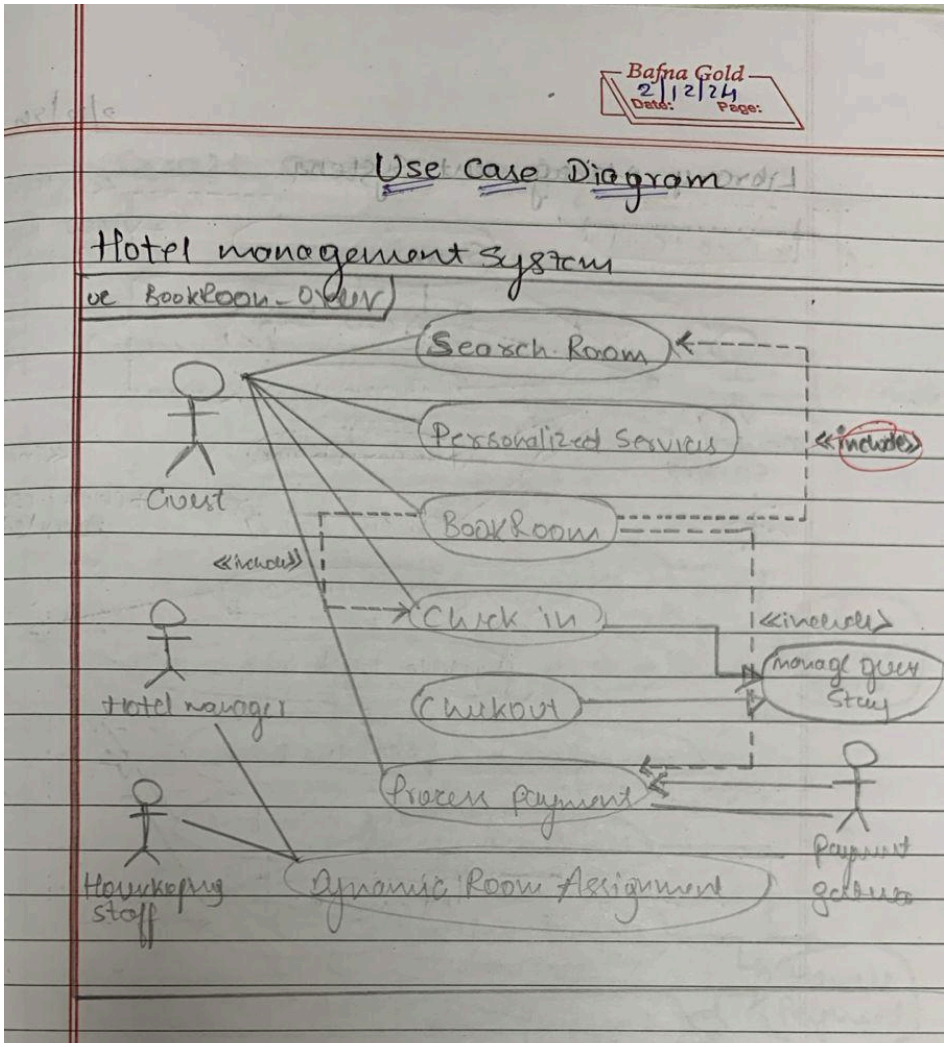

Fig 1.7: Hotel Management System Use Case Diagram Observation

Sequence Diagram

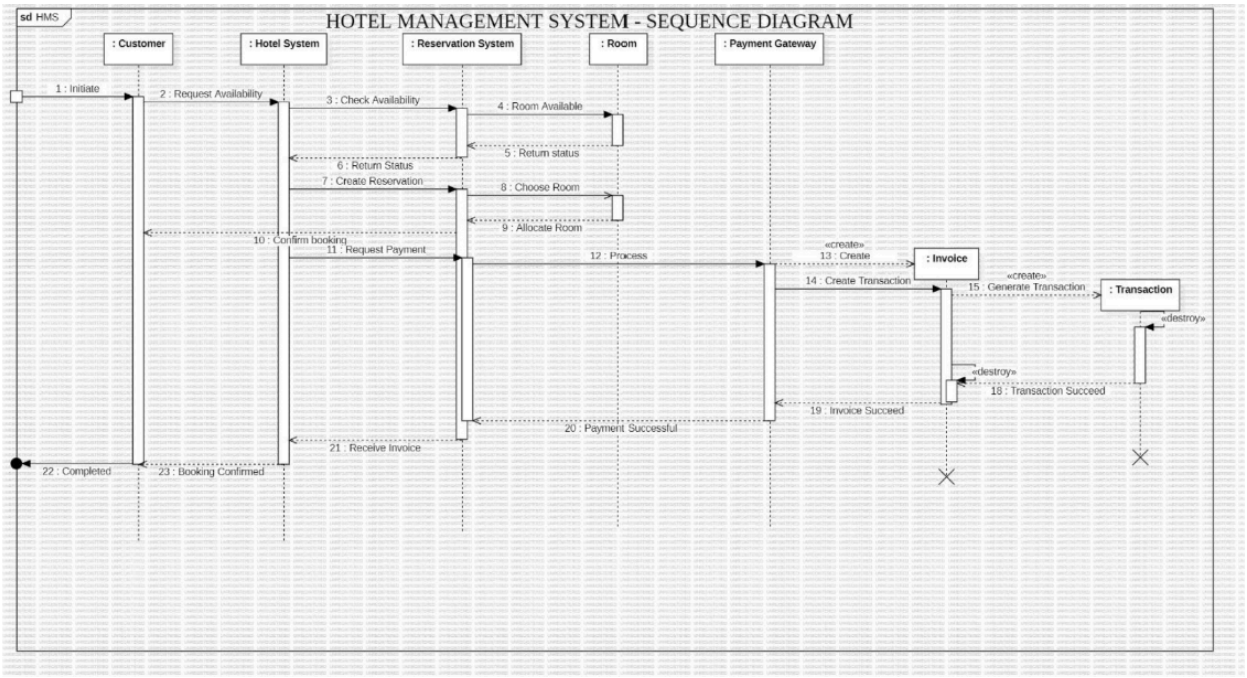

Fig 1.8: Hotel Management System Advanced Sequence Diagram

1. Customer Initiates: The customer starts the booking process.
2. Request Availability: The customer sends a request to the Hotel System to check room availability.
3. Check Availability: The Hotel System forwards the request to the Reservation System.
4. Room Available: The Reservation System checks for available rooms.
5. Return Status: The Reservation System returns the availability status to the Hotel System.
6. Return Status: The Hotel System informs the customer about the availability.
7. Create Reservation: If available, the Reservation System creates a tentative reservation.
8. Choose Room: The Reservation System assigns or allows the customer to select a room.
9. Allocate Room: The Reservation System finalizes the room allocation.
10. Confirm Booking: The customer confirms the booking.
11. Request Payment: The Hotel System requests payment from the Payment Gateway.
12. Process: The Payment Gateway begins processing the payment.
13. Create Transaction: A transaction instance is created to handle the payment.
14. Generate Transaction: The transaction process starts (e.g., deducting the amount).
15. Transaction Success: Upon successful payment, the Payment Gateway confirms the transaction.

16. Invoice Succeed: An invoice is created and returned.
17. Receive Invoice: The Hotel System receives the invoice from the Payment Gateway.
18. Booking Confirmed: The Hotel System confirms the booking and informs the customer.
19. Completed: The entire process is successfully completed.

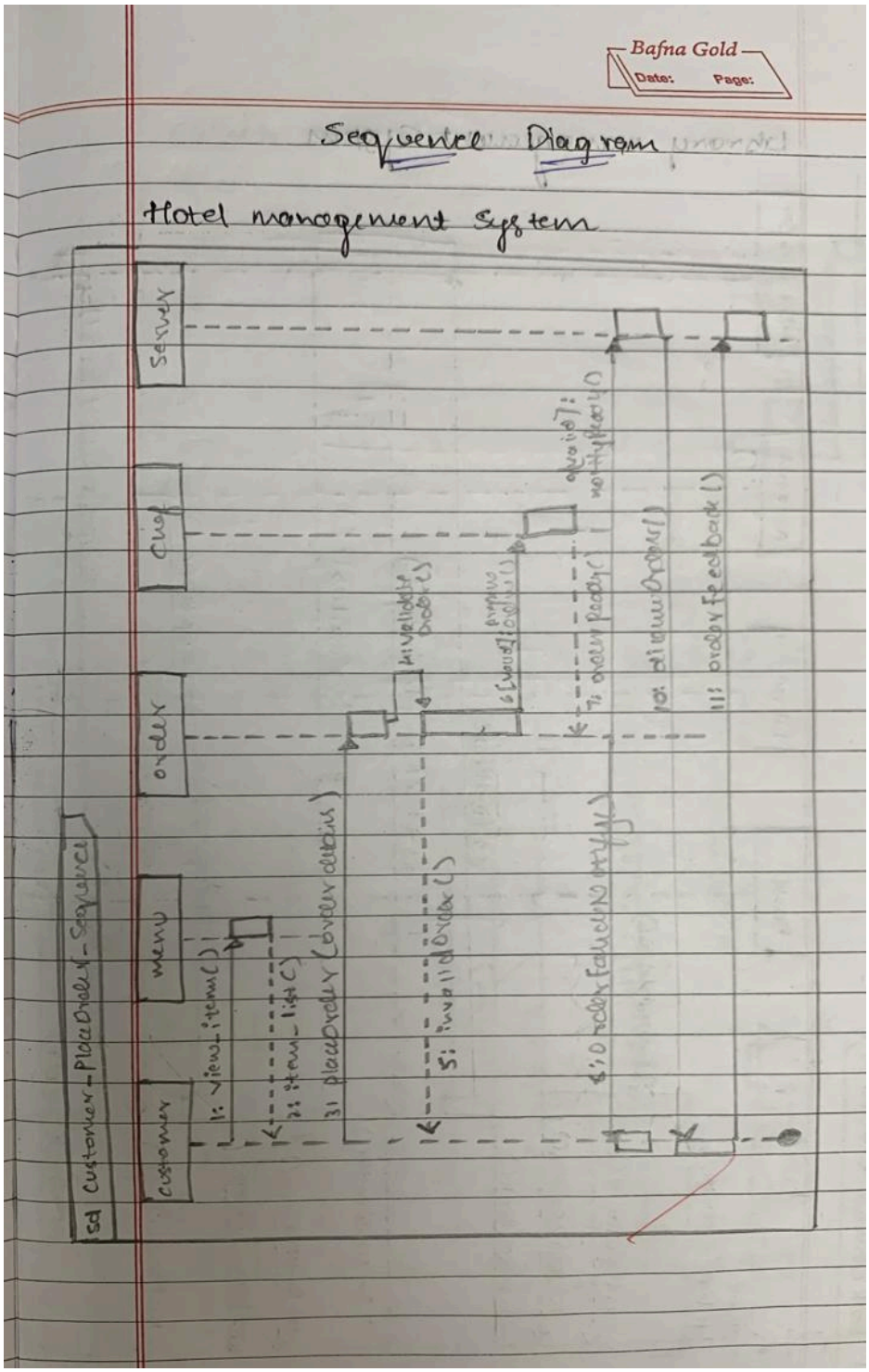

Fig 1.9: Hotel Management System Advanced Sequence Diagram Observation

Activity Diagram

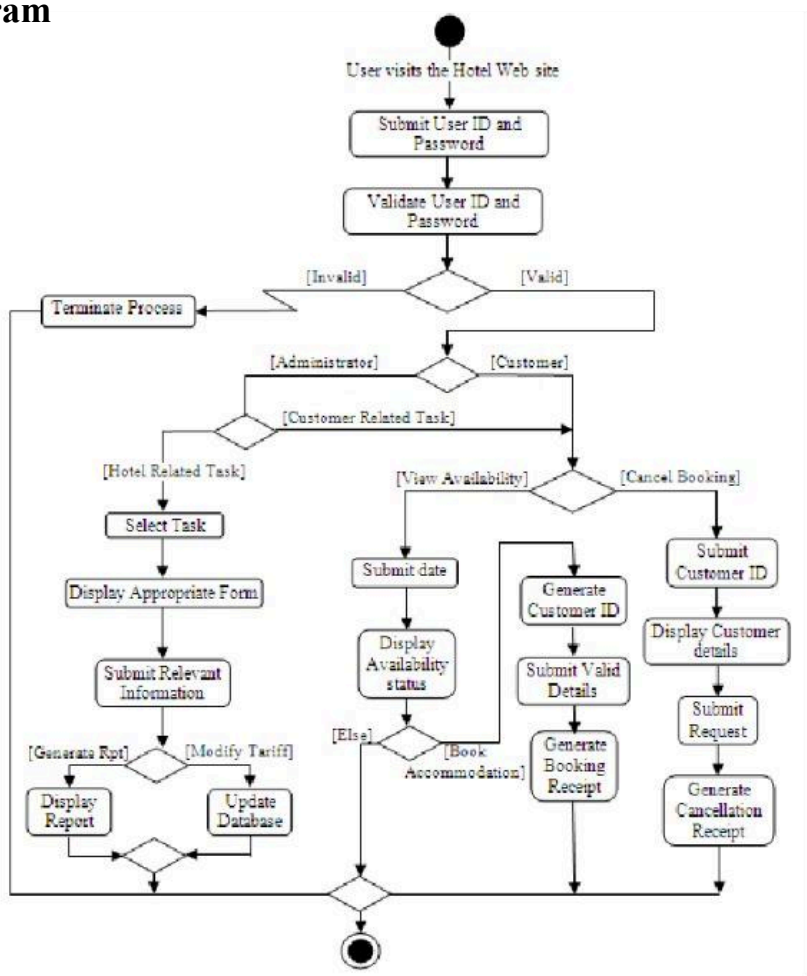

Fig 1.10: Hotel Management System Activity Diagram

Start Point: The process starts when a user visits the hotel website.

Submit User ID and Password: The user inputs their credentials.

Validate User ID and Password:

- If invalid, the process terminates.
- If valid, the user is classified as either an Administrator or a Customer.

Administrator Path:

- The administrator selects a hotel-related task.
- The system displays an appropriate form for the selected task.
- The administrator submits relevant information.
    - Generate Report: Creates a report based on hotel data.
    - Modify Tariff: Updates the database with modified tariffs.

Customer Path:

- Customer-Related Tasks: The customer chooses one of the following:
    1. View Availability:
        - The customer submits the desired date.
        - The system displays availability status.
        - If available, the customer can book accommodation.
        - Generates a booking receipt.
    2. Cancel Booking:
        - The customer submits their ID.
        - Customer details are displayed.
        - The system generates a cancellation receipt.

End Point: The process concludes once the respective task (booking, cancellation, or report generation) is completed.

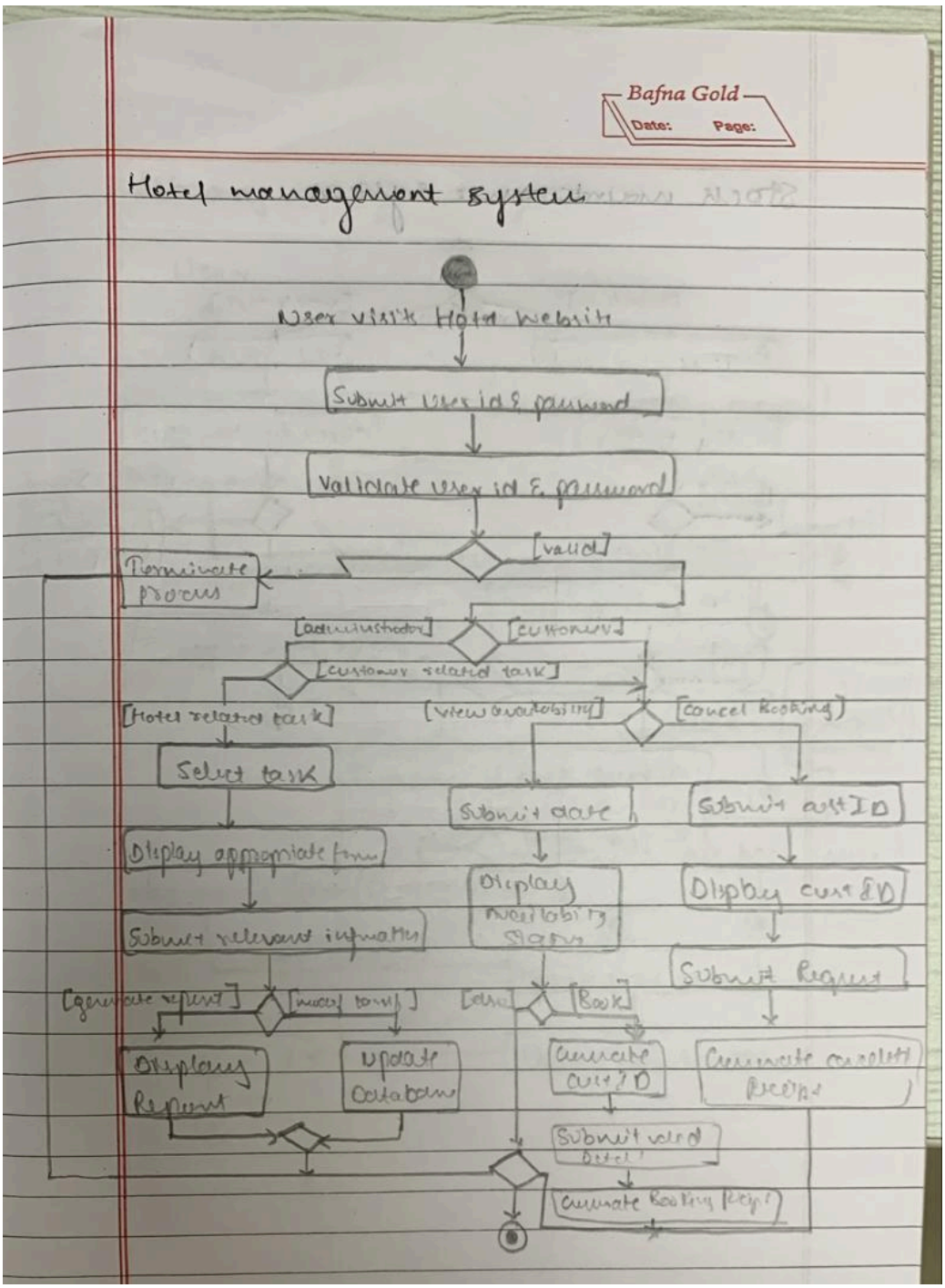

Fig 1.11: Hotel Management System Activity Diagram Observation

2. Credit Card Processing System

SRS-Software Requirements Specification

1. Introduction

1.1 Purpose This document outlines the functional, non-functional, and design requirements for the Credit Card Processing System. It serves as a guideline for developers to build a secure, efficient, and user-friendly system.

1.2 Scope The Credit Card Processing System is designed to handle credit card transactions, including authorization, refunds, and reporting. It will integrate with external banking systems and adhere to industry standards, such as PCI DSS, to ensure secure handling of customer data.

1.3 Definitions, Acronyms, and Abbreviations

- PCI DSS: Payment Card Industry Data Security Standard.
- CVV: Card Verification Value.
- Transaction: A financial operation such as payment, refund, or cancellation.

2. Overall Description

2.1 Product Perspective The system is a standalone application that interfaces with banks and other financial institutions. It will provide web-based and mobile-friendly access.

2.2 Product Functions

- Process credit card transactions: authorizations, refunds, and cancellations.
- Generate and manage statements.
- Provide reporting and reconciliation tools.
- Support customer account management.

2.3 User Characteristics Users include:

- Customers: Regular and Premium.
- Bank administrators.
- Retailers using the system for payment processing.

2.4 Constraints

- Compliance with PCI DSS standards.
- Integration with external banking APIs.

3. Functional Requirements

3.1 Customer Management

- Users can register, update contact details, and view transaction histories.
- Premium customers have additional privileges, such as priority support.

3.2 Credit Card Management

- Validate card information and manage card blocking.
- Support card replacement requests.

3.3 Transaction Processing

- Authorize, confirm, cancel, and refund transactions.

3.4 Statement Generation

- Generate monthly statements for customers detailing transaction history.

3.5 Reporting and Reconciliation

- Provide detailed transaction reports to banks and retailers.

4. Non-Functional Requirements

4.1 Security

- Encrypt sensitive data such as card numbers.
- Use tokenization for secure transaction processing.

4.2 Availability

- Ensure 99.9% uptime for transaction services.

4.3 Usability

- Provide an intuitive user interface for both web and mobile users.

5. System Architecture

5.1 Components

- Customer Module
- Transaction Module
- Credit Card Module
- Statement Module
- Bank Module

5.2 External Interfaces

- Integration with banking systems for transaction authorization and settlement.

- APIs for external retailers.

6. Design Constraints

- Must adhere to PCI DSS standards.
- Support for multi-currency transactions.

7. Preliminary Schedule

- Requirements Gathering: 2 weeks
- Design: 1 month
- Development: 3 months
- Testing: 1 month
- Deployment and Training: 2 weeks

8. Budget Estimated budget: $150,000

- Requirements Gathering: $15,000
- Design: $25,000
- Development: $80,000
- Testing: $20,000
- Deployment and Training: $10,000

9. Assumptions and Dependencies

- Dependence on third-party banking APIs.
- Assumes stable network infrastructure for seamless operation.

Class Diagram

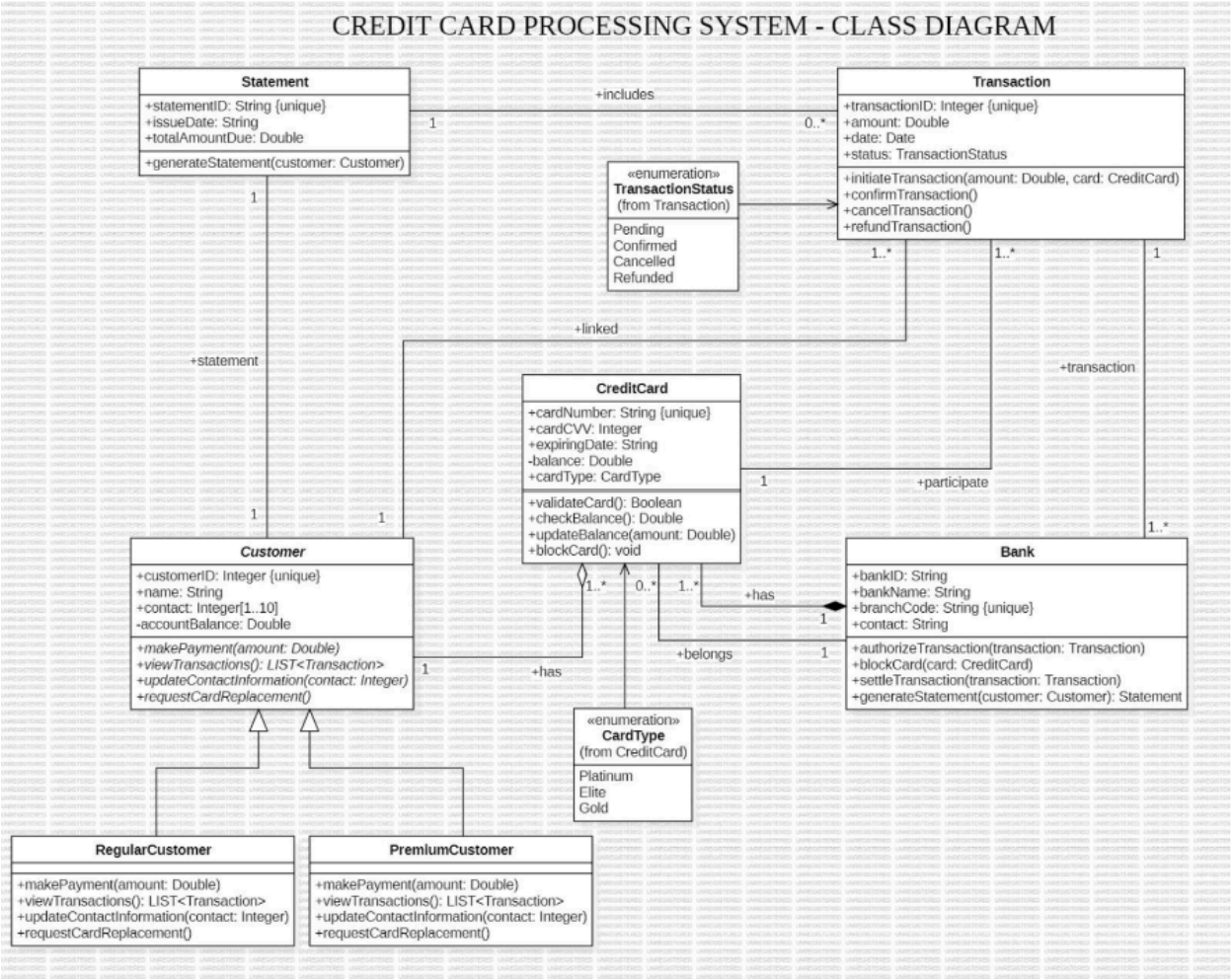

Fig 2.1: Credit Card Processing System Advanced Class Diagram

Statement

- Attributes: statementID, issueDate, totalAmountDue.
- Operations: Generates a statement for a specific customer.

Transaction

- Attributes: transactionID, amount, date, status (enumerated as Pending, Confirmed, Cancelled, Refunded).
- Operations: Handles initialization, confirmation, cancellation, and refund of transactions.

Customer

- Attributes: customerID, name, contact, accountBalance.
- Operations: Manages payments, views transactions, updates contact details, and requests card replacements.
- Two types of customers inherit from this: RegularCustomer and PremiumCustomer.

CreditCard

- Attributes: cardNumber, cardCVV, expiringDate, balance, cardType (enumerated as Platinum, Elite, Gold).
- Operations: Validates card details, checks balance, updates balance, and blocks the card.

Bank

- Attributes: bankID, bankName, branchCode, contact.
- Operations: Authorizes and settles transactions, blocks cards, and generates customer statements.

Relationships

- A Customer can have multiple CreditCards.
- A CreditCard participates in multiple Transactions.
- A Statement is linked to a Customer and includes Transactions.
- Bank interacts with Transactions and CreditCards for authorization, settlement, and blocking.

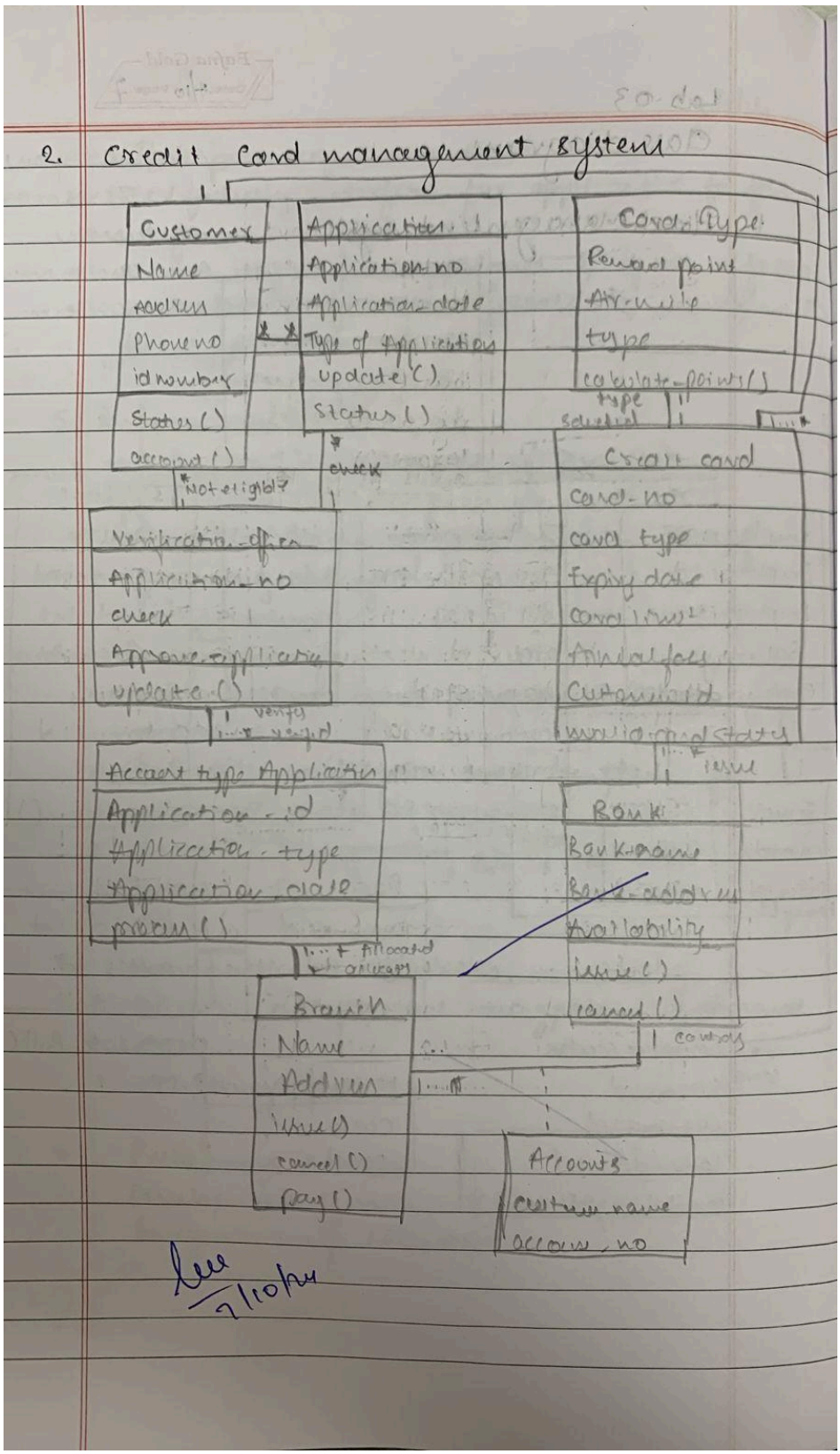

Fig 2.2: Credit Card Processing System Advanced Class Diagram Observation

State Diagram

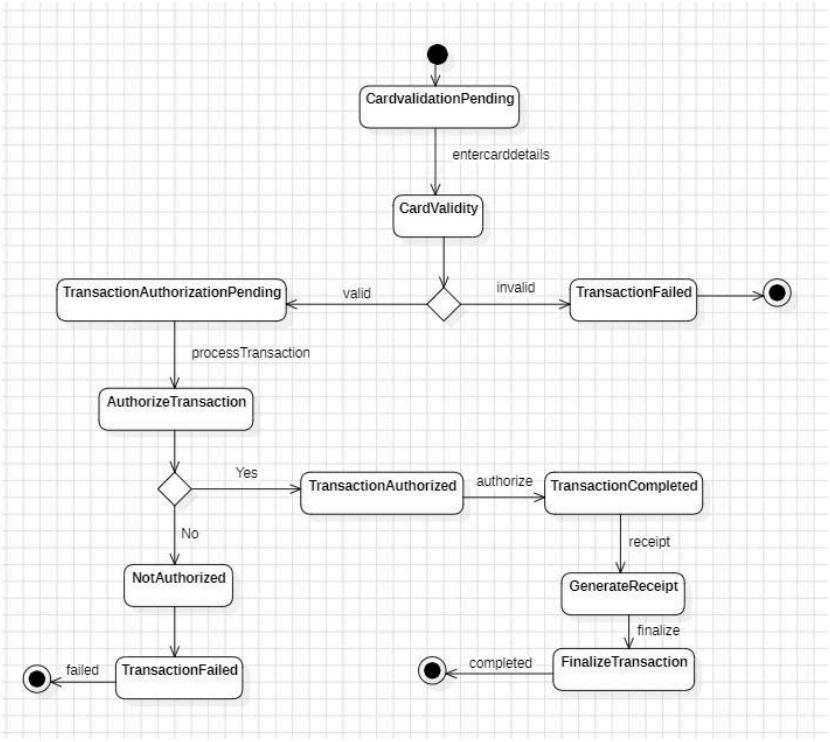

Fig 2.3: Credit Card Processing System Simple State diagram

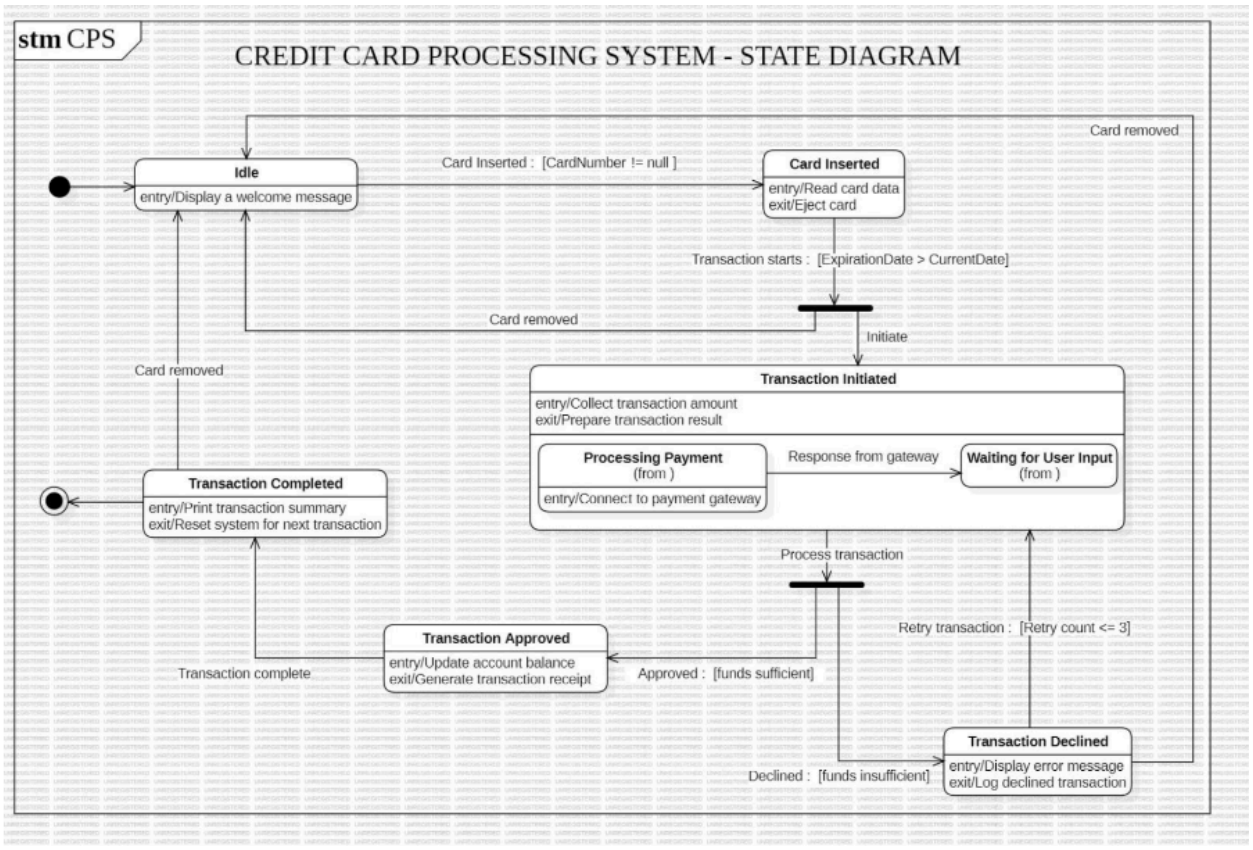

Fig 2.4: Credit Card Processing System Advanced State diagram

1. Idle:
    - The system is inactive and displays a welcome message.
    - Transition: Moves to the Card Inserted state when a card is detected.
2. Card Inserted:
    - The system reads the card details and ejects the card after reading.
    - Transition: Proceeds to the Transaction Initiated state if the card is valid and not expired (ExpirationDate > CurrentDate).
3. Transaction Initiated:
    - The system collects the transaction amount and prepares to process it.
    - Substates:
        - Processing Payment: Connects to the payment gateway for authorization.
        - Waiting for User Input: Awaits user confirmation or additional details if required.
    - Transition:
        - Moves to Transaction Approved if funds are sufficient.
        - Moves to Transaction Declined if funds are insufficient or an error occurs.
        - Allows retries if Retry count <= 3.
4. Transaction Approved:
    - Updates the account balance, generates a transaction receipt, and marks the transaction as complete.
    - Transition: Proceeds to Transaction Completed.
5. Transaction Declined:
    - Displays an error message and logs the declined transaction.
    - Transition: Returns to Idle or allows retries within the allowed limit.
6. Transaction Completed:
    - Prints the transaction summary and resets the system for the next transaction.
    - Transition: Returns to the Idle state after card removal.

Key Features:

- Retry Mechanism: Allows up to three retries for declined transactions.
- Error Handling: Logs declined transactions and displays relevant error messages.
- User Interaction: Waits for user input during critical stages like payment confirmation.

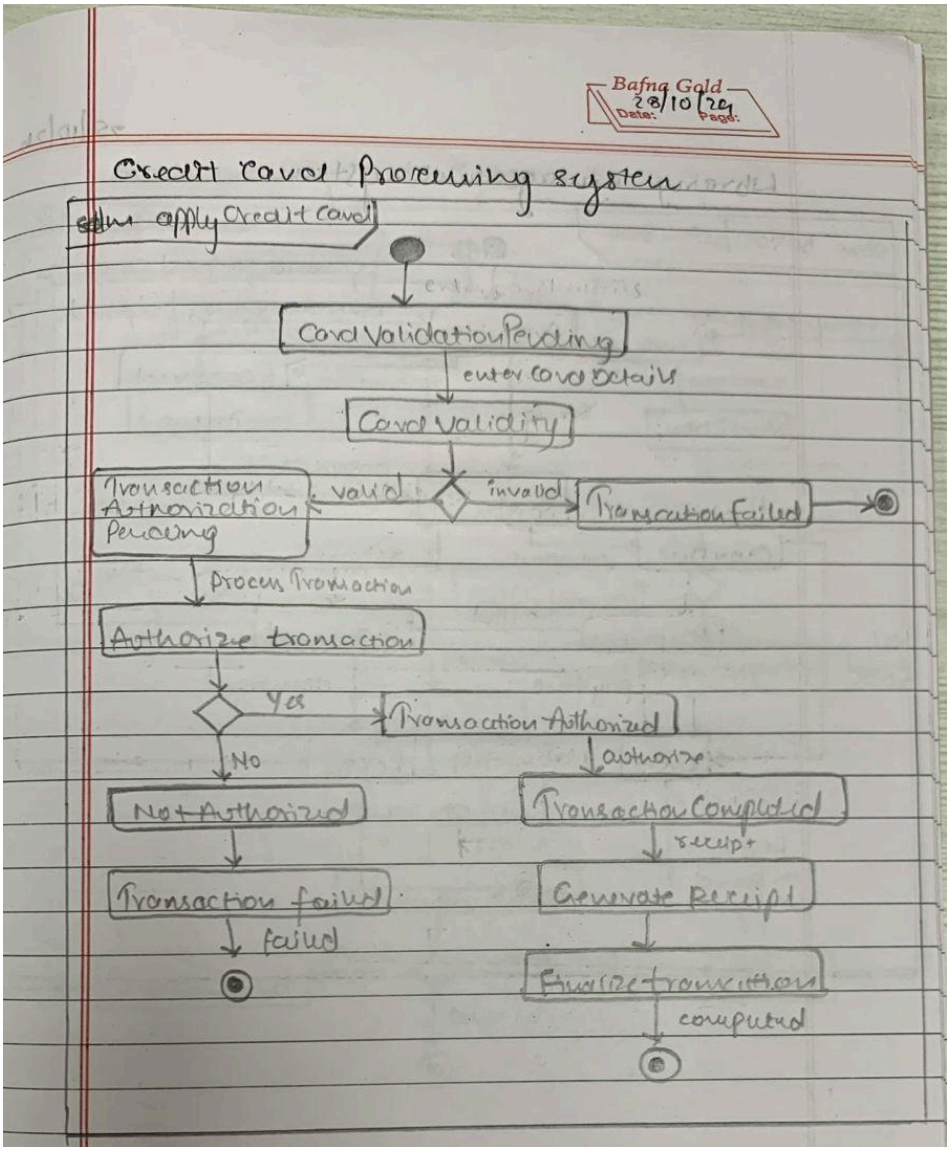

Fig 2.5: Credit Card Processing System Advanced State diagram Observation

Use Case Diagram

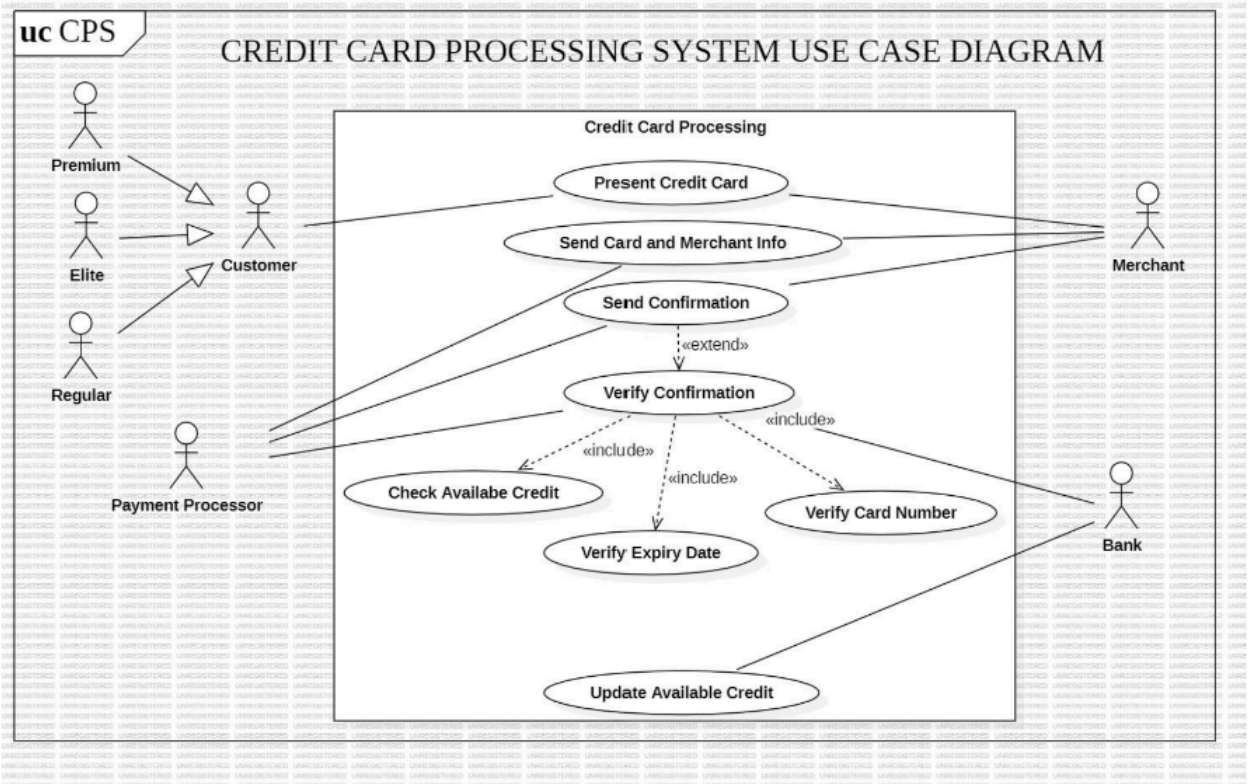

Fig 2.6: Credit Card Processing System Advanced UseCase diagram

Actors:

1. Customer: Represents the users of the credit card system (Premium, Elite, and Regular customers).
2. Merchant: Receives payment for transactions processed through the system.
3. Payment Processor: Facilitates the processing of credit card payments between customers, merchants, and banks.
4. Bank: Verifies card details, approves transactions, and updates credit information.

Key Use Cases:

1. Present Credit Card:
    - The process starts when a customer provides their credit card for payment.
    - Associated actor: Customer.
2. Send Card and Merchant Info:
    - Transfers the customer's card details and merchant information to initiate the payment process.

    - Associated actors: Customer and Merchant.
3. Send Confirmation:
    - A notification is sent to verify if the customer approves the transaction.
    - Associated actor: Customer.
    - Extension: Verify Confirmation.
4. Verify Confirmation:
    - Confirms the transaction details, including card validity and user consent.
    - Includes:
        - Check Available Credit: Ensures that sufficient credit is available for the transaction.
        - Verify Card Number: Validates the format and authenticity of the credit card number.
        - Verify Expiry Date: Ensures the card has not expired.
5. Update Available Credit:
    - Adjusts the customer's available credit after the transaction is approved.
    - Associated actor: Bank.

Relationships:

- Include: Indicates mandatory functionalities required for a use case to execute successfully. Examples include verifying the card number, expiry date, and checking available credit during the confirmation process.
- Extend: Highlights optional or conditional actions that extend the functionality of another use case. For instance, "Send Confirmation" extends to "Verify Confirmation."

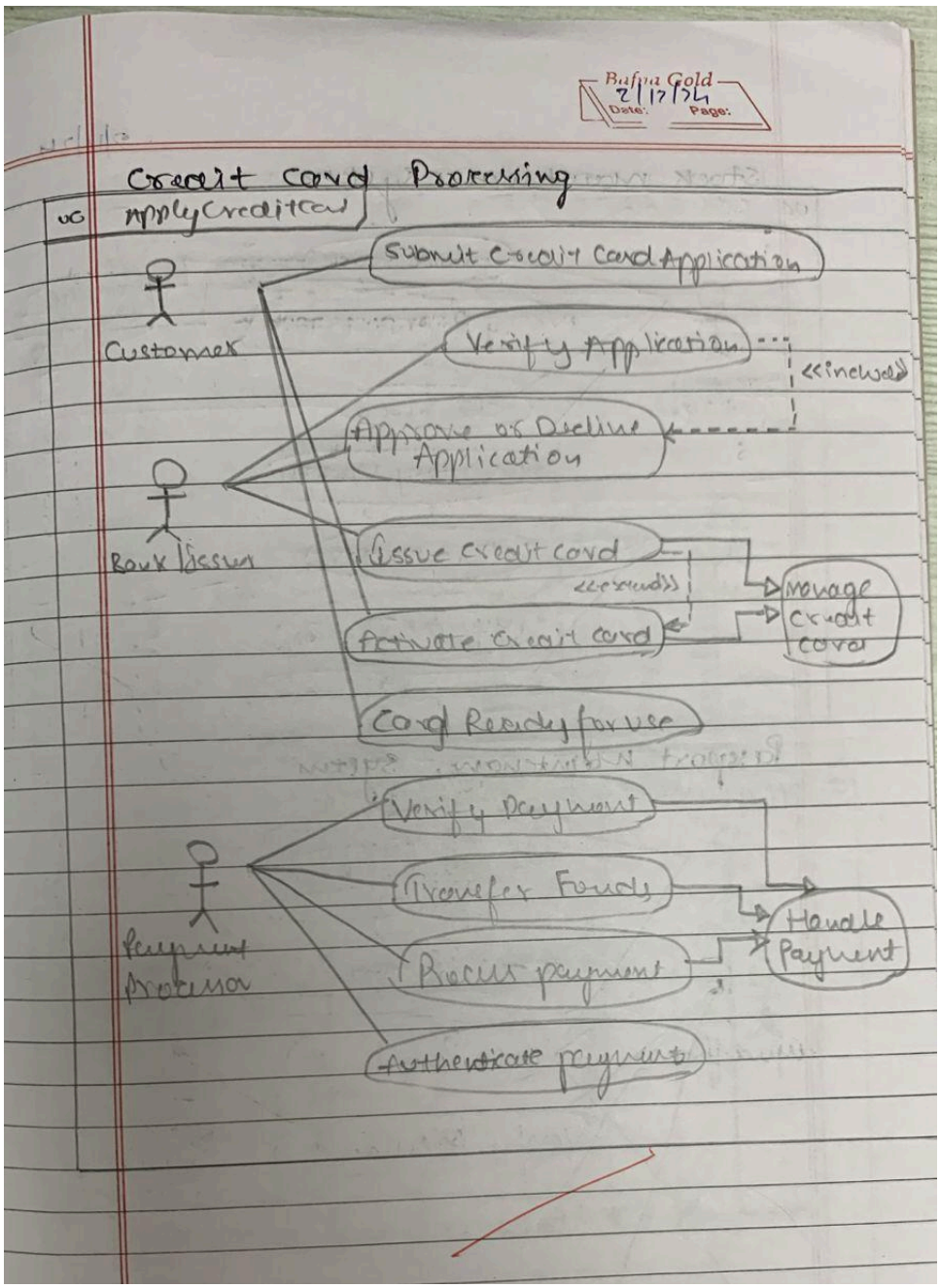

Fig 2.7: Credit Card Processing System Advanced UseCase diagram Observation

Sequence Diagram

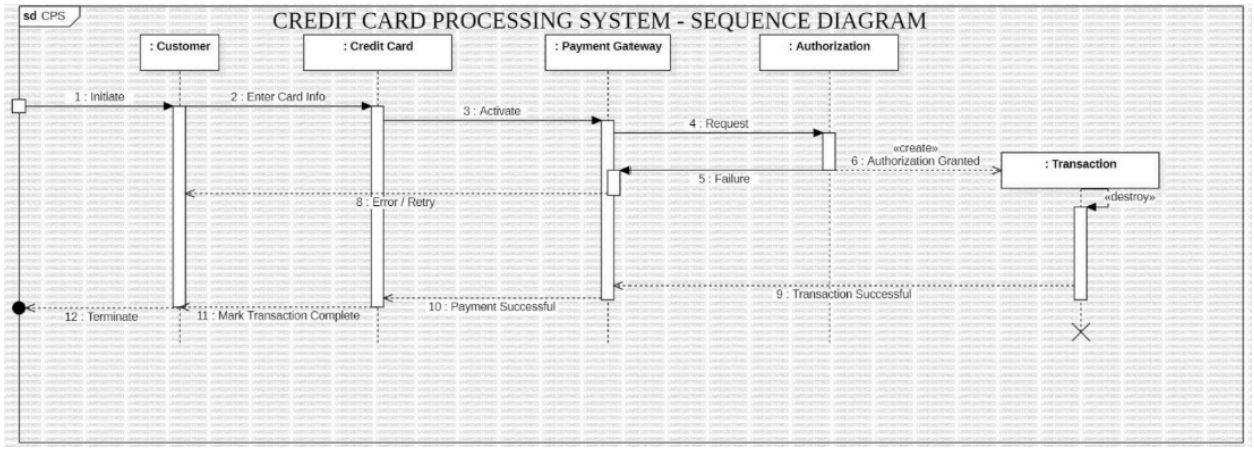

Fig 2.8: Credit Card Processing System Advanced Sequence diagram

Lifelines Involved:

1. Customer: Initiates the transaction by providing their credit card details.
2. Credit Card: Represents the card details provided for the transaction.
3. Payment Gateway: Facilitates communication between the customer and the bank, ensuring secure payment processing.
4. Authorization: Validates the transaction by checking card credentials, available credit, and other parameters.
5. Transaction: Represents the actual processing and finalization of the transaction.

Key Steps:

1. Initiate Transaction:
    - The process starts when the customer begins a transaction (e.g., swiping or entering card details).
2. Enter Card Information:
    - The credit card details are provided by the customer for processing.
3. Activate Payment Gateway:
    - The payment gateway is activated to handle the transaction request securely.
4. Request Authorization:
    - The payment gateway sends a request to the authorization system for transaction validation.

5. Authorization Validation:
    - The authorization system checks card details, available credit, and validity.
    - Possible outcomes:
        - Failure: If validation fails (e.g., insufficient funds or invalid card details), an error message is sent back.
        - Authorization Granted: If successful, an authorization is created, and the transaction proceeds.
6. Process Transaction:
    - A transaction is initiated and processed upon successful authorization.
7. Retry on Error:
    - If there is an error during payment, the system may allow retry attempts within specified limits.
8. Transaction Successful:
    - If the transaction is successful, payment is confirmed, and the customer is notified.
9. Mark Transaction Complete:
    - The transaction is finalized, and the system prepares for the next one.
10. Terminate Process:
    - The transaction session ends, and all relevant data is logged or destroyed securely.

Key Features:

- Error Handling: Includes retry mechanisms in case of errors during the transaction process.
- Secure Communication: Ensures sensitive data, such as card details, is processed securely through the payment gateway and authorization system.
- Finalization: Successfully marks and completes the transaction cycle, ensuring proper closure.

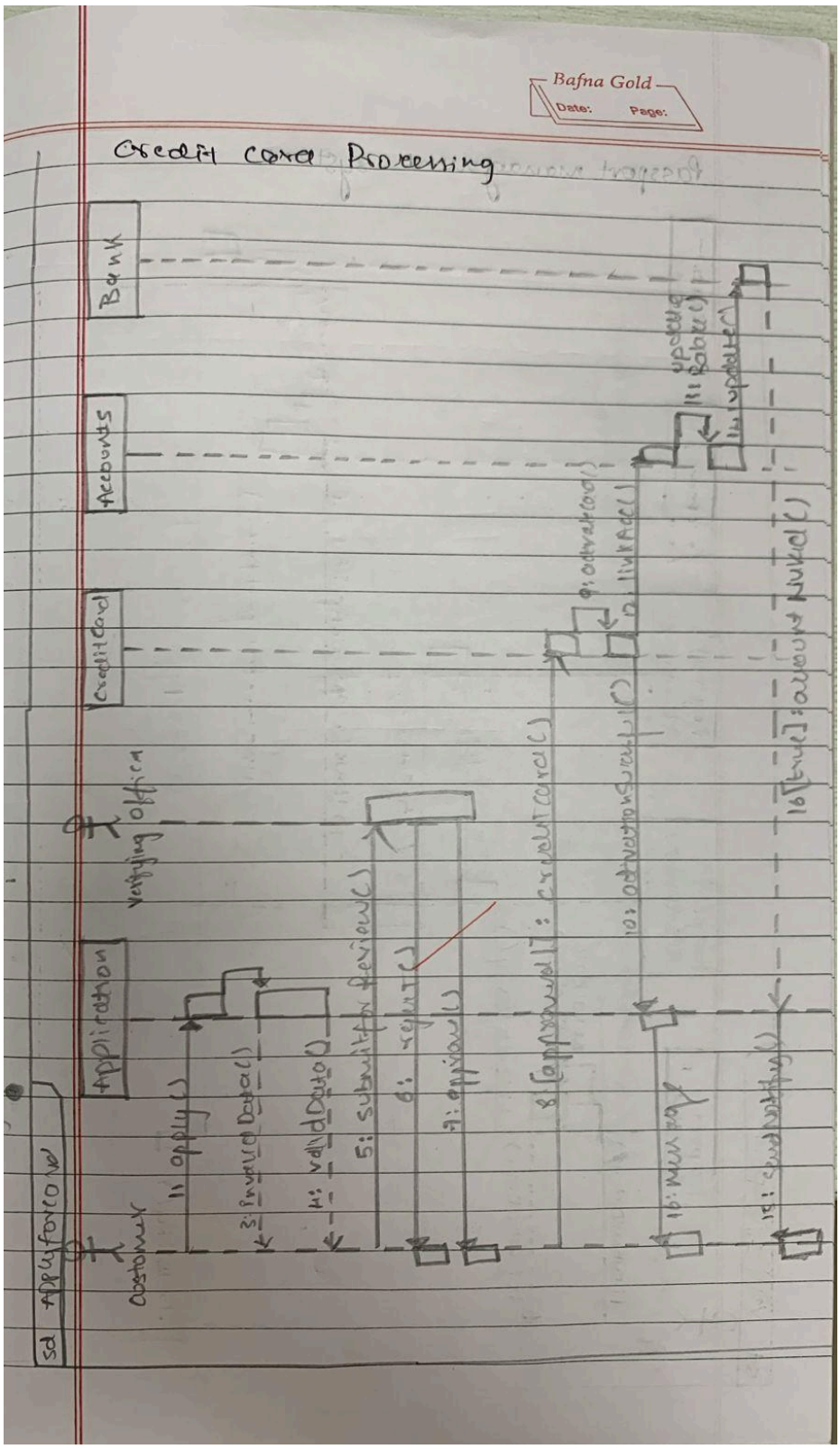

Fig 2.9: Credit Card Processing System Advanced Sequence diagram Observation

Activity Diagram

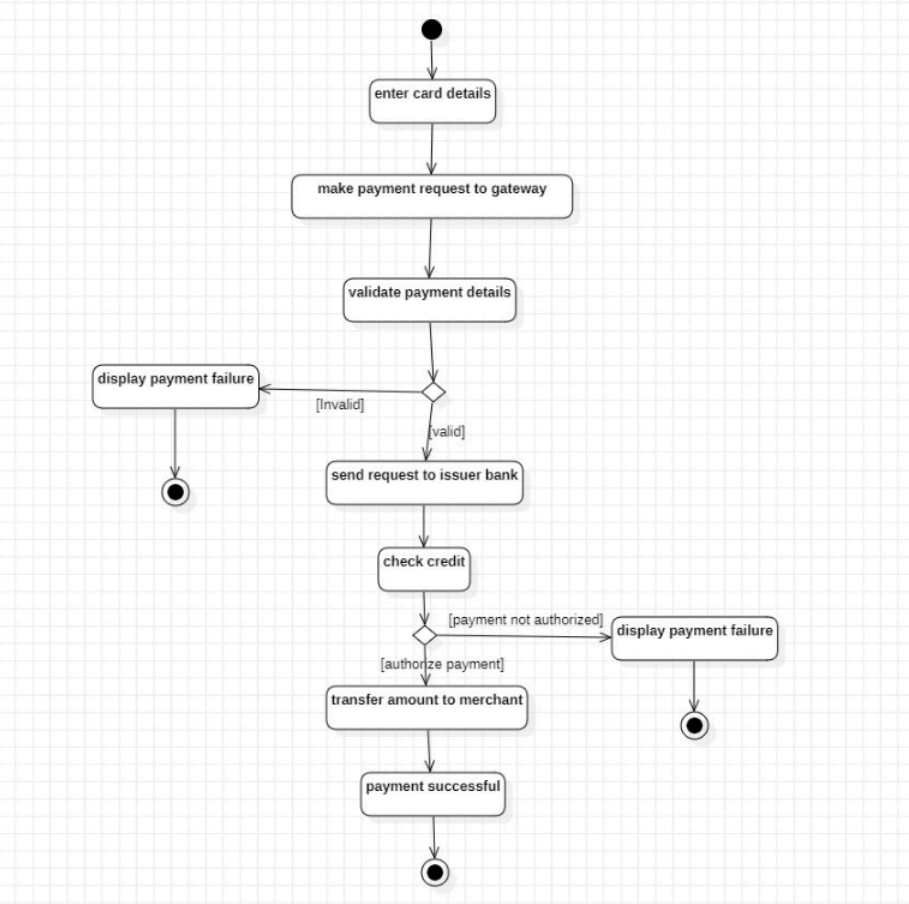

Fig 2.10: Credit Card processing system Activity Diagram

1. Enter Card Details: The customer initiates the transaction by entering their card details.
2. Make Payment Request to Gateway: The payment request is sent to the payment gateway.
3. Validate Payment Details: The payment gateway validates the provided card details.
    - If the details are invalid, the system displays a Payment Failure message.
    - If the details are valid, the process proceeds to the next step.
4. Send Request to Issuer Bank: The gateway forwards the payment request to the card issuer bank for further verification.
5. Check Credit: The issuer bank checks the cardholder's credit or balance.
    - If the payment is not authorized (e.g., insufficient funds), the system displays a Payment Failure message.
    - If the payment is authorized, the process continues.
6. Transfer Amount to Merchant: Once the payment is authorized, the required amount is transferred to the merchant's account.

7. Payment Successful: The transaction is completed successfully, and a success message is displayed to the customer.

This flow ensures that payments are processed securely, handling both successful and failed transactions effectively.

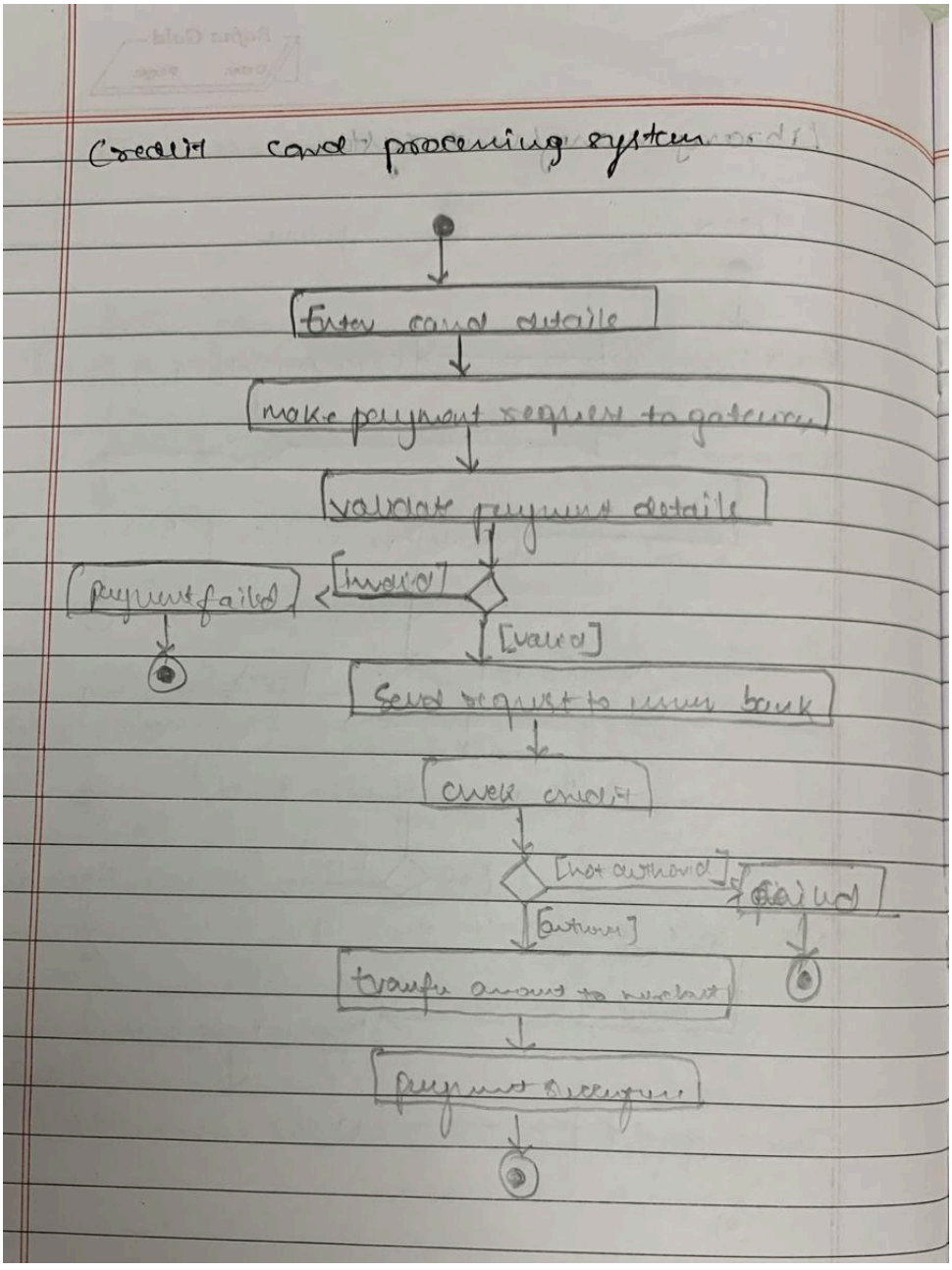

Fig 2.11: Credit Card processing system Activity Diagram Observation

3.Library Management System

SRS-Software Requirements Specification

1. Introduction

1.1 Purpose of this Document

This document outlines the software requirements for the Library Management System. It provides a clear understanding of the functionalities, design constraints, and overall goals of the software.

1.2 Scope of this Document

This document details the objectives, features, and user requirements for the Library Management System while addressing development costs and timelines.

1.3 Overview

The Library Management System will provide a solution for managing book inventory, member accounts, borrowing/returning processes, and late fee calculations. This system will enhance operational efficiency and improve the user experience for librarians and library members.

2. General Description

- Objectives: Simplify book management, streamline borrowing/returning processes, and provide comprehensive reporting tools for library operations.
- User Characteristics: Librarians, library staff, and library members.
- System Features:
    - Book catalog management.
    - Member account management.
    - Borrowing/returning processes with late fee handling.
    - Integration with online resources and e-books.
    - Reporting tools for inventory, borrowing trends, and overdue reports.

3. Functional Requirements

- Book Inventory Management: Add, update, and manage book details, including stock availability.
- Member Account Management: Create and manage library member profiles and keep a record of borrowing history.
- Borrow/Return Process: Facilitate easy borrowing and returning of books, with automated reminders for due dates.
- Fine Calculation: Calculate and process late fees for overdue returns.
- Search Functionality: Allow users to search for books by title, author, or genre.
- Reports: Generate reports on popular books, borrowing trends, and overdue records.

4. Non-functional Requirements

- User Interface: Must be user-friendly for library staff and members, accessible via desktop and mobile interfaces.
- API Integration: Integrate with online libraries, payment gateways, and e-book platforms.
- Real-Time Updates: Provide real-time updates for book availability and borrowing status.

5. Performance Requirements

- Response Time: The system should respond to user actions within 2 seconds.
- Availability: 99.9% uptime for all core functionalities.
- Scalability: Must handle up to 500 concurrent users during peak hours.

6. Design Constraints

- Must integrate with existing library systems, such as barcode scanners or RFID tags.
- Should comply with copyright laws and data protection regulations.

7. Non-functional Attributes

- Security: Ensure data security to prevent unauthorized access to book records and member information.
- Usability: Intuitive and easy to use for both library staff and members.

- Maintainability: The system should be easy to scale, update, and maintain over time.

8. Schedule & Budget

- Time Estimation: 6 to 8 months for development and deployment.
- Budget: $8,000 to $12,000 depending on advanced feature implementations.

Class Diagram

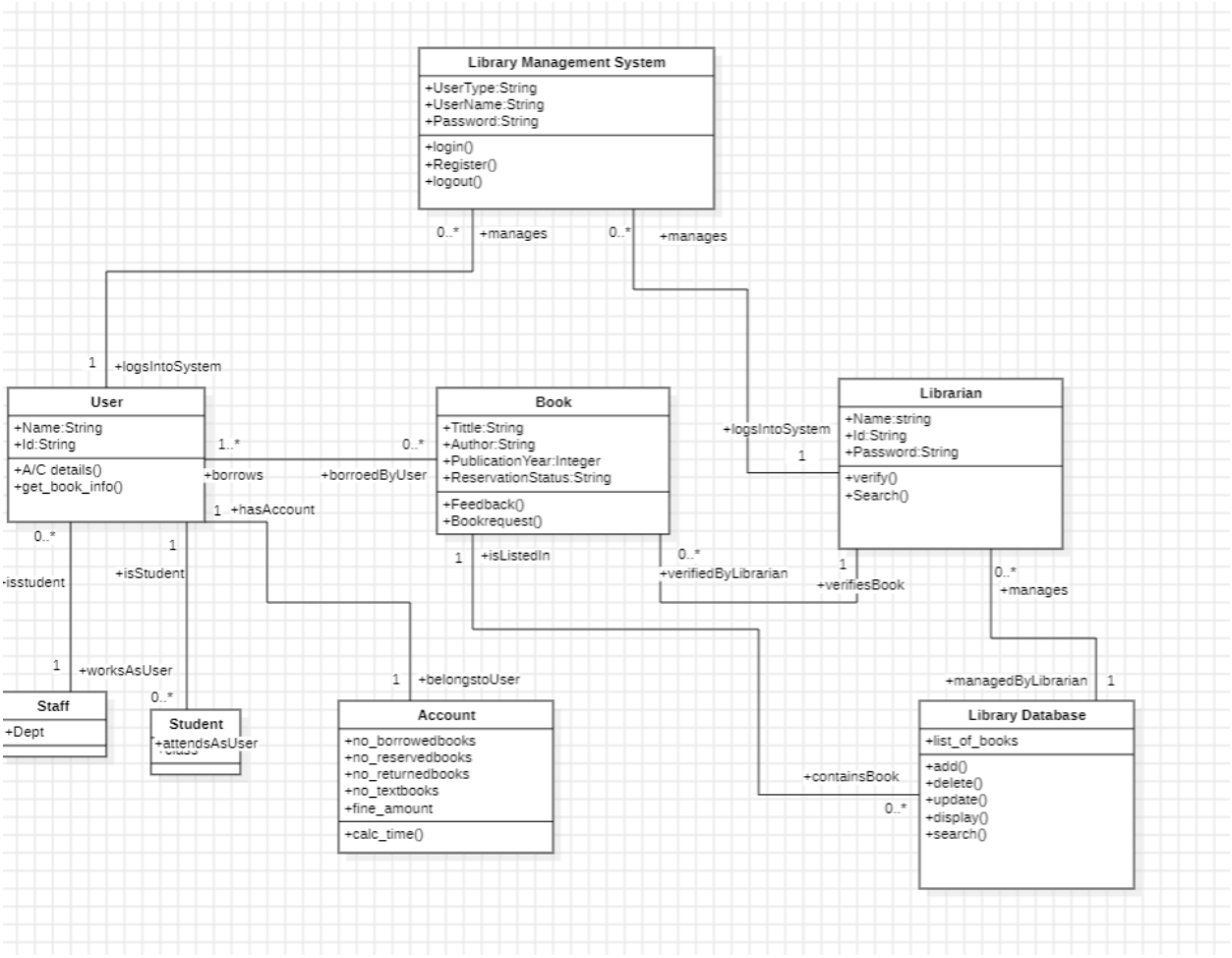

Fig.3.1: Library Management System Class Diagram

1. Library Management System

- Attributes:
    - UserType (String): Defines the type of user (e.g., Student, Librarian, Staff).
    - UserName (String): Stores the username for login.
    - Password (String): Stores the password for authentication.
- Operations:
    - login(): Handles user authentication.
    - Register(): Allows new users to register.
    - logout(): Logs the user out of the system.
- Relationships:
    - Manages multiple User, Book, and Library Database instances.

2. User

- Attributes:
    - Name (String): Name of the user.
    - Id (String): Unique ID for the user.
- Operations:
    - A/C details(): Retrieves account details.
    - get_book_info(): Fetches book information for the user.
- Relationships:
    - Each User can borrow multiple books (borrows).
    - Associated with an Account class (hasAccount).
    - Differentiated into Student or Staff using isStudent and worksAsUser.

3. Book

- Attributes:
    - Title (String): Title of the book.
    - Author (String): Author of the book.
    - PublicationYear (Integer): Year of publication.
    - ReservationStatus (String): Status of the book (e.g., Reserved, Available).

- Operations:
    - Feedback(): Allows users to leave feedback for the book.
    - Bookrequest(): Handles requests for reserving the book.
- Relationships:
    - Books can be borrowed by users and verified by the librarian.

4. Librarian

- Attributes:
    - Name (String): Name of the librarian.
    - Id (String): Unique ID for the librarian.
    - Password (String): Password for login.
- Operations:
    - verify(): Verifies book details or reservations.
    - Search(): Searches for specific books or users.
- Relationships:
    - Manages the Library Database and verifies books.

5. Account

- Attributes:
    - no_borrowedbooks: Number of books borrowed by the user.
    - no_reservedbooks: Number of reserved books.
    - no_returnedbooks: Number of returned books.
    - no_textbooks: Number of textbooks issued.
    - fine_amount: Fine amount for overdue books.
- Operations:
    - calc_time(): Calculates overdue time and fines.
- Relationships:
    - Each User has one Account.

6. Library Database

- Attributes:

    - list_of_books: Stores the list of all books in the library.
- Operations:
    - add(): Adds a new book.
    - delete(): Deletes an existing book.
    - update(): Updates book details.
    - display(): Displays the list of books.
    - search(): Searches for specific books.
- Relationships:
    - Managed by the Librarian.
    - Contains multiple Book instances.

7. Student (Subclass of User)

- Attributes:
    - rollNo: Stores the roll number of the student.
- Relationships:
    - Inherits User properties and attends the library as a user.

8. Staff (Subclass of User)

- Attributes:
    - Dept: The department the staff belongs to.
- Relationships:
    - Inherits User properties and works as a user in the library.

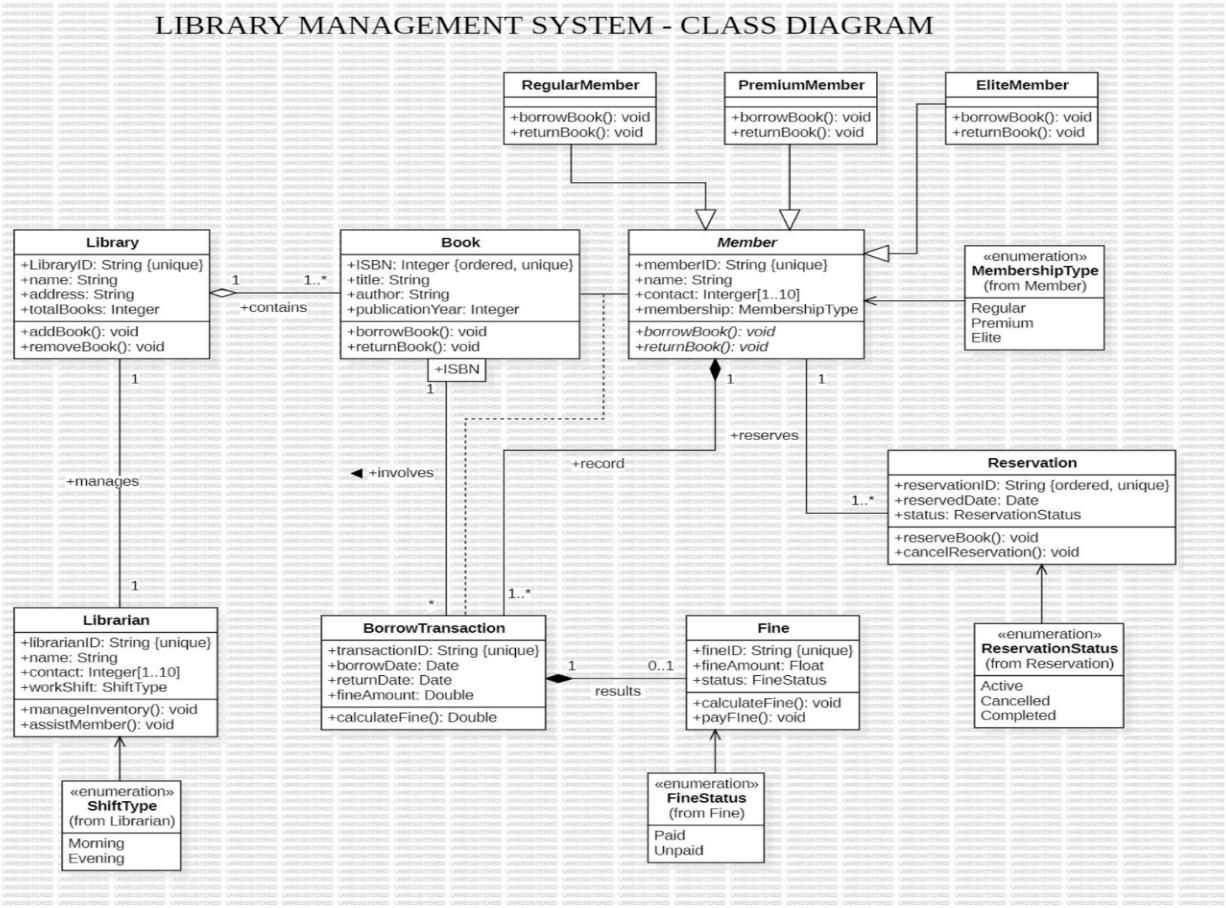

Fig.3.2: Library Management System Advanced Class Diagram

1. Library:
    - Attributes: libraryID, name, address, totalBooks.
    - Operations:
        - addBook(): Adds a book to the library's inventory.
        - removeBook(): Removes a book from the inventory.
    - Relationships:
        - contains multiple Books.
        - manages multiple BorrowTransactions.
2. Book:
    - Attributes: ISBN, title, author, publicationYear.
    - Operations:
        - borrowBook(): Marks a book as borrowed.
        - returnBook(): Marks a book as returned.

- Relationships:
    - Associated with Library and BorrowTransaction.

3. Member:
    - Attributes: memberID, name, contact, membership (linked to MembershipType).
    - Operations:
        - borrowBook(): Allows a member to borrow a book.
        - returnBook(): Allows a member to return a book.
    - Subclasses:
        - RegularMember
        - PremiumMember
        - EliteMember (Inheritance defines specialized members with shared functionality).
4. Librarian:
    - Attributes: librarianID, name, contact, workShift (linked to ShiftType).
    - Operations:
        - manageInventory(): Manages books in the library.
        - assistMember(): Helps library members.
5. BorrowTransaction:
    - Attributes: transactionID, borrowDate, returnDate, fineAmount.
    - Operations:
        - calculateFine(): Calculates fines for late returns.
    - Relationships:
        - Associated with Member, Book, and Fine.
6. Reservation:
    - Attributes: reservationID, reservedDate, status (linked to ReservationStatus).
    - Operations:
        - reserveBook(): Reserves a book.
        - cancelReservation(): Cancels an existing reservation.
7. Fine:
    - Attributes: fineID, fineAmount, status (linked to FineStatus).
    - Operations:
        - calculateFine(): Computes the amount of fine.

        - payFine(): Marks a fine as paid.

Enumerations:

1. MembershipType:
    - Defines member types: Regular, Premium, Elite.
2. ShiftType:
    - Defines librarian shifts: Morning, Evening.
3. ReservationStatus:
    - Defines reservation states: Active, Cancelled, Completed.
4. FineStatus:
    - Defines fine payment states: Paid, Unpaid.

Relationships:

- Composition/Association:
    - Library contains Book and manages BorrowTransaction.
    - Member reserves Books and performs BorrowTransaction.
    - Reservation links Member and Book.
- Generalization:
    - Member has subclasses (RegularMember, PremiumMember, EliteMember) to differentiate functionalities based on membership type.

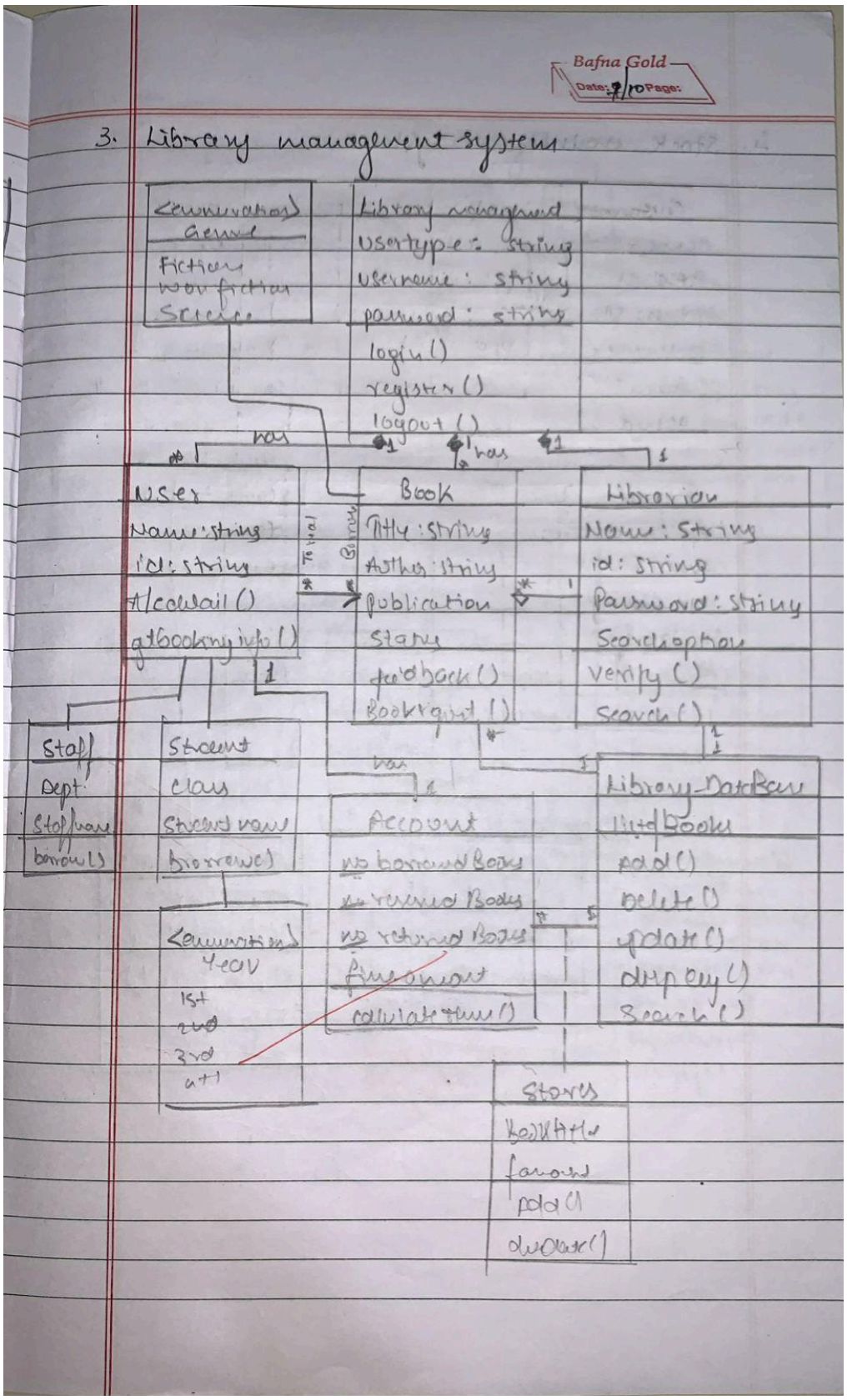

Fig.3.3: Library Management System Advanced Class Diagram Observation

State Diagram

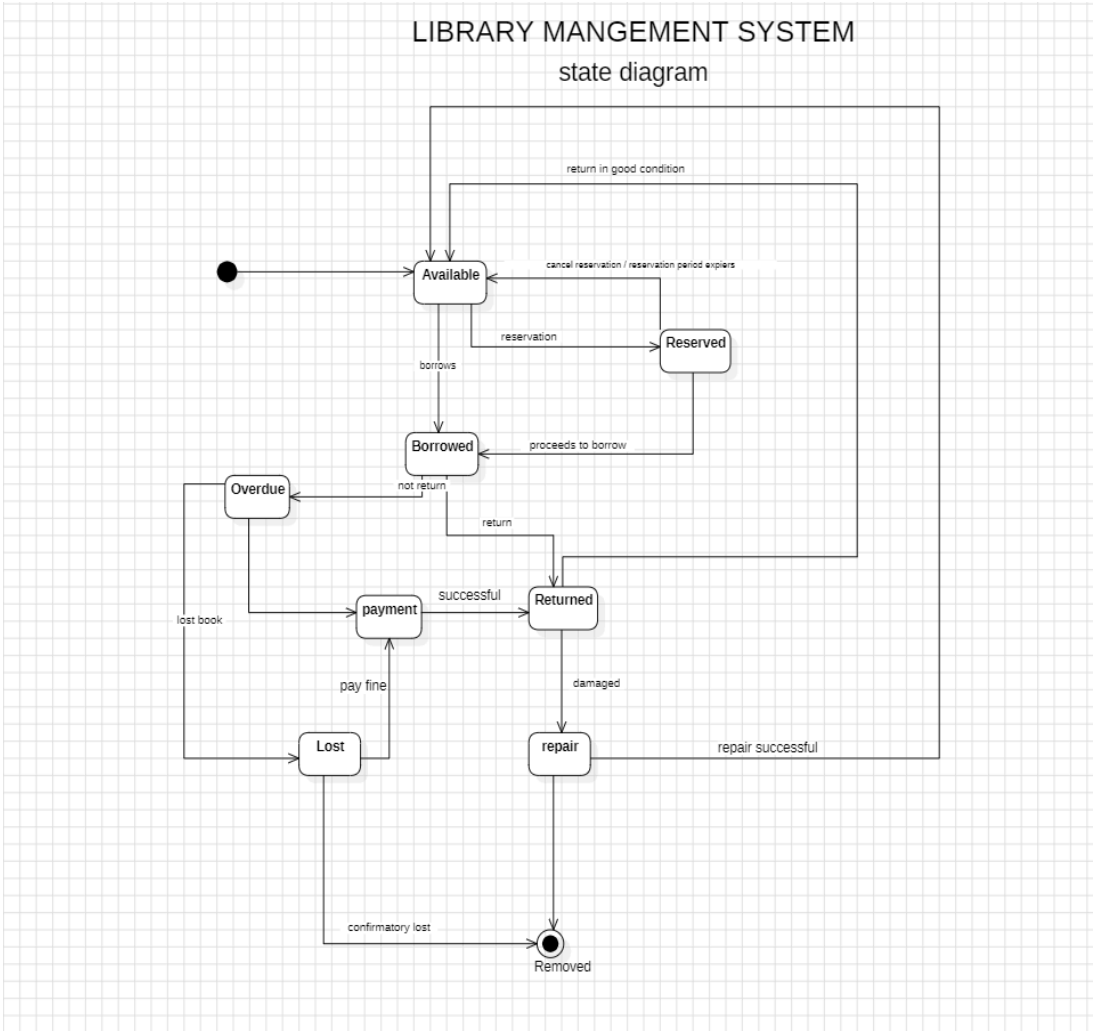

Fig.3.3.Simple State Diagram

States:

1. Available:
    - The book is available for borrowing or reservation.
    - Transition:
        - Moves to Reserved when a reservation is made.
        - Moves to Borrowed when a user borrows the book.
2. Reserved:
    - The book is reserved by a user.
    - Transition:

        - Returns to Available if the reservation is canceled or the reservation period expires.
        - Moves to Borrowed when the user proceeds to borrow the reserved book.
3. Borrowed:
    - The book is issued to a user.
    - Transition:
        - Moves to Overdue if the book is not returned within the due date.
        - Moves to Returned when the user returns the book.
        - Moves to Lost if the user reports the book as lost.
4. Overdue:
    - The book is not returned by the due date.
    - Transition:
        - Moves to Payment for fine collection.
        - Moves to Returned if the user pays the fine and returns the book.
5. Returned:
    - The book is returned by the user.
    - Transition:
        - Moves to Available if the book is in good condition.
        - Moves to Repair if the book is returned damaged.
6. Lost:
    - The book is reported as lost by the user.
    - Transition:
        - Moves to Payment for paying the replacement cost.
        - Finally transitions to Removed if the book is confirmed as lost.
7. Repair:
    - The book is damaged and under repair.
    - Transition:
        - Moves to Available if the repair is successful.
        - Moves to Removed if the book cannot be repaired.
8. Removed:
    - The book is permanently removed from the system due to irreparable damage or being lost.

Transitions:

- Borrow/Return: Users borrow books from the available state and return them to either the available state (if in good condition) or repair state.
- Reservation: Books can be reserved and later borrowed.
- Fine/Payment: Overdue or lost books lead to payment of fines or replacement costs.
- Repair/Remove: Damaged books can either be repaired or removed based on their condition.

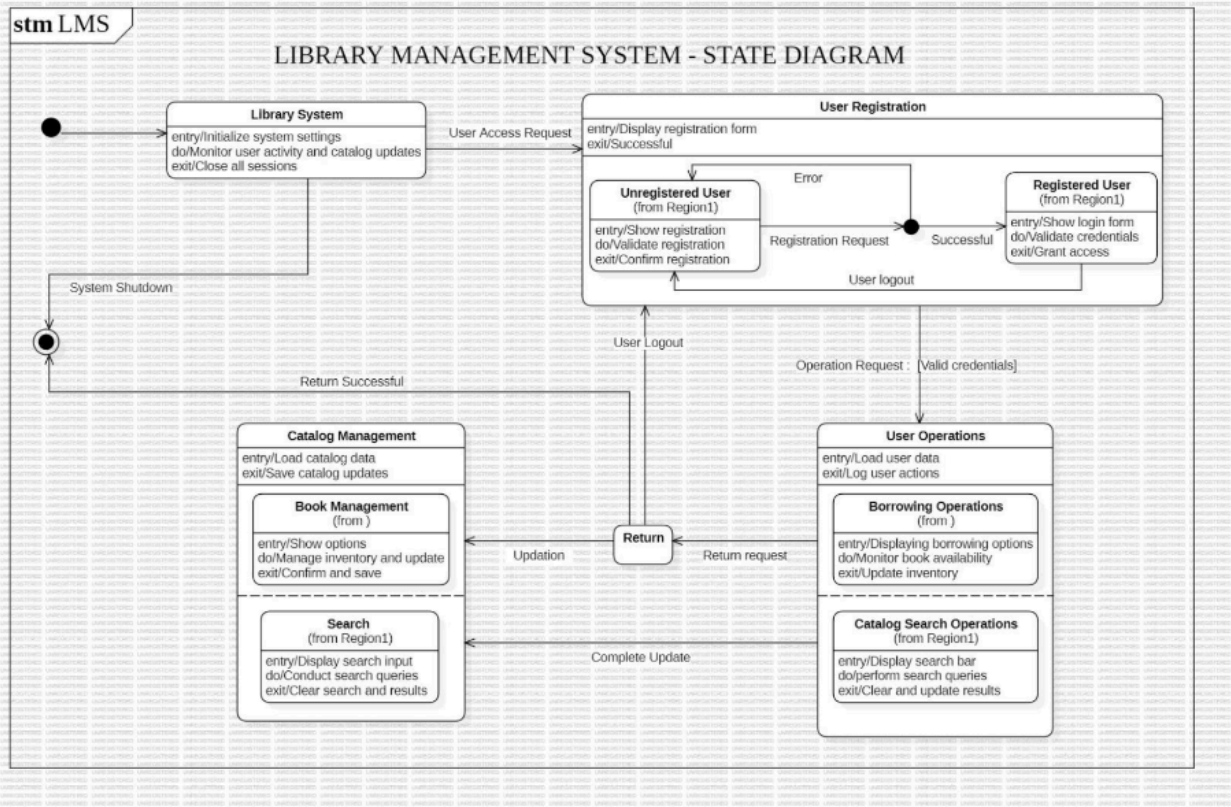

Fig.3.4. Advanced StateDiagram

1. Library System:
    - Initial State: The system initializes settings, monitors user activity, and updates the catalog.
    - Final State: System shutdown after all sessions are closed.
2. User Registration:
    - Unregistered User: Displays a registration form and validates user information.
        - On successful registration, transitions to a Registered User.
        - On error, the system handles validation failures.
    - Registered User: Displays login form, validates credentials, and grants access upon success.
3. Catalog Management:
    - Manages catalog data, including:
        - Book Management: Handles inventory updates and confirms changes.
        - Search: Allows search queries, processes them, and clears results.
4. User Operations:
    - Includes actions performed by users, such as:
        - Borrowing Operations: Monitors book availability and updates the inventory.
        - Catalog Search Operations: Displays a search bar and retrieves results.
5. Return Process:
    - Users can return books, triggering inventory updates and marking the return as complete.

Workflow Summary:

- Users start at the Library System and either register or log in.
- Registered users can perform operations like catalog searches or borrowing books.
- The system manages transitions, ensuring data is updated appropriately, and eventually transitions to shutdown.

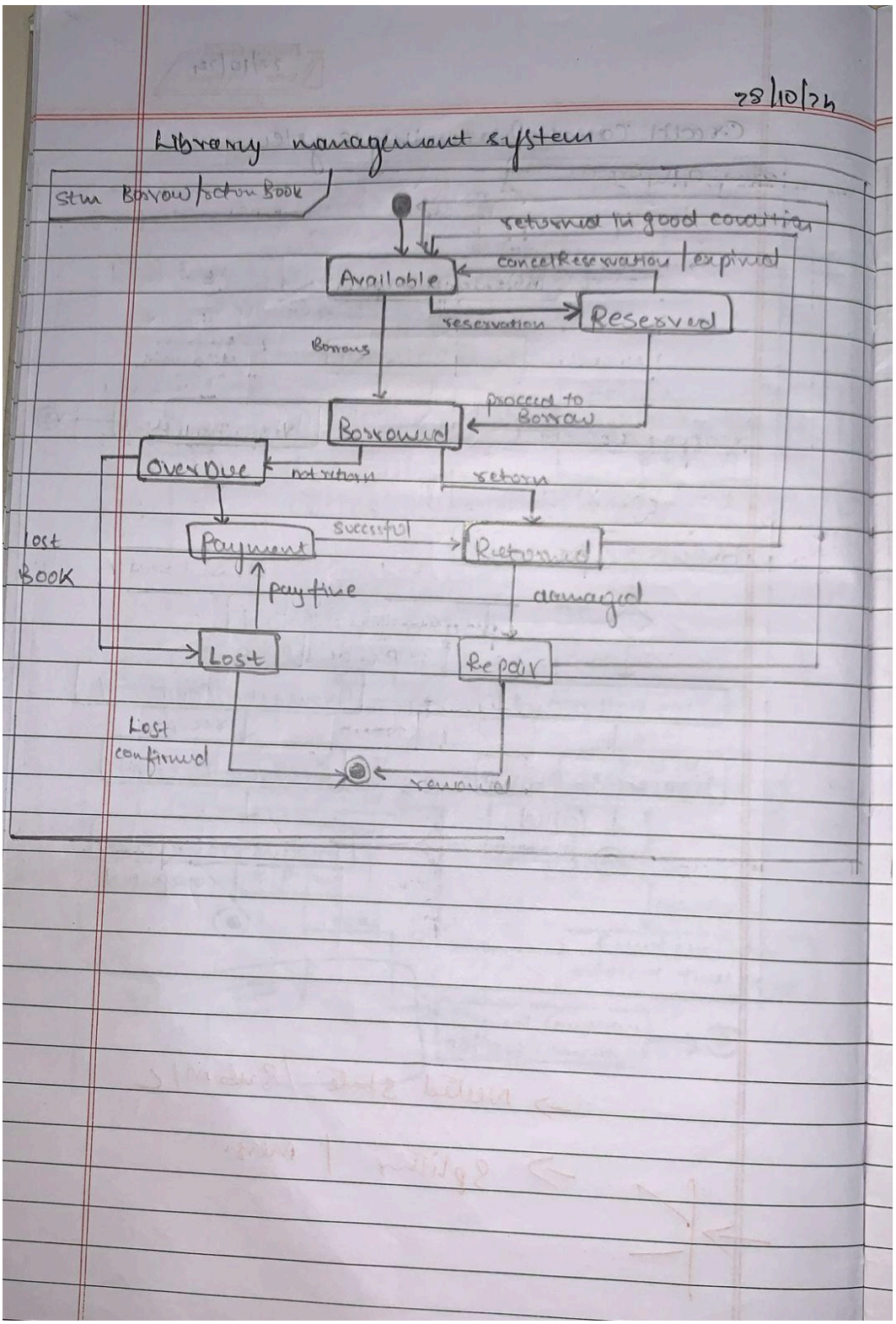

Fig.3.4. Advanced State Diagram Observation

Sequence Diagram

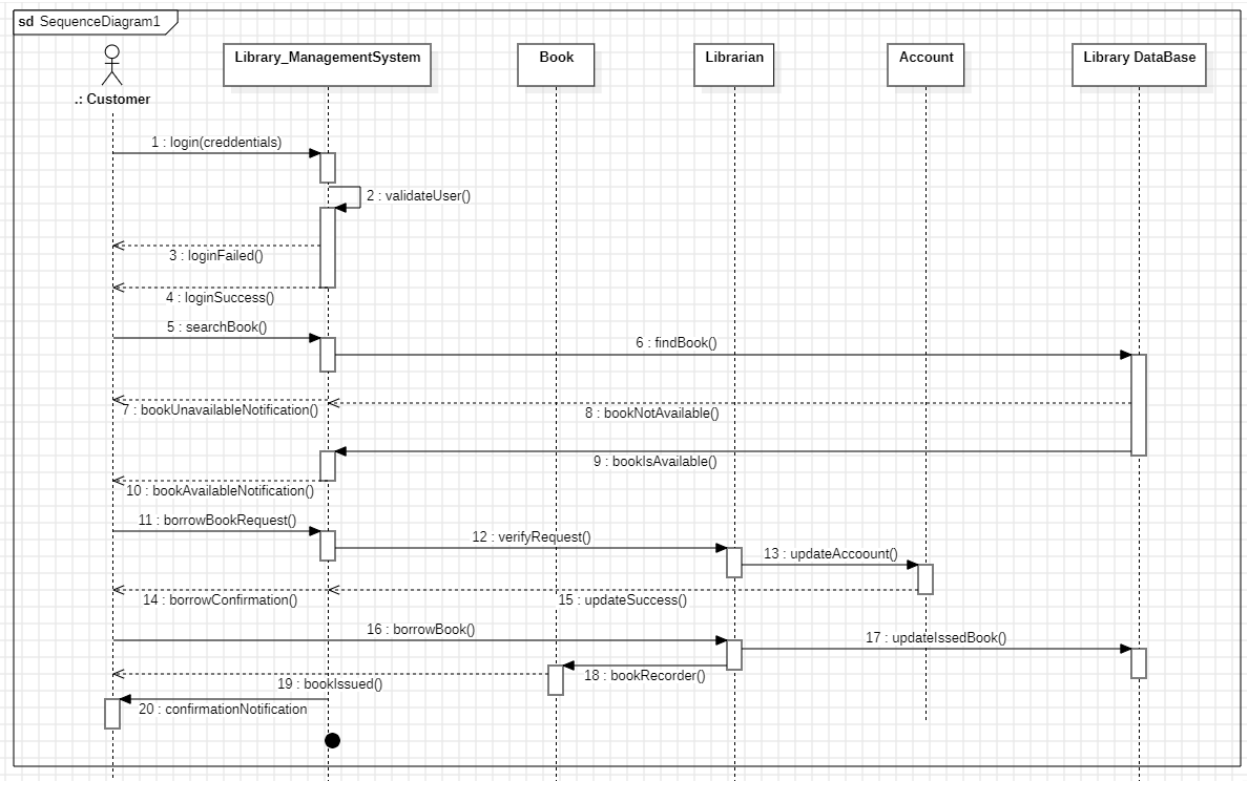

Fig.3.5. Sequence Diagram

1. Customer: Initiates login, searches for a book, and requests to borrow a book.
2. Library Management System: Handles login validation, book search, and request processing.
3. Book: Represents the entity being searched or borrowed.
4. Librarian: Verifies availability and approves book borrowing.
5. Library Database/Account: Updates and manages the borrowing record.

Steps:

1. Login Process:
    - The customer provides credentials to the system.
    - If credentials are valid, the user proceeds; otherwise, login fails.
2. Book Search:
    - The customer searches for a book. The system communicates with the library database to check availability.

- If the book is unavailable, a notification is sent to the user.

3. Book Borrowing:
   - The system validates and processes the borrow request.
   - The librarian verifies and updates the account to reflect the borrowed book.
   - Once successful, a confirmation is sent, and the book is issued.

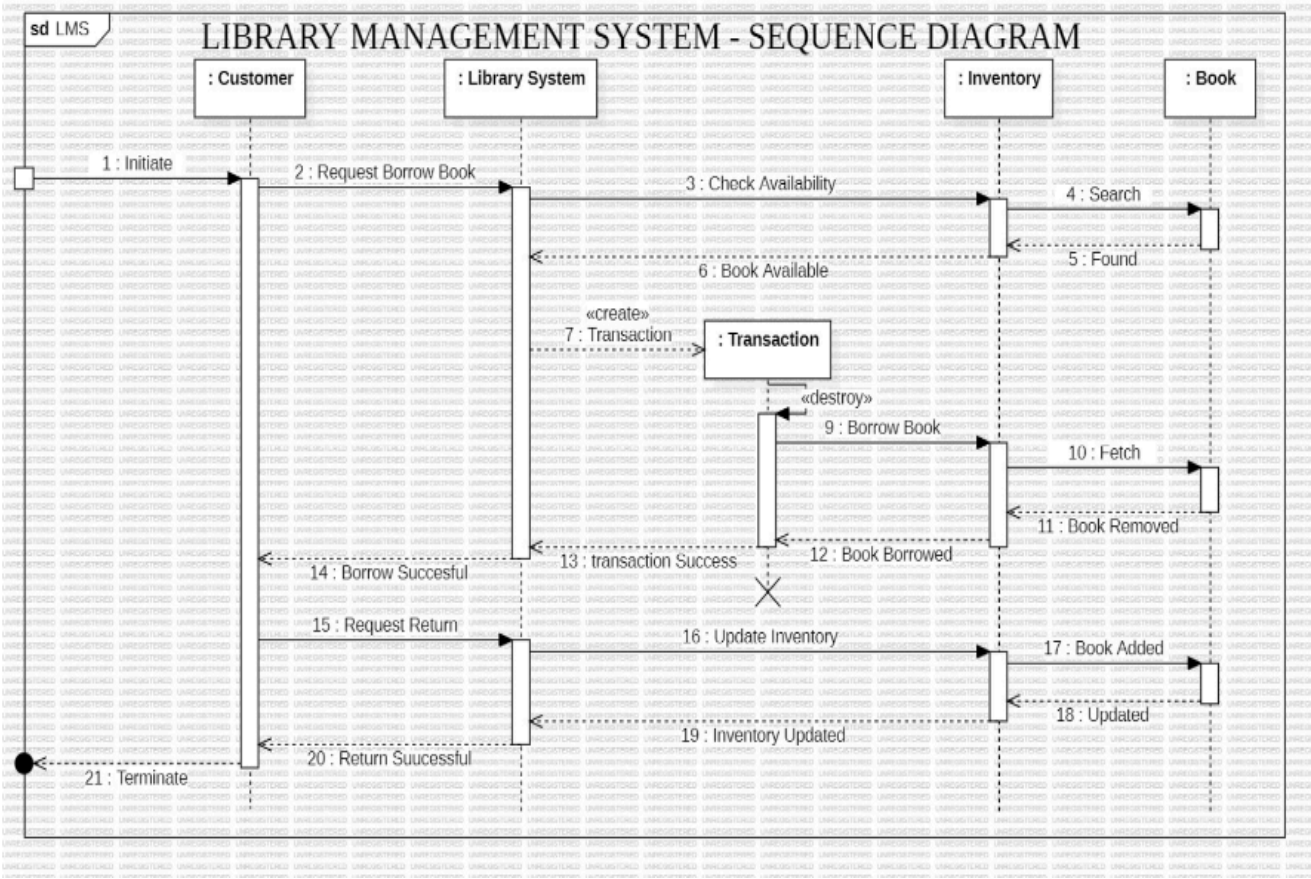

Fig.3.6: Advanced Sequence Diagram

1. Customer: Initiates a borrow or return transaction.
2. Library System: Centralized system managing all library-related processes.
3. Inventory: Tracks available and borrowed books.
4. Book: Represents the book being processed.
5. Transaction (Transient Object): Created to manage a specific borrowing or return operation.

Steps:

1. Initiation:
    - The customer initiates a request to borrow or return a book.
2. Book Availability Check:
    - The system verifies the book's availability by communicating with the inventory and book entities.
3. Transaction Management:
    - A transient transaction object is created to encapsulate the borrowing/return process.
    - Once the transaction is completed, the object is destroyed.
4. Book Borrowing:
    - If the book is available, the inventory is updated, and the book is marked as borrowed.
5. Return Process:
    - The customer can request a return, and the system updates the inventory to add the book back.
6. Completion:
    - A confirmation is sent to the customer upon successful borrowing or return.

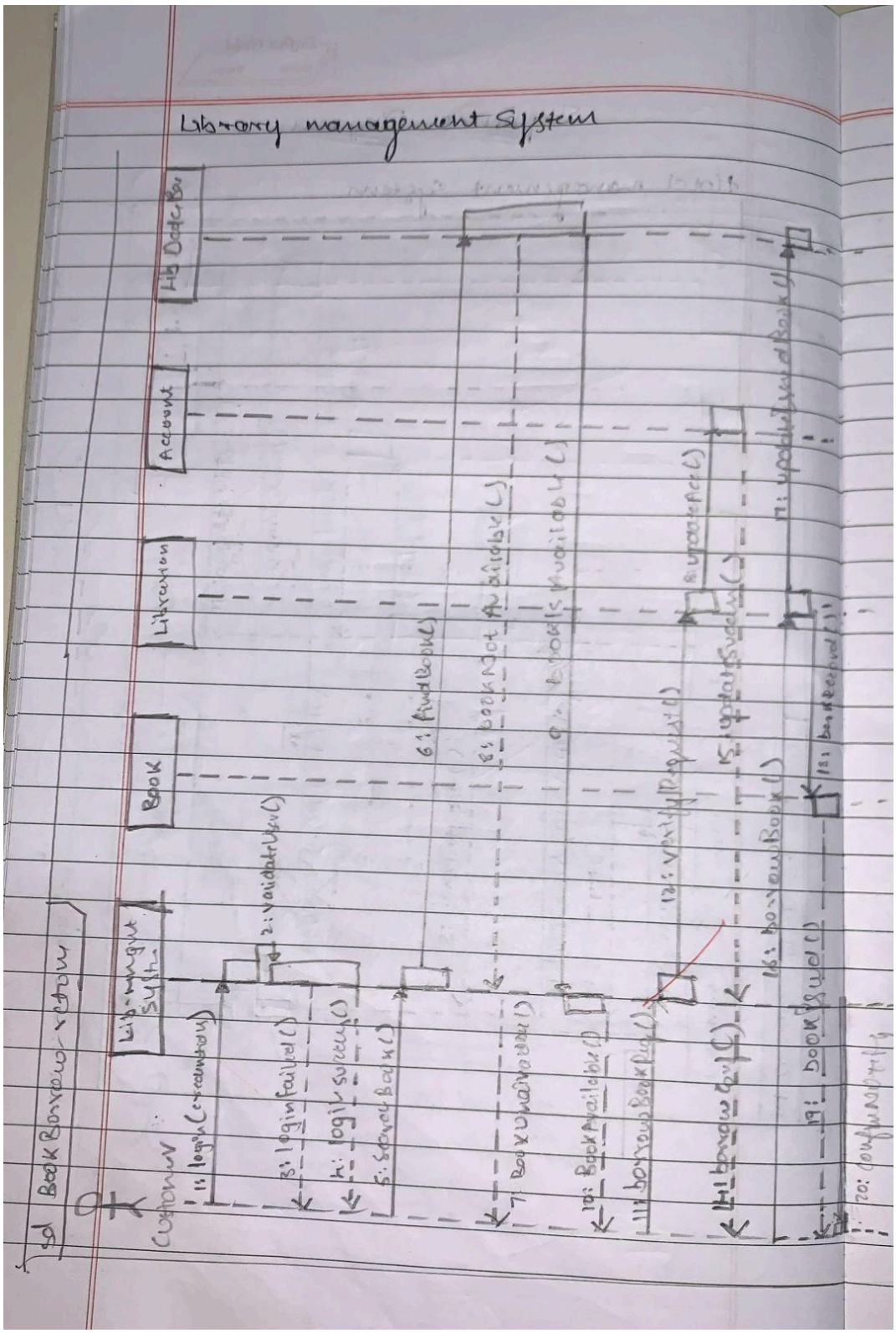

Fig.3.7: Advanced Sequence Diagram Observation

UseCase Diagram

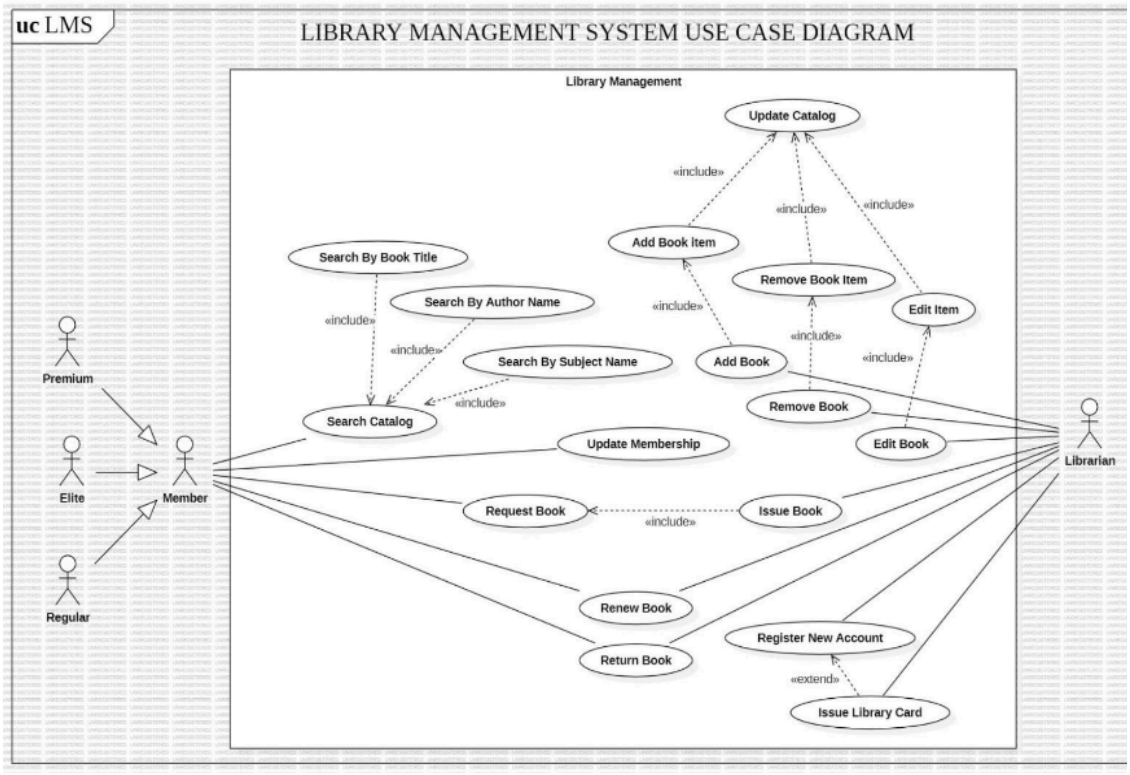

Fig.3.8: UseCase Diagram

Actors:

- Premium, Elite, and Regular Members: Can search the catalog, request books, renew books, return books, and update memberships.
- Librarian: Manages administrative tasks such as adding, removing, and editing book items, updating the catalog, issuing books, registering new accounts, and issuing library cards.

Use Cases:

- User-level functions include searching the catalog by title, author, or subject.
- Administrative functions involve catalog management, book issuance, and account registration.

Relationships:

- Includes relationships between use cases, e.g., "Search by Title" is a subset of "Search Catalog."
- Extension relationships like "Issue Library Card" extend from "Register New Account."

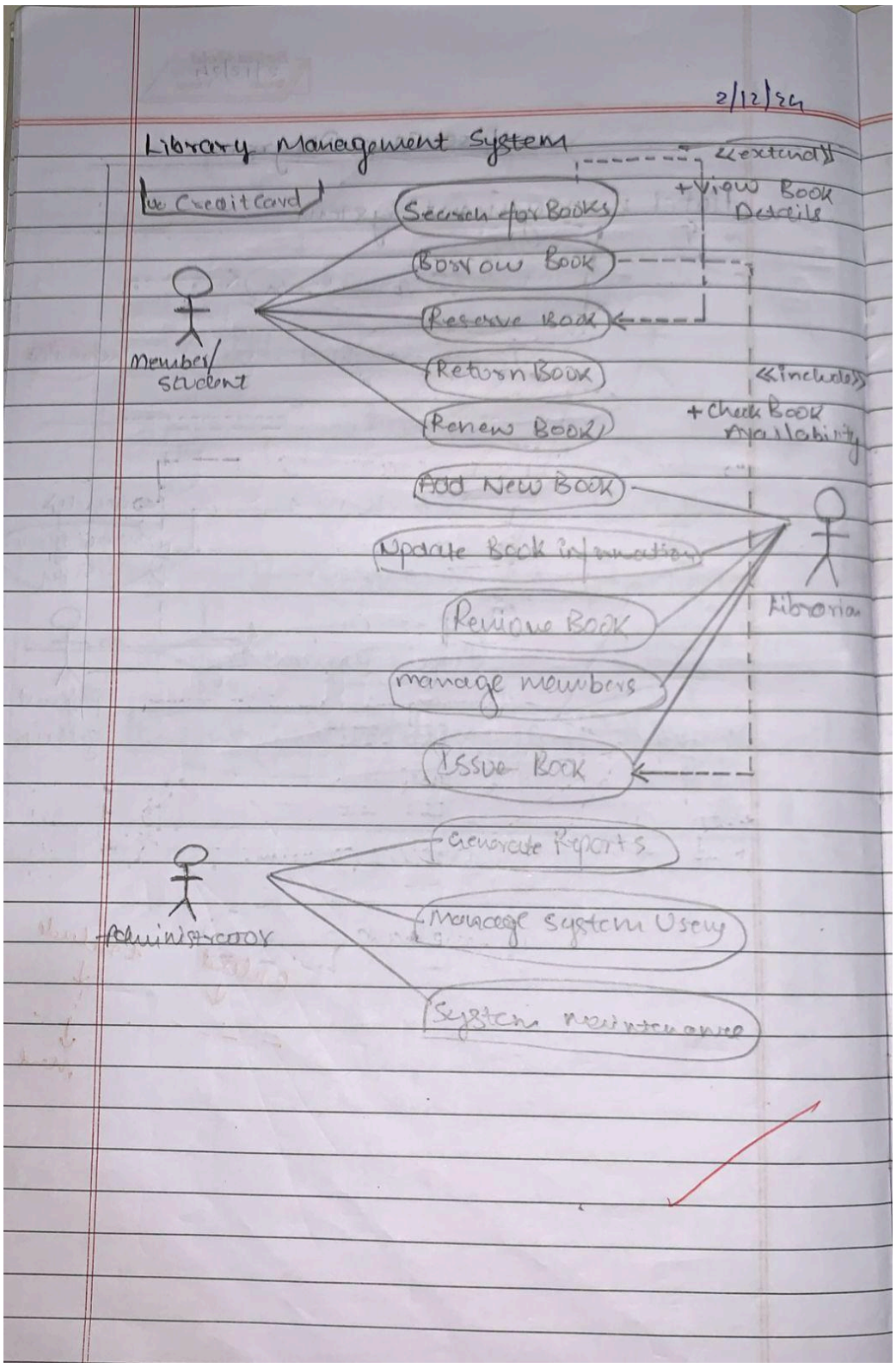

Fig.3.9: Use Case Diagram Observation

Activity Diagram

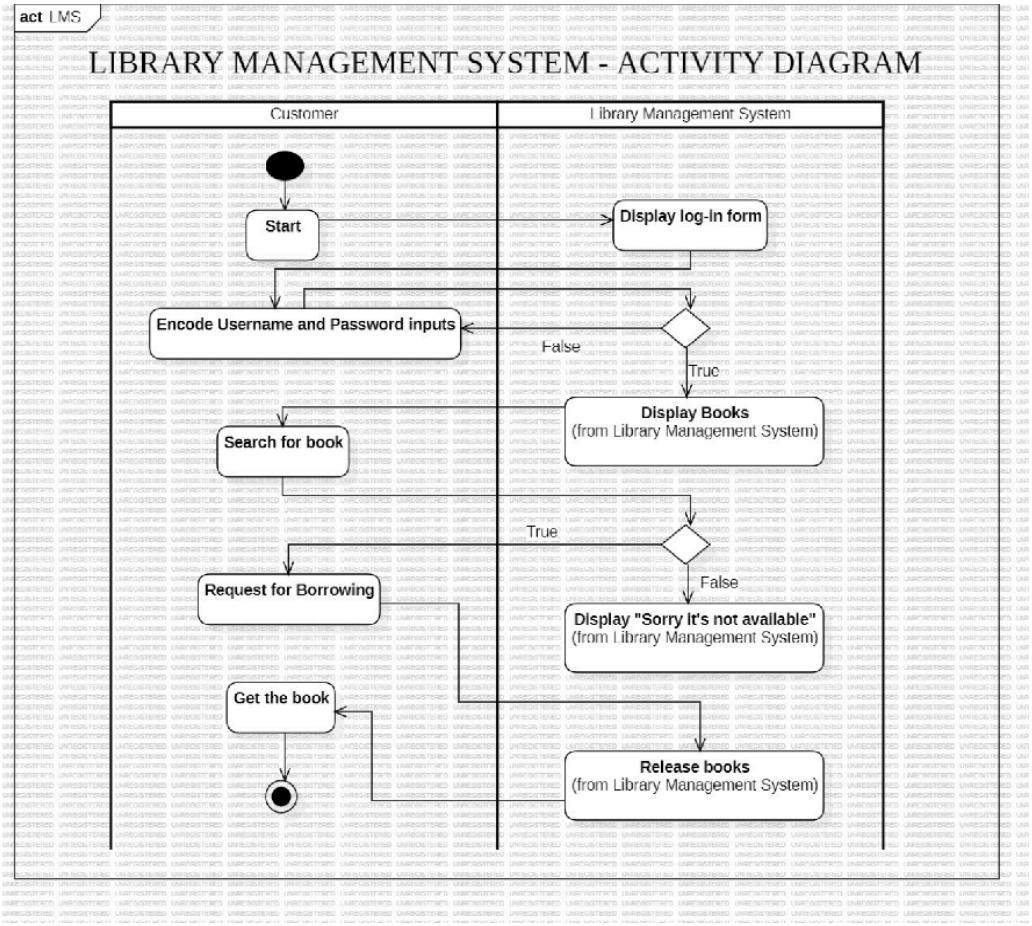

Fig.3.10: Activity Diagram

Swimlanes:

- Customer: Encodes credentials, searches for a book, and requests borrowing.
- Library Management System: Processes the log-in form, displays books, checks availability, and releases the book.

Flow:

- Starts with the customer entering credentials.
- The system validates the log-in and allows the user to search for a book.
- If the book is available, it processes the borrowing request and releases the book. If unavailable, it displays an error message.

Decisions: Branching based on log-in validity and book availability.

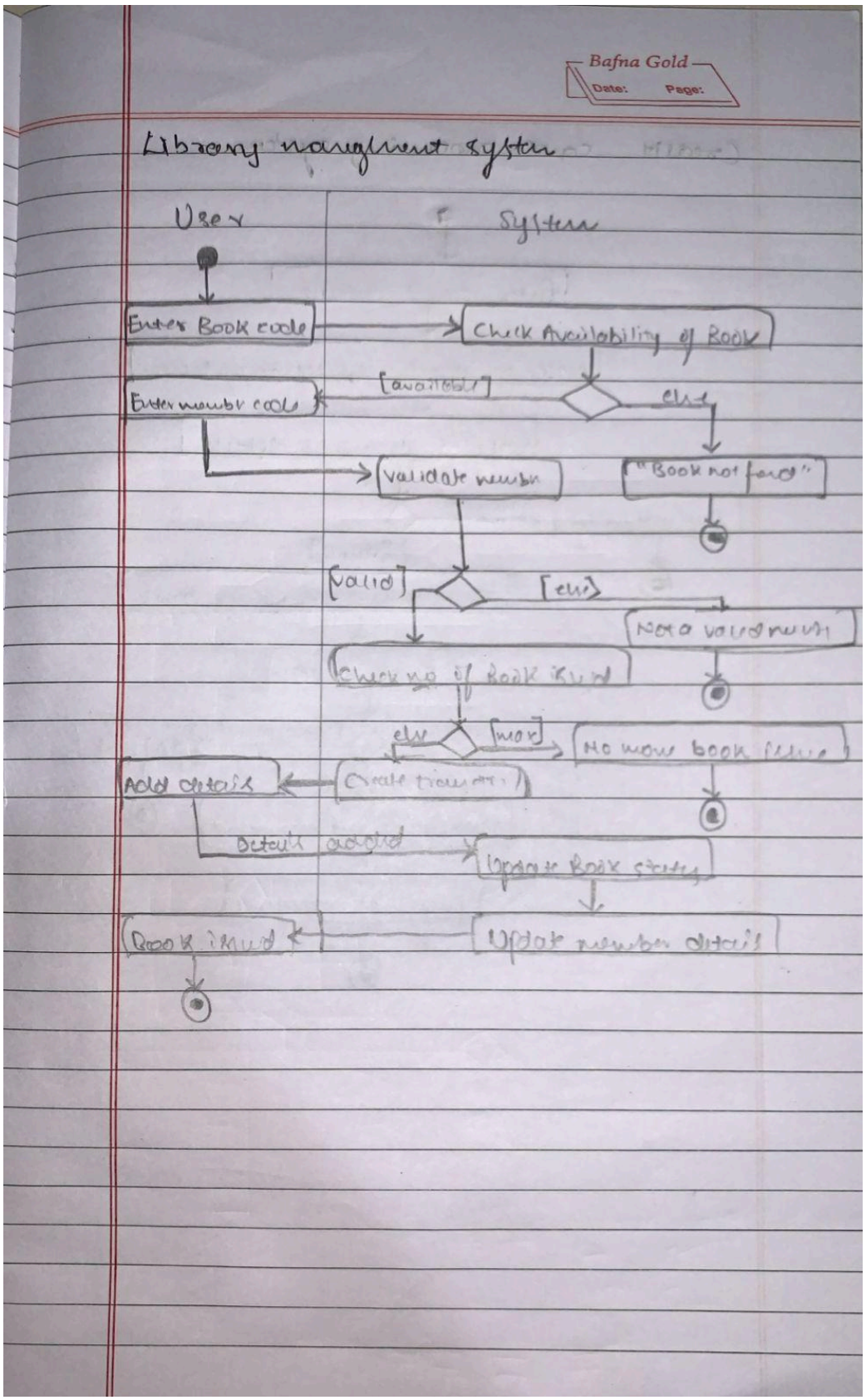

Fig.3.11: Activity Diagram Observation

4.Stock Maintenance System

SRS-Software Requirements Specification

1. Introduction

1.1 Purpose of this Document

This document outlines the software requirements for the Stock Maintenance System. It provides a detailed understanding of the functionalities, design constraints, and overall objectives of the software.

1.2 Scope of this Document

This document details the objectives, features, and user requirements for the Stock Maintenance System, addressing advanced development costs and timelines.

1.3 Overview

The Stock Maintenance System is designed to streamline inventory tracking, restocking, and reporting processes. This system aims to minimize stock shortages, reduce overstocking, and enhance overall operational efficiency.

2. General Description

- Objectives: Simplify stock management, ensure real-time inventory updates, and improve decision-making through analytics.
- User Characteristics: Warehouse staff, inventory managers, and business owners.
- Features: Inventory tracking, automated restocking alerts, supplier management, and reporting tools.
- Importance: The system is vital for businesses to maintain efficient stock control, minimize losses, and maximize profitability

.

3. Functional Requirements

- Inventory Tracking: Real-time tracking of stock levels with categorization.
- Restocking Alerts: Notifications for low stock levels based on predefined thresholds.
- Supplier Management: Maintain supplier records and manage purchase orders.
- Reporting: Generate reports on stock usage, trends, and supplier performance.
- Stock Adjustments: Allow adjustments for damaged, expired, or misplaced items.

4. Non-functional Requirements

- User Interface: Intuitive and user-friendly for ease of use by warehouse staff and managers.
- API Integration: Integrate with third-party systems like ERP software and supplier databases.
- Data: Real-time updates and synchronization of stock data.

5. Performance Requirements

- Response Time: The system should respond to user actions within 2 seconds.
- Availability: Ensure 99.9% uptime for critical stock management operations.
- Scalability: Support up to 500 concurrent users and 100,000 stock items.

6. Design Constraints

3. Must be compatible with existing hotel management software.
4. Should comply with data protection regulations.

7. Non-functional Attributes

4. Security: The data must be protected against unauthorized access.
5. Usability: Intuitive for staff and customers.
6. Maintainability: Scalable and upgradable.

8. Schedule & Budget

- Time Estimation: 6 months to 8 months
- Budget: $10,000

Class Diagram

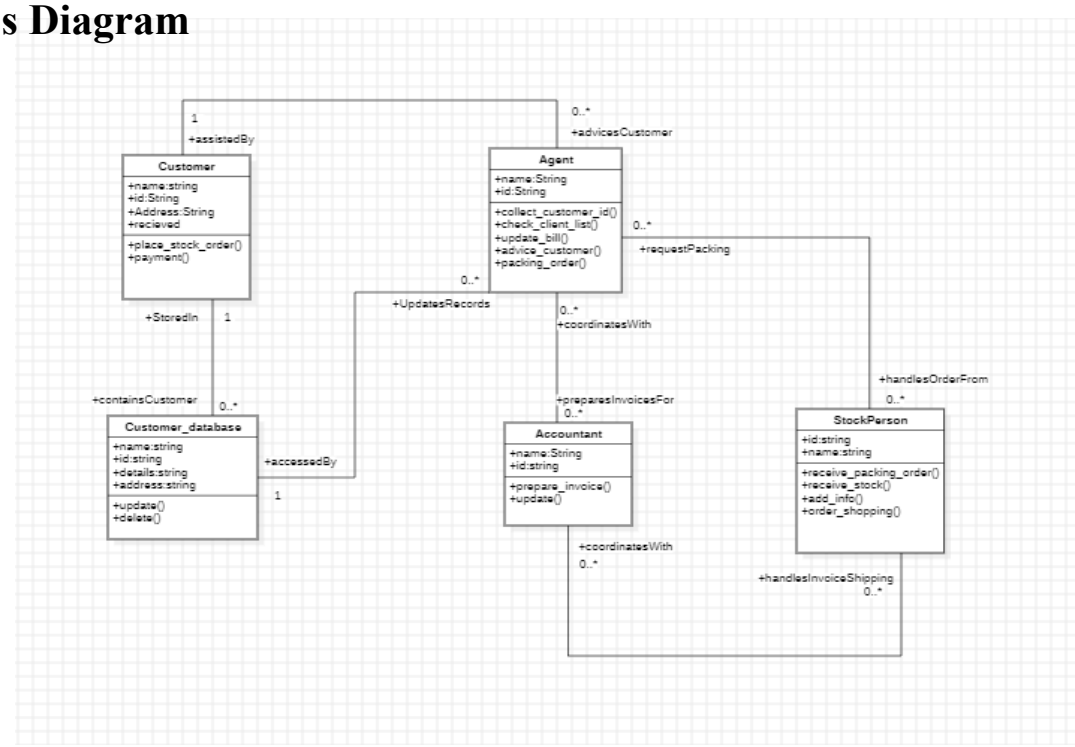

Fig.4.1: Simple Class Diagram

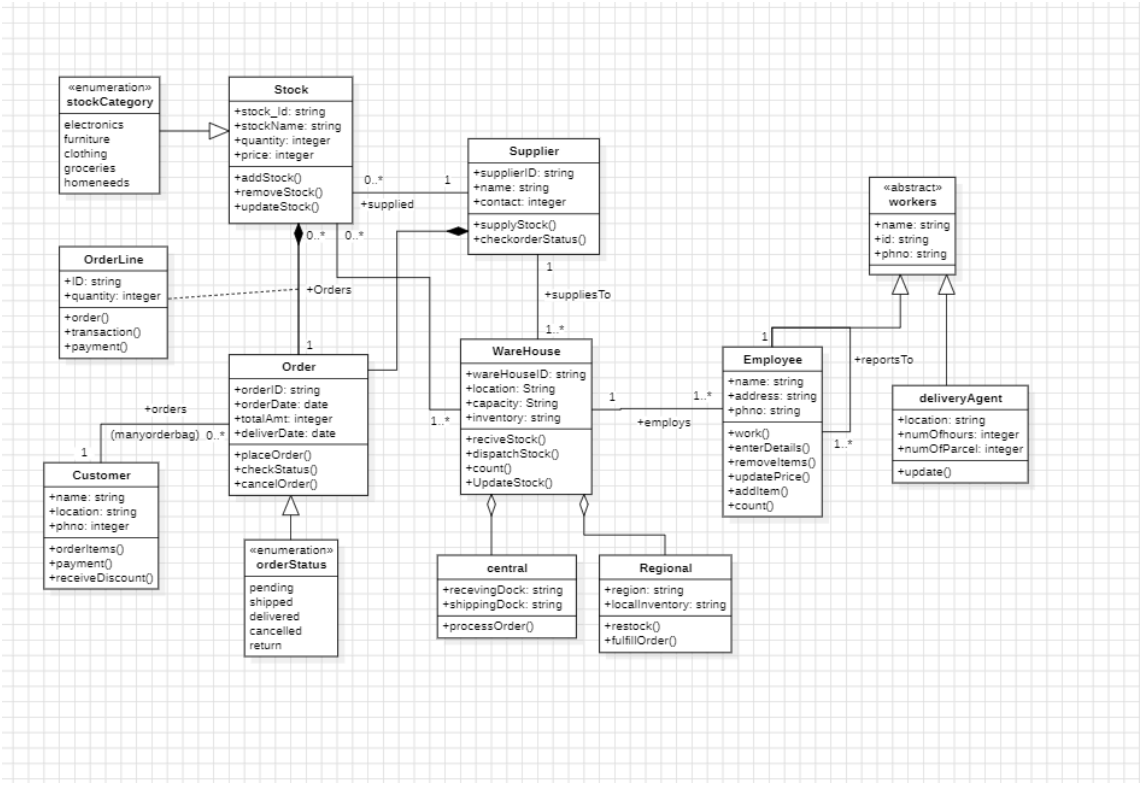

Fig.4.2: Advanced Class Diagram

Entities and Attributes:

1. Stock:
    - Attributes: stock_Id, stockName, quantity, price, stockCategory (enumeration: electronics, furniture, clothing, etc.).
    - Operations: Add, remove, or update stock.
    - Relationships: Supplied by Supplier, ordered in Order.
2. Supplier:
    - Attributes: supplierID, name, contact.
    - Operations: Supply stock, check order status.
    - Relationships: Supplies Stock.
3. Order:
    - Attributes: orderID, orderDate, totalAmt, deliverDate, orderStatus (enumeration: pending, shipped, delivered, etc.).
    - Operations: Place, check, or cancel orders.
    - Relationships: Contains multiple OrderLine items and relates to Customer.
4. OrderLine:
    - Attributes: ID, quantity.
    - Operations: Handle transaction and payment.
    - Relationships: Linked to Order.
5. Customer:
    - Attributes: name, location, phone.
    - Operations: Make payments, order items, and receive discounts.
    - Relationships: Places Order.
6. Warehouse:
    - Attributes: warehouseID, location, capacity, inventory.
    - Operations: Receive, dispatch, and update stock.
    - Relationships: Employs Employee, manages Stock.
7. Employee:
    - Attributes: name, address, phone.
    - Operations: Work, manage inventory, and update pricing.
    - Relationships: Employed by Warehouse.
8. DeliveryAgent:

    - Inherits from abstract class Workers.
    - Attributes: location, numOfHours, numOfParcel.
    - Operations: Update details.
9. Workers (Abstract Class):
    - Attributes: name, id, phone.
10. Central Warehouse:
    - Attributes: receivingDock, shippingDock.
    - Operations: Process orders.
    - Relationships: Subclass of Warehouse.
11. Regional Warehouse:
    - Attributes: region, localInventory.
    - Operations: Restock and fulfill orders.
    - Relationships: Subclass of Warehouse.

Key Relationships:

- Stock is supplied by Supplier and managed by Warehouse.
- Order contains multiple OrderLine items and is linked to a Customer.
- Employee works in Warehouse and manages Stock.
- DeliveryAgent handles parcel deliveries and reports to Warehouse.

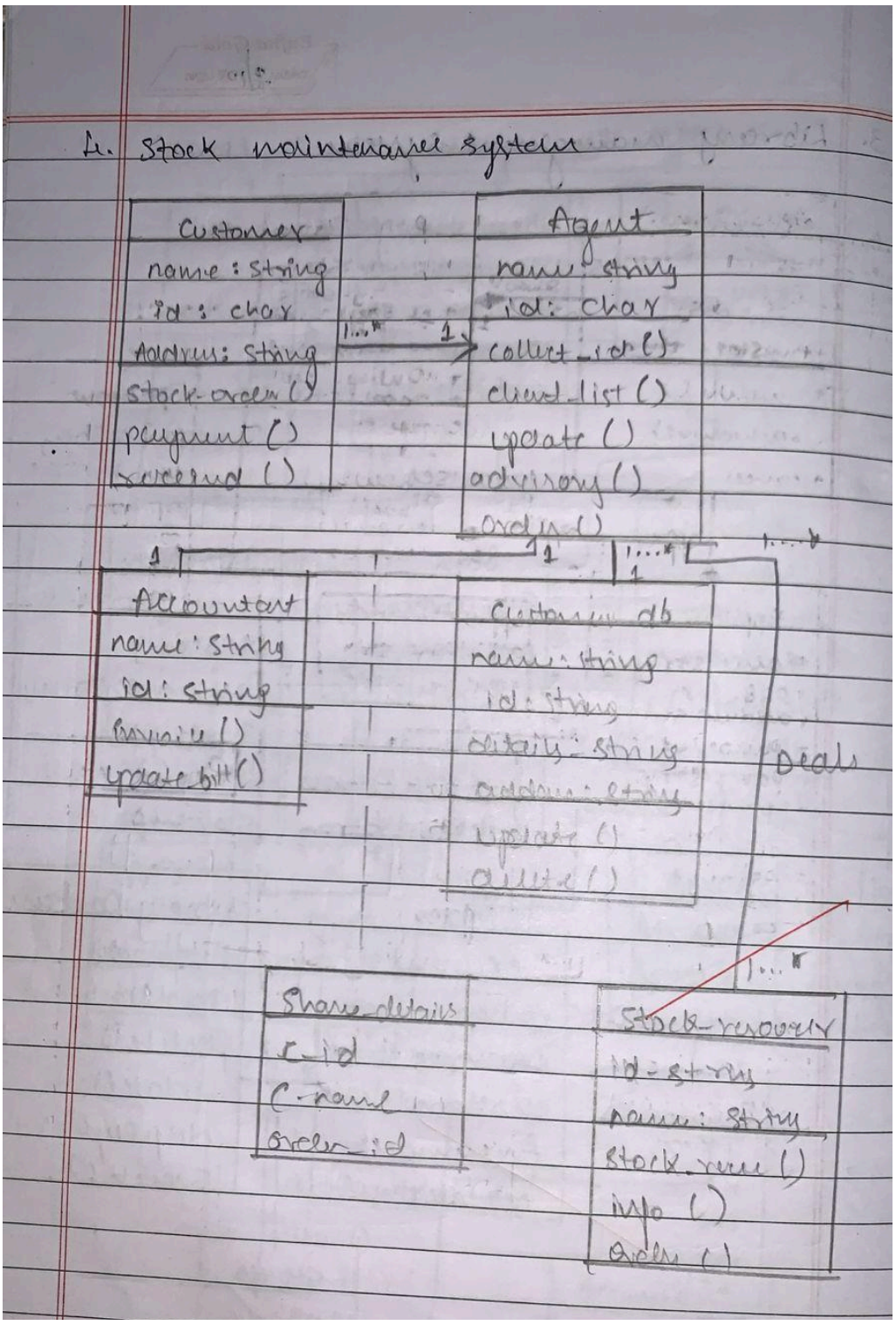

Fig.4.3: Advanced Class Diagram

State Diagram

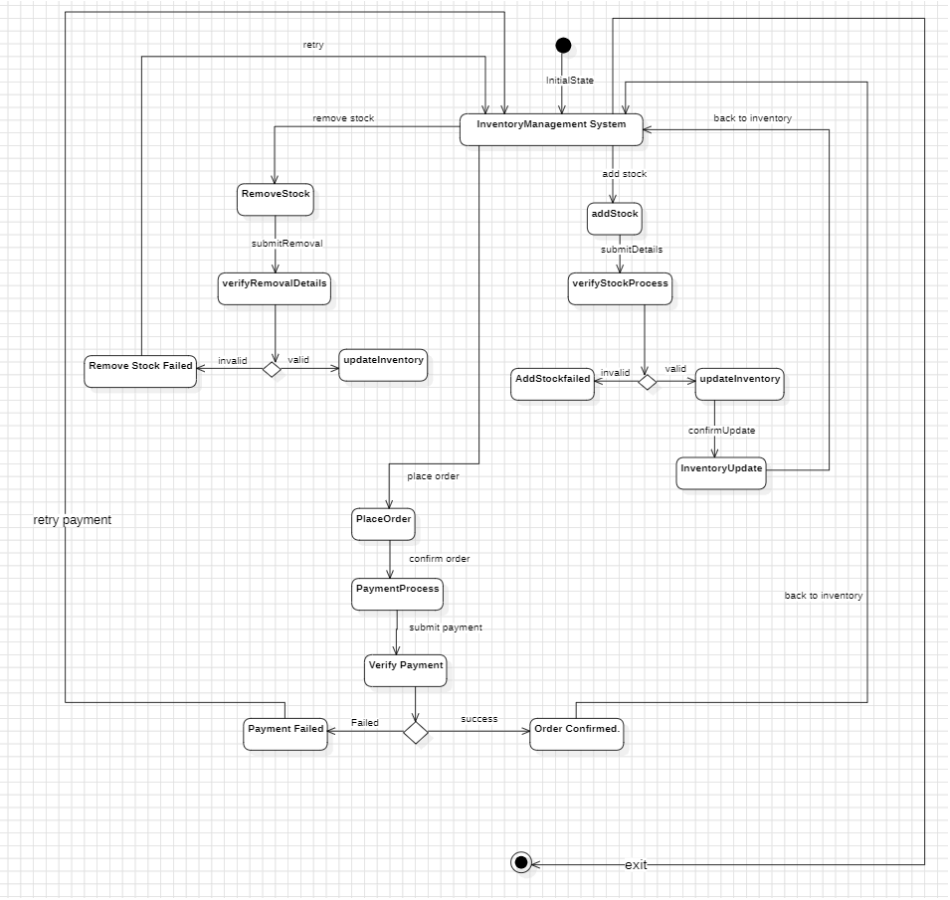

Fig 4.4: Simple state Diagram

Key Activities:

1. Initial State:
    - System begins in the InitialState and transitions to the Inventory Management System.
2. Add Stock:
    - Action: addStock submits details for validation.
    - If valid: Transitions to updateInventory.
    - If invalid: Leads to AddStockFailed.
3. Remove Stock:
    - Action: removeStock validates removal details.
    - If valid: Updates inventory.
    - If invalid: Moves to RemoveStockFailed.
4. Order Processing:

- PlaceOrder:
  - Initiates the order placement process.
- PaymentProcess:
  - Verifies payment after order confirmation.
  - If payment is successful: Transitions to Order Confirmed.
  - If payment fails: Returns to retry payment.

5. Inventory Update:
   - Updates the inventory after successful addition or removal of stock and order confirmation.
6. Exit:
   - Completes the workflow, ensuring inventory consistency and data finalization.

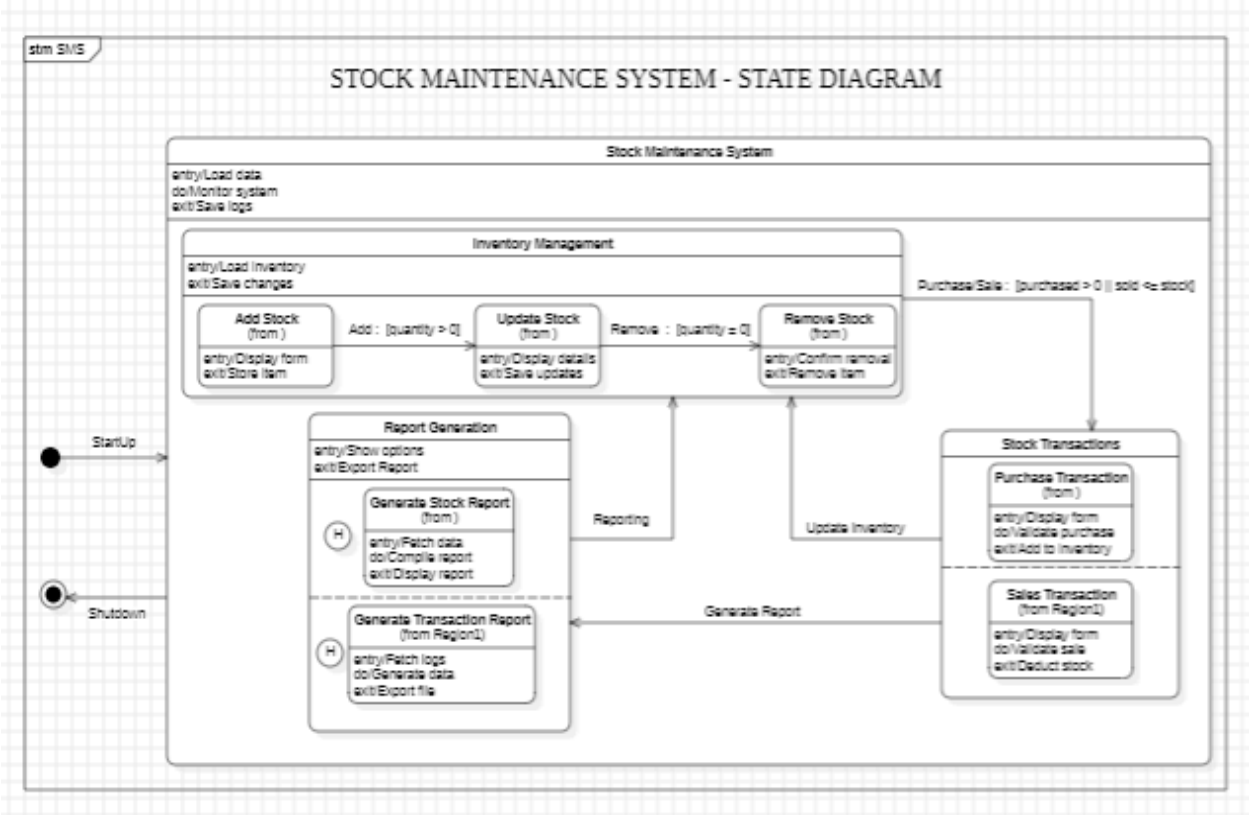

Fig.4.5: Advanced State Diagram

Key States and Actions:

1. Startup and Initialization:
    - The system starts by loading local inventory and configuration details.
    - Transitions to the Inventory Management state after initialization.
2. Inventory Management:
    - Add Stock:
        - Allows the user to add new stock with a specific quantity.
        - Displays a form for input and saves changes upon exit.
    - Update Stock:
        - Enables updating the quantity or details of existing stock.
        - Ensures data consistency by saving updates.
    - Remove Stock:
        - Provides functionality to remove stock items, confirming the removal before finalization.
3. Stock Transactions:
    - Handles purchase and sales transactions.
    - Verifies stock availability (purchased > 0 && sold <= stock) before updating inventory.
4. Report Generation:
    - Two types of reports can be generated:
        - Stock Report: Summarizes the current inventory details.
        - Transaction Report: Provides details of purchase and sales transactions.
    - Both processes allow exporting the generated reports.
5. Shutdown:
    - Finalizes operations, saves data, and transitions to the shutdown state.

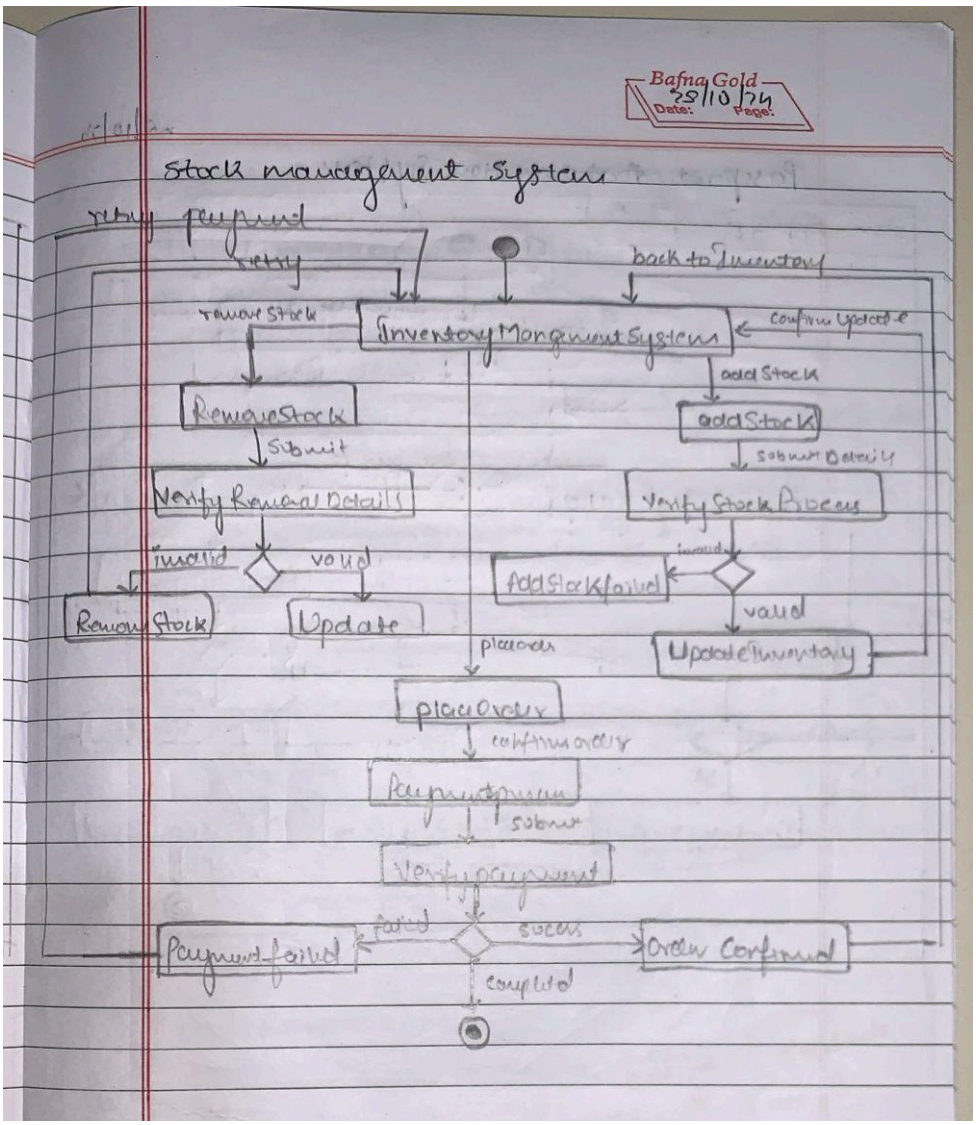

Fig.4.6: Advanced State Diagram Observation

UseCase Diagram

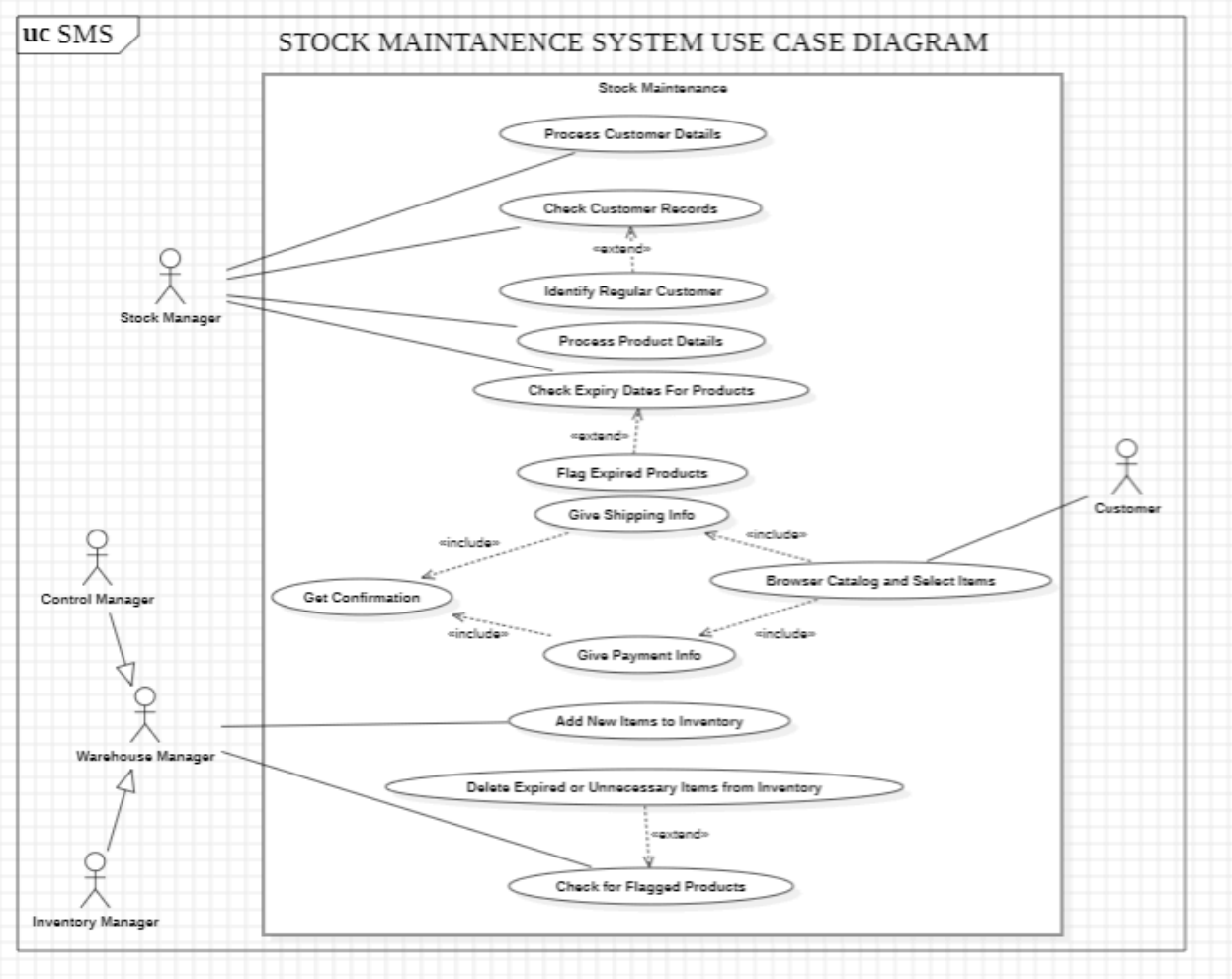

Fig.4.7: Use-Case Diagram

Purpose: Depicts the functional requirements of the Stock Maintenance System and its interactions with various actors.

Actors:

- Stock Manager: Handles customer and product-related tasks.
- Customer: Browses catalog, selects items, and completes purchases.
- Control Manager: Confirms shipping and payment details.
- Warehouse Manager: Manages inventory.
- Inventory Manager: Reviews and removes expired or flagged products.

Use Cases:

- Process Customer Details: Includes checking customer records and identifying regular customers.
- Process Product Details: Includes verifying product expiry and flagging expired items.
- Inventory Management:
    - Adding new items.
    - Deleting expired or unnecessary items based on flagged products.
- Customer Interaction:
    - Browsing the catalog and selecting items.
    - Providing payment and shipping details.

Relationships:

- Includes: Indicates mandatory sub-tasks (e.g., shipping info is part of order processing).
- Extends: Denotes optional actions triggered by specific conditions (e.g., flagging expired products).

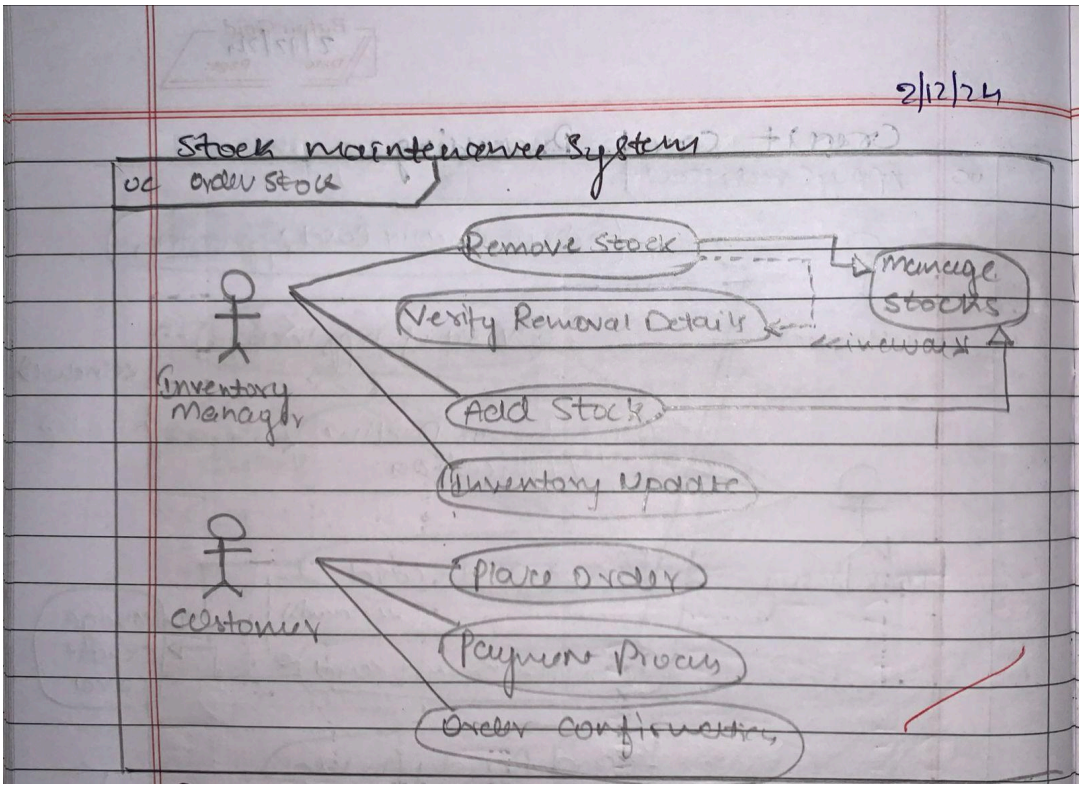

Fig.4.8: Use-Case Diagram Observation

Sequence Diagram

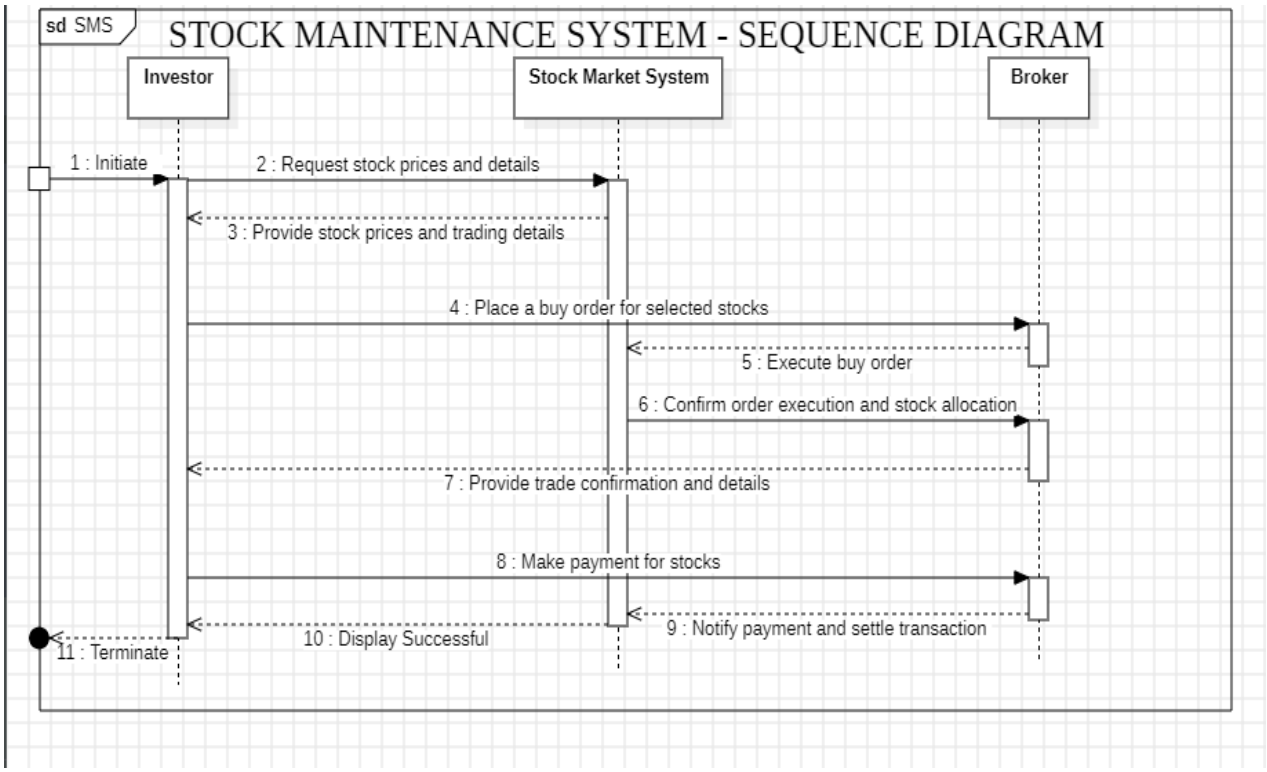

Fig.4.9: Sequence Diagram

Purpose: Illustrates the dynamic flow of processes during a stock trading transaction between the system and users.

Actors:

- Investor: Initiates and places orders for stocks.
- Stock Market System: Processes requests, provides stock prices, and manages orders.
- Broker: Executes buy orders and confirms trades.

Process Steps:

1. Initiate: Investor begins the interaction.
2. Request Stock Prices: Investor requests stock details from the system.
3. Provide Details: System responds with prices and trading details.
4. Place Buy Order: Investor selects stocks to purchase.
5. Execute Order: Broker processes the buy order.
6. Confirm Execution: Broker confirms the order and allocates stocks.

7. Provide Confirmation: System communicates trade details to the investor.
8. Make Payment: Investor processes payment for the purchased stocks.
9. Settle Transaction: Broker handles payment settlement.
10. Display Success: System confirms a successful transaction.
11. Terminate: Interaction ends.

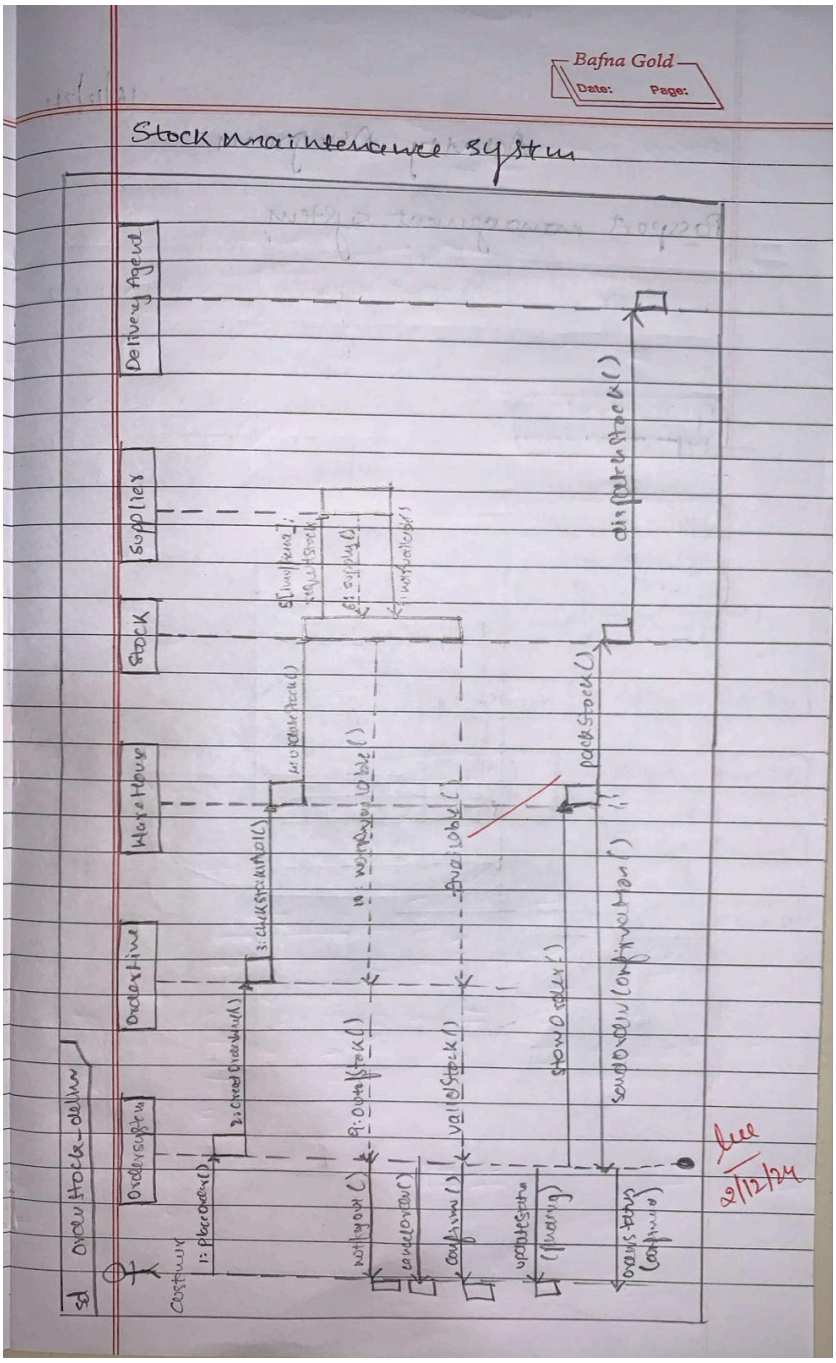

Fig.4.10: Sequence Diagram Observation

Activity Diagram

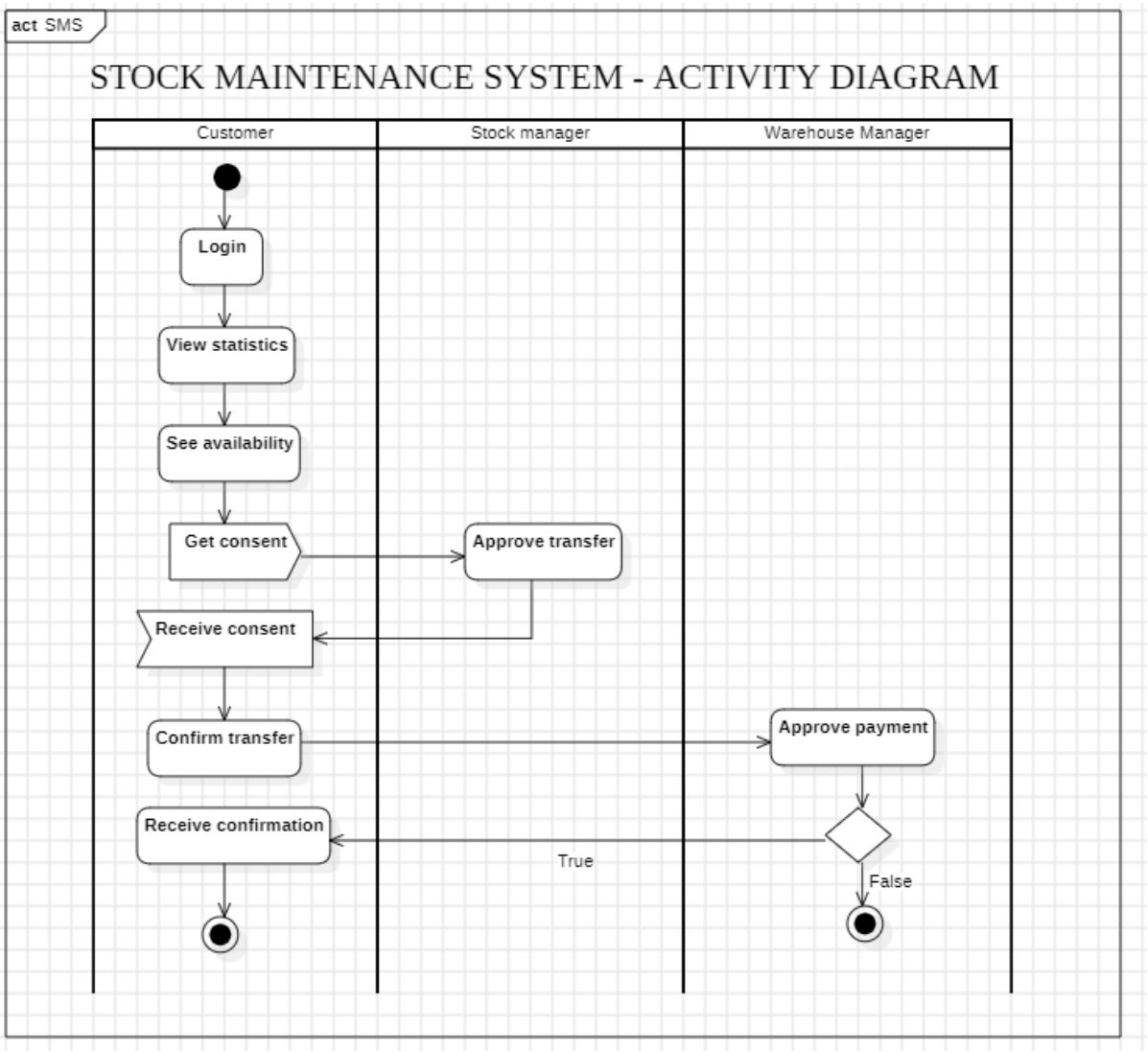

Fig.4.11: Advanced Activity Diagram

Customer:

- Logs in to the system.
- Views statistics and availability of stocks.
- Requests consent for a stock transfer.
- Confirms the transfer after receiving consent.
- Receives final confirmation of the transfer.

Stock Manager:

- Reviews and approves the transfer requested by the customer.

Warehouse Manager:

- Approves payment after evaluating the transaction.
- If payment is approved, the process completes. If not, the process terminates.

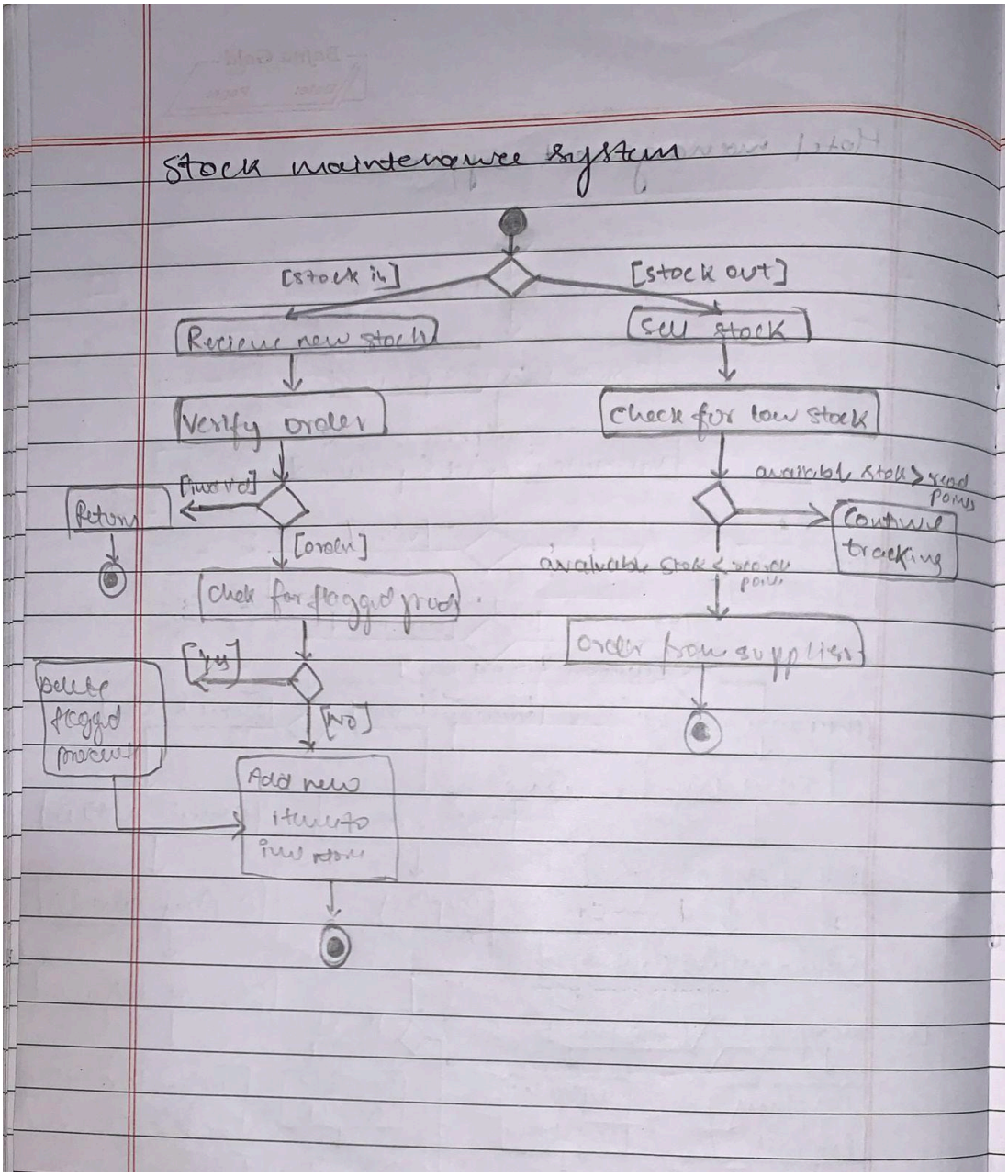

Fig.4.12: Advanced Activity Diagram Observation

5.Passport Automation System

SRS-Software Requirements Specification

1. Introduction

1.1 Purpose of this Document

This document outlines the software requirements for the Passport Automation System. It provides a clear understanding of the functionalities, design constraints, and overall objectives of the system.

1.2 Scope of this Document

This document details the objectives, features, and user requirements for the Passport Automation System, addressing development timelines and cost estimation for its implementation.

1.3 Overview

The Passport Automation System will provide an efficient solution for managing passport applications, renewals, and verifications. The system aims to minimize manual intervention, enhance processing speed, and improve user satisfaction by offering a seamless and automated service.

2. General Description

Objectives:

Simplify passport application and renewal processes.

Enhance customer experience through a user-friendly portal.

Ensure faster processing and document verification.

Characteristics of Users:

- Applicants: Individuals applying for new passports or renewals.
- Government Staff: Officials involved in verification and processing.

- System Administrators: Responsible for maintaining and monitoring the system.

3. Functional Requirements

1. Application Submission:
    - Allow users to fill out passport application forms online.
    - Upload required documents (e.g., ID proof, address proof).
2. Status Tracking:
    - Enable users to check the status of their application in real time.
3. Appointment Scheduling:
    - Facilitate scheduling of biometric and document verification appointments.
4. Document Verification:
    - Automated validation of uploaded documents against the database.
5. Payment Processing:
    - Integrate multiple payment options (credit/debit cards, UPI, net banking).
6. Notification System:
    - Notify users via email/SMS about application status, appointments, and approvals.
7. Passport Issuance:
    - Generate and issue passports after successful verification.
8. Reports & Analytics:
    - Provide reports on application statistics, revenue, and processing timelines.

4. Non-functional Requirements

1. User Interface: Must be simple and adaptable for staff & mobile interfaces via application for customers.
2. API Integration: Integrate with third-party services for payment gateways & other systems.
3. Data: Real-time updates on room availability.

4. Non-functional Requirements

1. User Interface:
    - Intuitive and responsive web portal with multi-language support.
2. API Integration:

- Integrate with government databases for ID validation (e.g., Aadhaar, Social Security).

3. Data:
    - Real-time updates on application status and appointment schedules

5. Performance Requirements

- Response Time: The system should respond to user requests within 3 seconds.
- Availability: 24/7 uptime for critical services.
- Scalability: Must handle up to 5000 concurrent users.

6. Design Constraints

- Compatibility with existing government systems.
- Compliance with data protection and privacy regulations.
- Support for both web-based and mobile-based platforms.

7. Non-functional Attributes

1. Security:
    - Ensure data encryption during transmission and storage.
    - Prevent unauthorized access to sensitive information.
2. Usability:
    - Designed for ease of use by citizens of all age groups.
3. Maintainability:
    - The system should be scalable for future enhancements and upgrades.

8. Schedule & Budget

- Time Estimation: 8 months to 10 months for development and deployment.
- Budget: $15,000 - $20,000

CLASS DIAGRAM

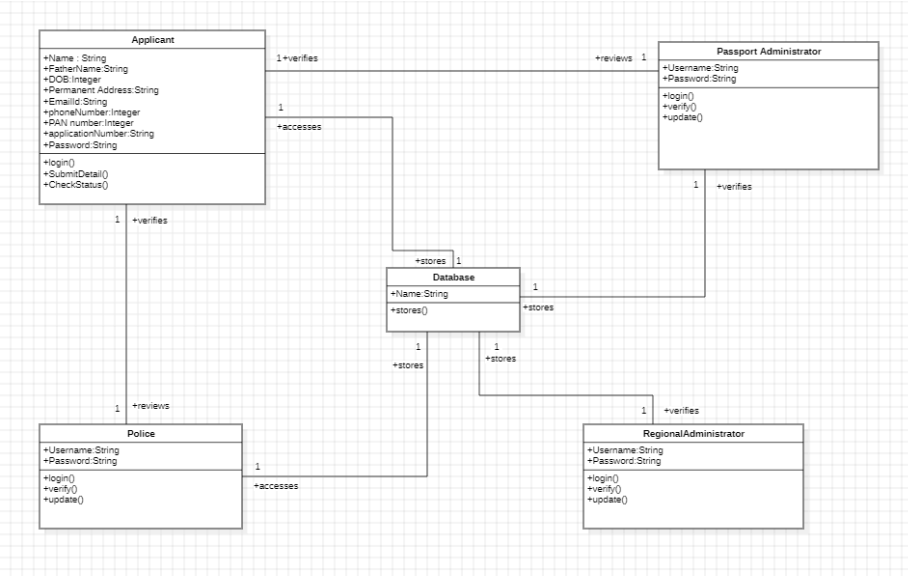

Fig.5.1: Simple Class Diagram

Classes and Attributes:

- Focuses on core entities like Applicant, Database, Police, Passport Administrator, and Regional Administrator.
- Attributes include basic details like username, password, name, and contact information.

Methods:

- Shared methods like login(), verify(), and update() highlight the shared responsibilities across different roles.

Relationships:

- Shows interactions between the Applicant and other system components like the Database, Police, and administrators.

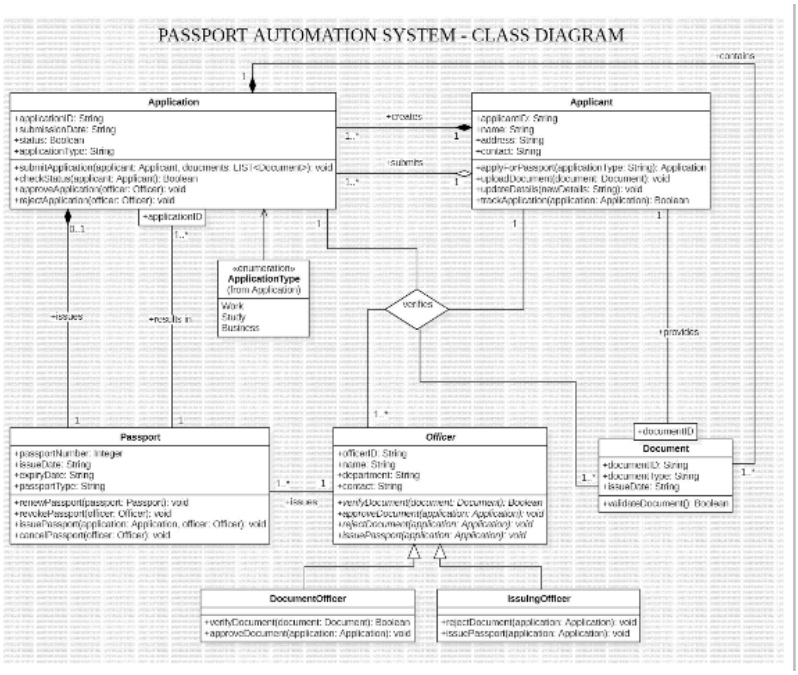

Fig 5.2: Advanced Class Diagram

Classes and Relationships:

- Application: Manages passport applications, including submission, status checks, approvals, and rejections.
- Applicant: Represents individuals applying for passports, with attributes like name, address, and contact details.
- Officer: Handles the verification, approval, and issuance of documents.
- Passport: Represents the issued passport, detailing its issue date, expiry, and other attributes.
- Document: Defines documents related to the application process, with functionality for validation.
- Specialized roles such as DocumentOfficer and IssuingOfficer inherit from the Officer class.

Enumeration:

- ApplicationType: Specifies the type of applications (Work, Study, Business).

Functional Scope:

- Focuses on methods like approveApplication(), uploadDocument(), and trackApplication() to define responsibilities and interactions.

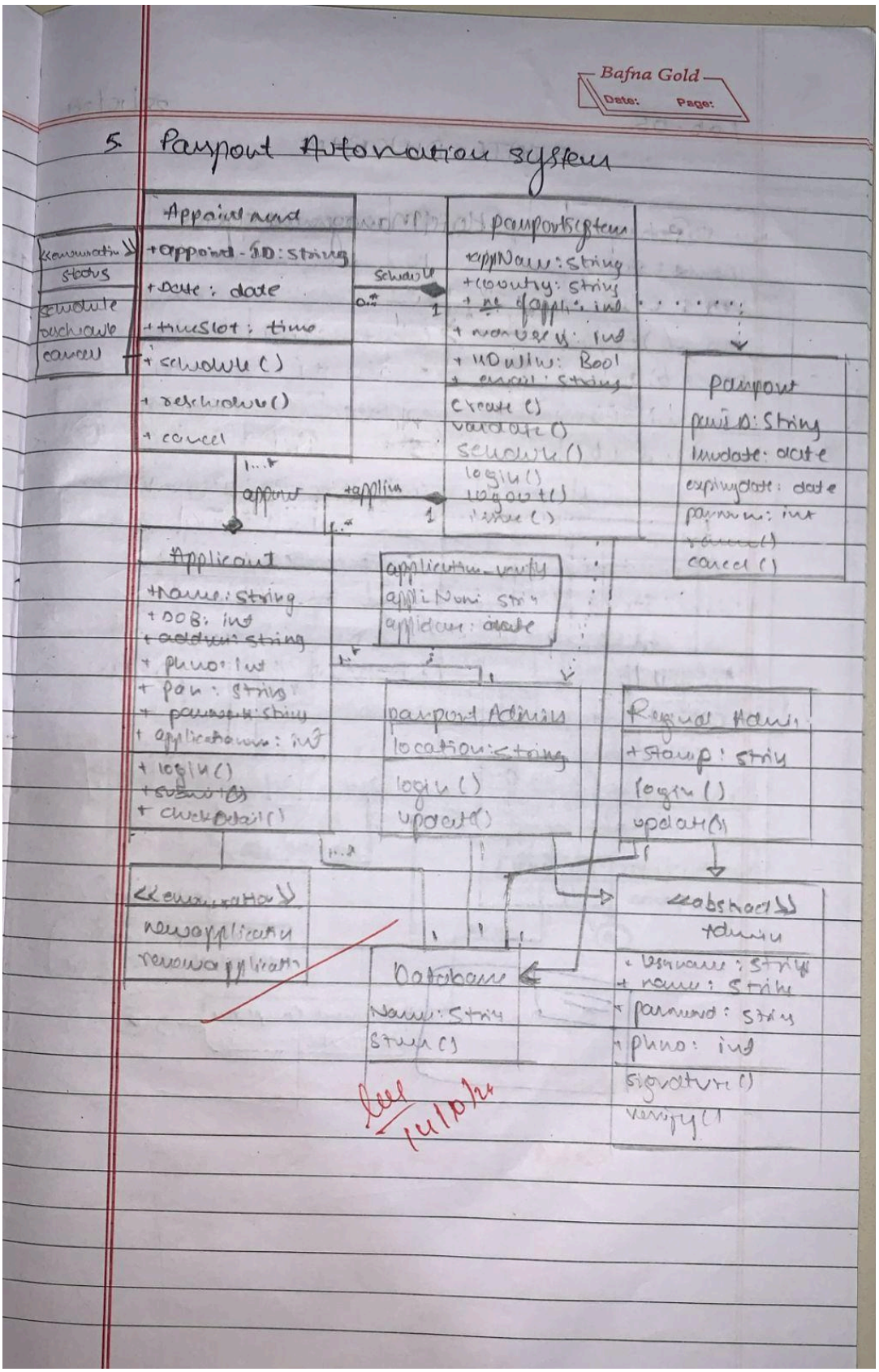

Fig 5.3: Advanced Class Diagram Observation

STATE DIAGRAM

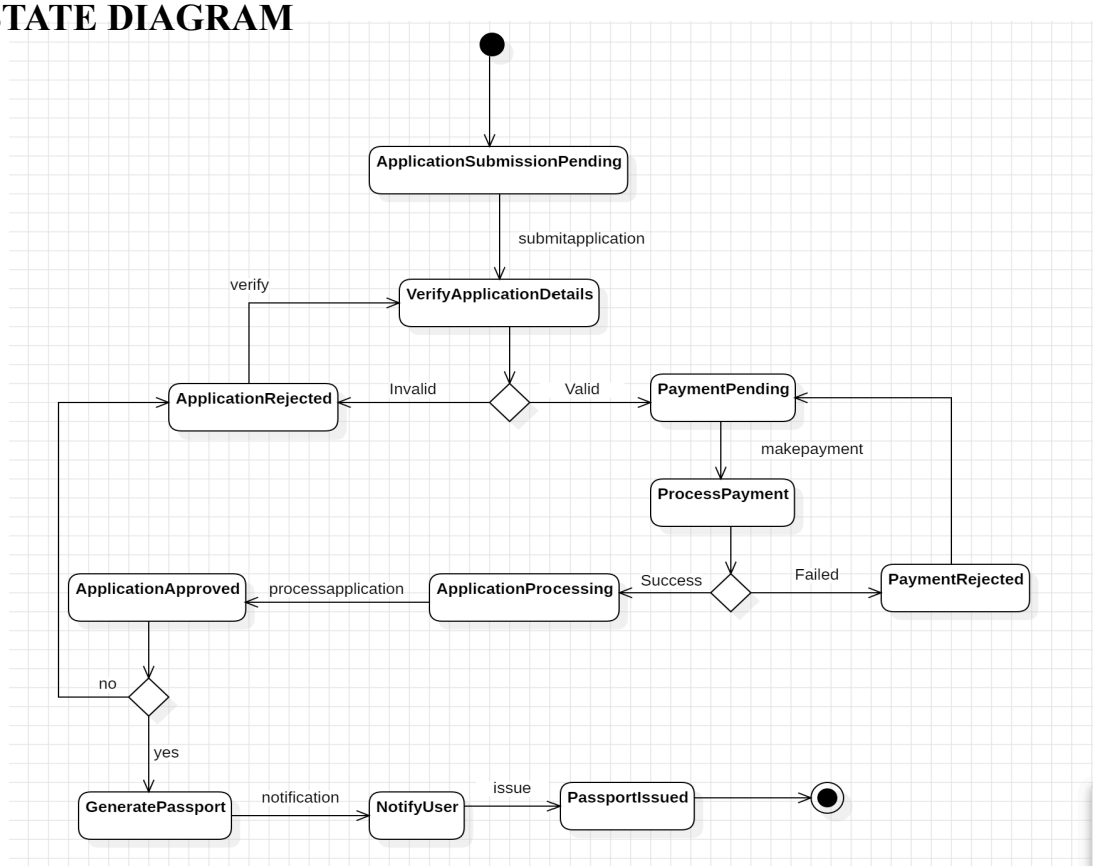

Fig.5.4: Simple State Diagram

ApplicationSubmissionPending: Initial state where the application is yet to be verified.
VerifyApplicationDetails: Validates application details.

- If invalid, transitions to ApplicationRejected.
- If valid, move to PaymentPending.

Payment Process:

- Successful payment transitions to ApplicationProcessing.
- Failed payment leads to PaymentRejected.

Approval and Issuance:

- If the application is approved, the system generates the passport and notifies the user.

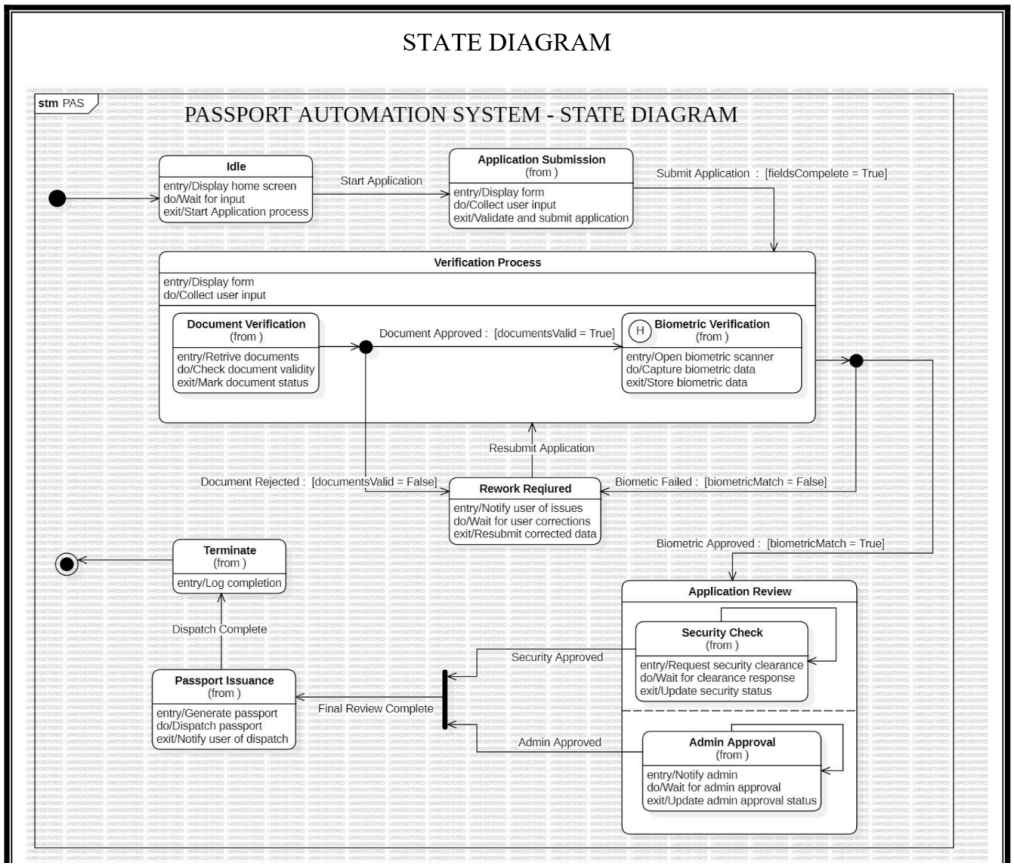

Fig.5.5: Advanced State Diagram

Idle State: Initial state where the system waits for user input to start the application process.
Application Submission: Users submit their applications with form details.
Verification Process:

- Document Verification: Verifies the validity of the documents submitted by the user.
- Biometric Verification: Checks biometric data for accuracy and compliance.
- Transition conditions such as documentsValid or biometricMatch determine success or failure.

Rework Required: If errors are detected (e.g., invalid documents), the user must resubmit.
Application Review:

- Includes stages like Security Check and Admin Approval.

Passport Issuance: After final approval, the passport is dispatched.
Termination State: Marks the end of the process if the application fails.

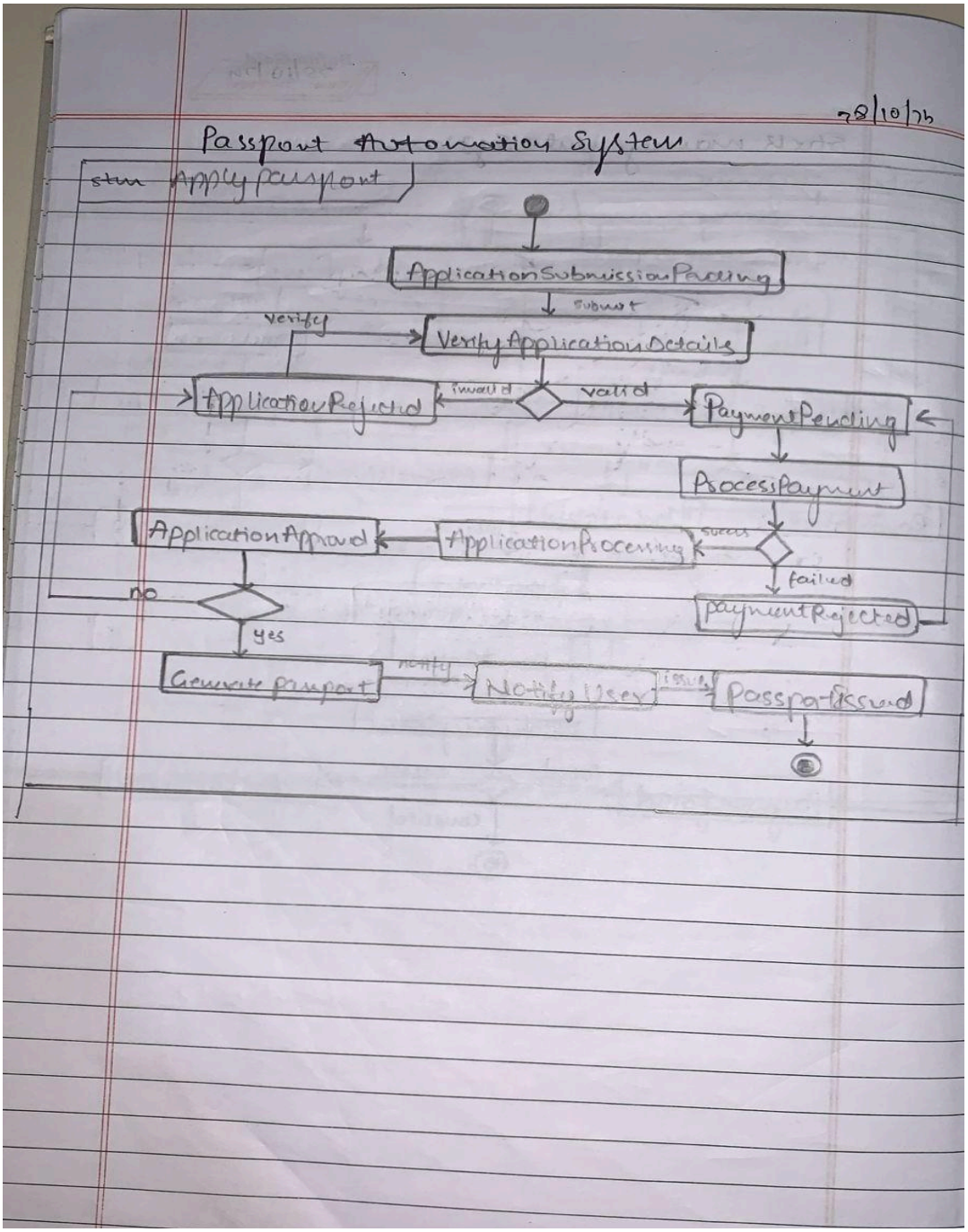

Fig.5.6: Advanced State Diagram Observation

SEQUENCE DIAGRAM

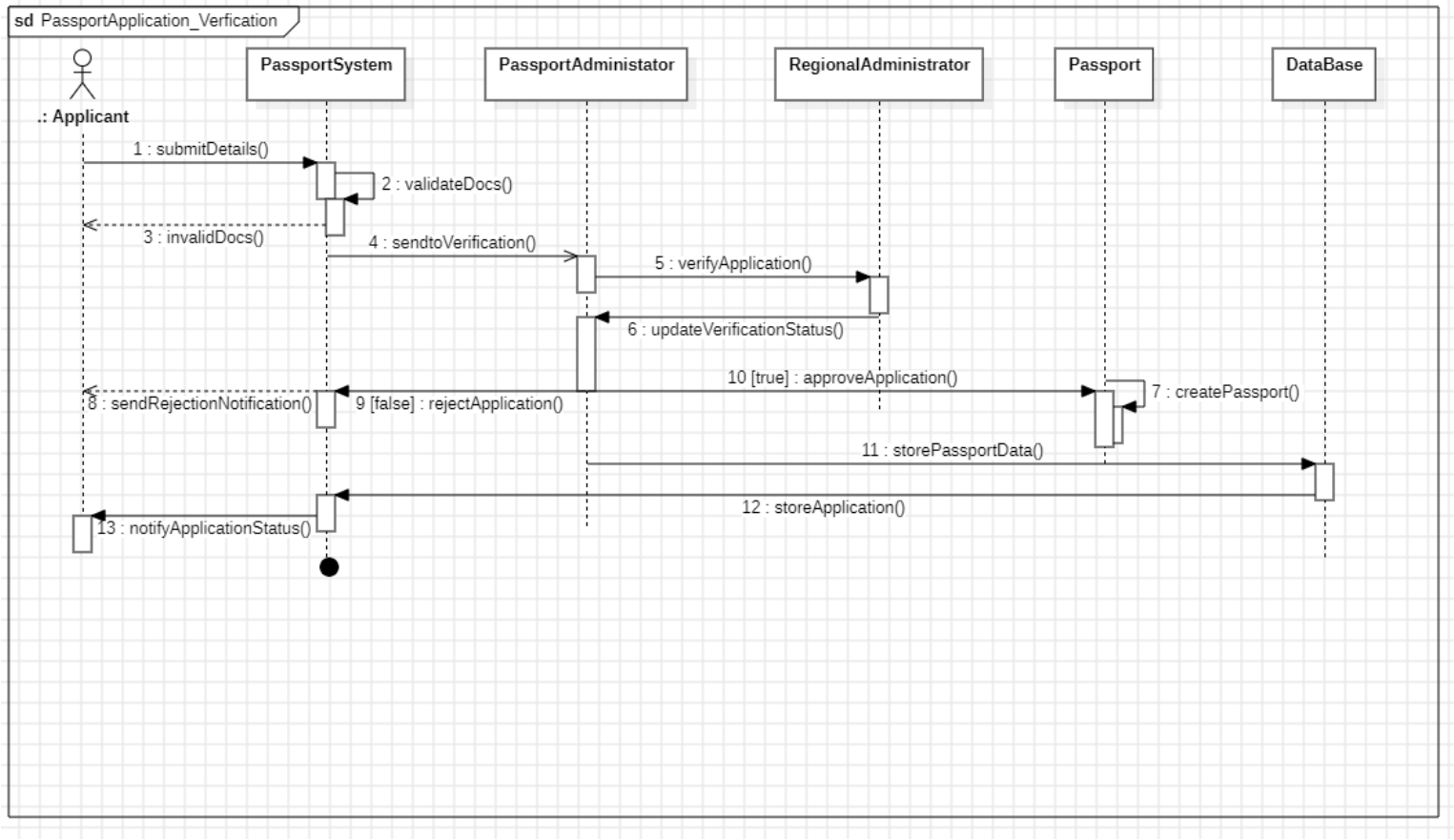

Fig.5.7: Advanced Sequence Diagram

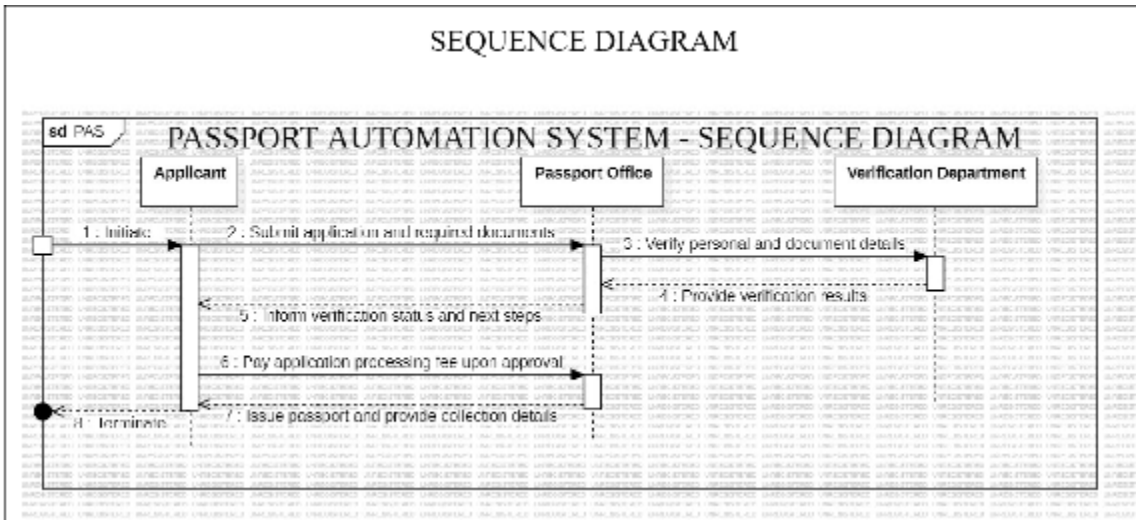

Fig.5.8: Simple Sequence Diagram

- Applicant: Initiates the application process by submitting details and documents.
- Passport Office: Receives the application and verifies the personal and document details.
- Verification Department: Performs the verification process and provides feedback to the Passport Office.
- Passport Office: Informs the applicant of the verification status and further steps.
- Applicant: Upon approval, pays the application processing fee.
- Passport Office: Issues the passport and provides collection details to the applicant.

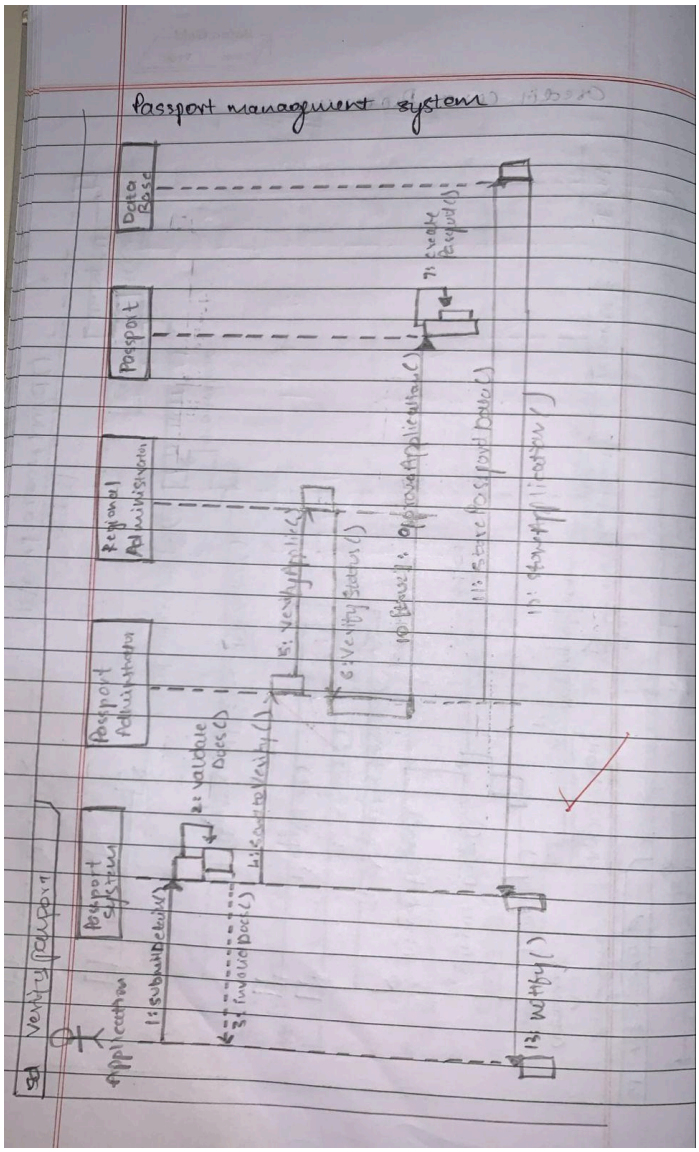

Fig.5.9: Advanced Sequence Diagram Observation

USE CASE DIAGRAM

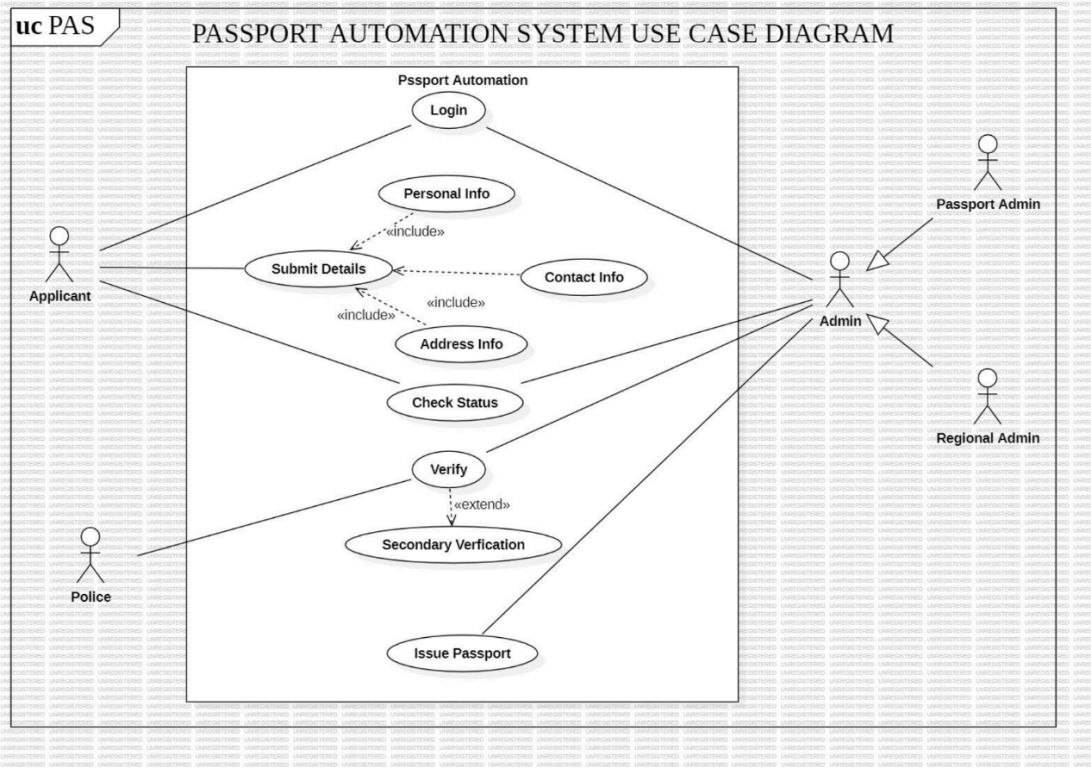

Fig.5.10: use case diagram

- Applicant is the primary actor, interacting with the system through various use cases:
    - Login: Authenticate to access the system.
    - Submit Details: Provide personal, contact, and address information.
    - Check Status: Track the progress of the application.
- Admin and Regional Admin roles manage system operations, including:
    - Verify: Verify the applicant's information.
    - Secondary Verification: Perform additional verification steps.
    - Issue Passport: Process and issue the passport.
- Passport Admin role manages the system's administration and user accounts.
- Police are involved in verification processes, possibly for background checks.

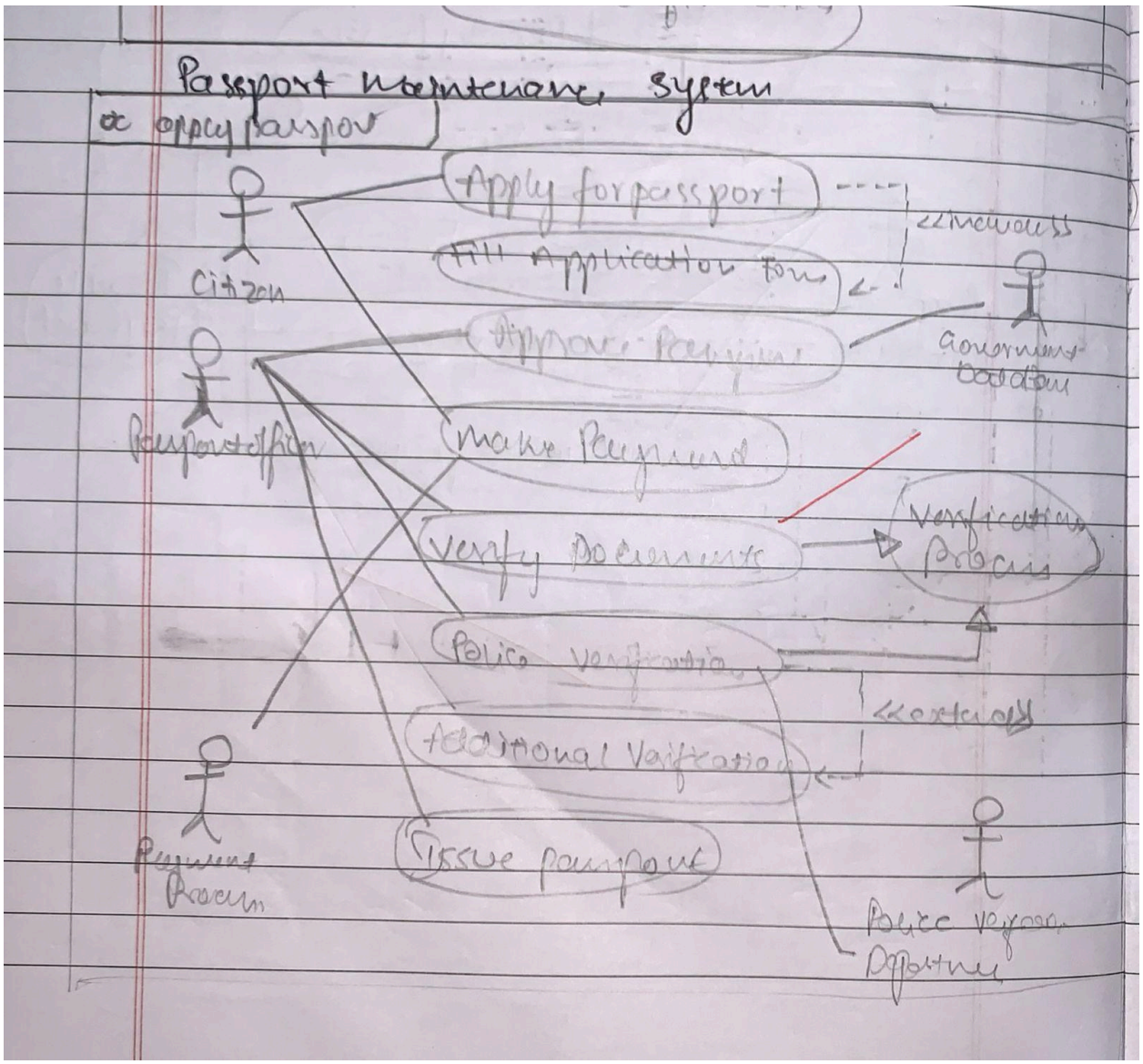

Fig.5.11: Use case diagram

ACTIVITY DIAGRAM

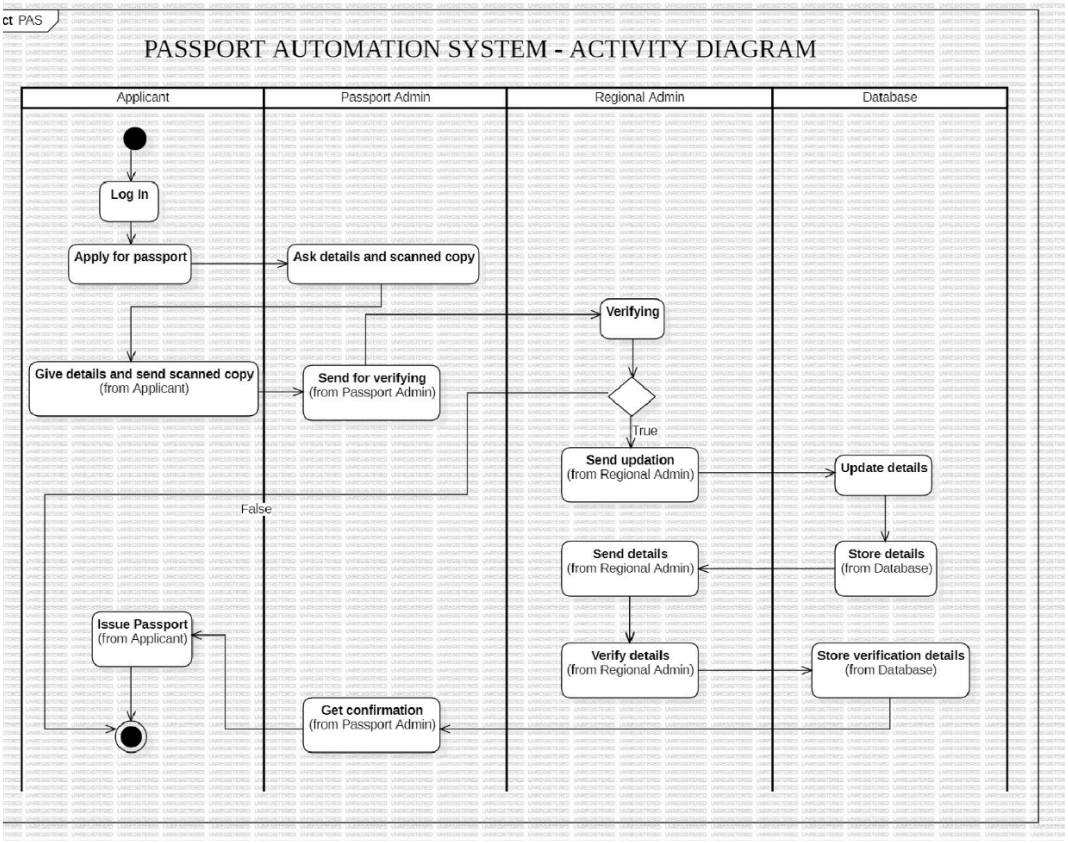

Fig.5.12: Activity Diagram

- Applicant initiates the process by logging in and applying for a passport.
- Passport Admin receives the application and asks for details and scanned copies.
- Verification takes place, where the provided details are verified.
- Regional Admin processes the application and updates the applicant's details.
- The passport is issued to the applicant, and a confirmation is sent.
- The diagram highlights the key steps involved in the entire passport application process.

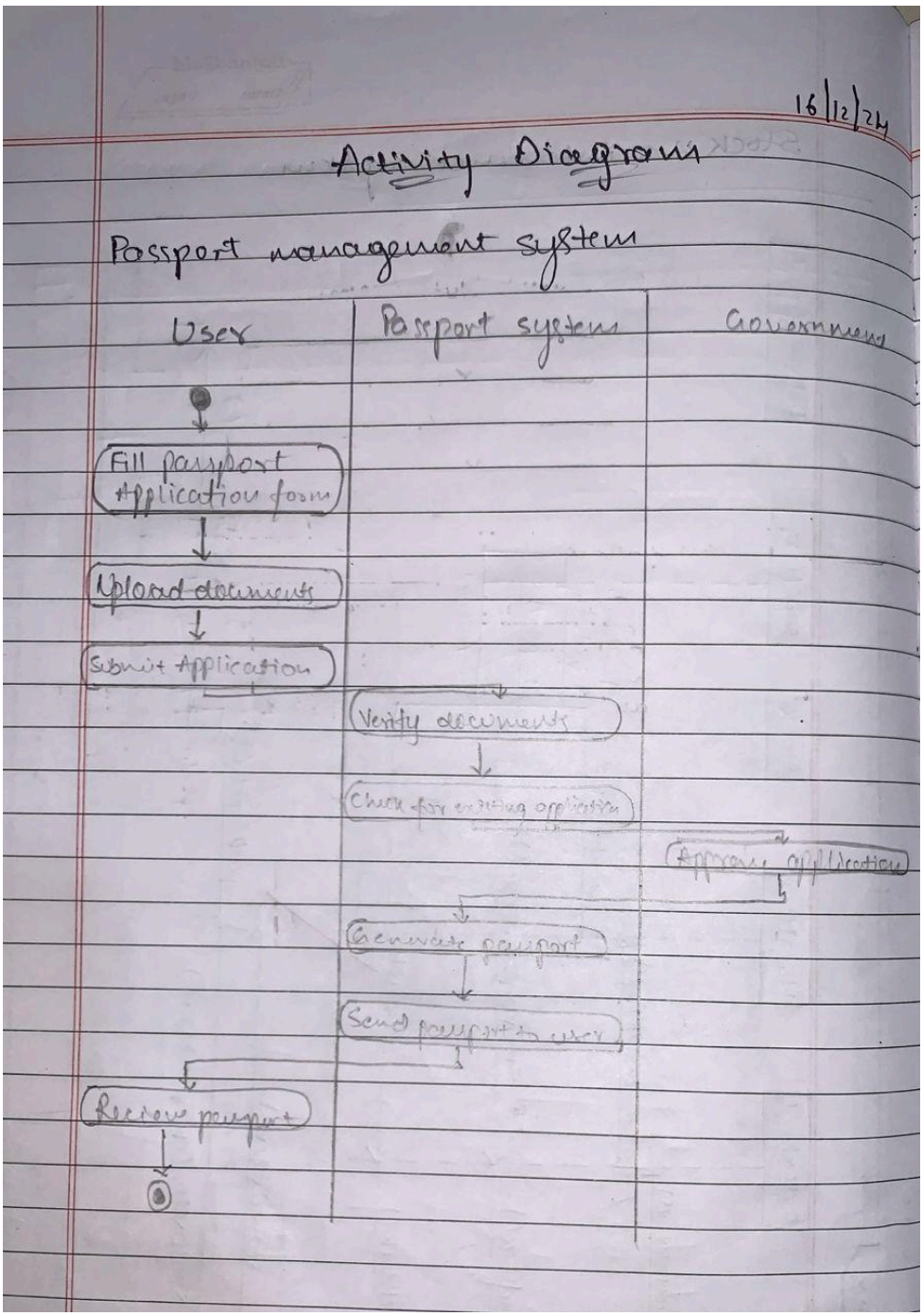

Fig.5.13: Activity Diagram Observation