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

Lab Record

Object Oriented Modelling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submittedby:

Mallikarjun M Kuri(1BM23CS144)

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 Modelling **(23CS5PCOOM)** laboratory has been carried out by **Mallikarjun M Kuri (1BM23CS144)** during the 5th Semester Sep 24- Jan2025.

Signature of the Faculty Incharge:

Spoorthi D M
Asst. Professor
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

1. Hotel Management System

Problem Statement

Managing hotel operations efficiently is a complex task due to the diverse range of activities involved, such as room reservations, guest check-ins and check-outs, staff coordination, inventory management, and billing. Many hotels still rely on manual processes or outdated systems, which can lead to inefficiencies, human errors, and poor customer experiences.

The lack of a centralized, user-friendly system makes it challenging to manage guest bookings, track room availability, handle billing, and maintain seamless communication between departments. Additionally, there is often no integration with modern features like online bookings, digital payments, or data analytics, which are essential for staying competitive in the hospitality industry.

SRS – Software Requirements Specification

Figure 1.1: SRS Document Observation Bk (1)

SRS Document for
1) Hotel Management System
1. Introduction
1.1 purpose of this Document
The purpose of this document is to outline the requirements for the Hotel management system (HMS). It serves as a guide for stakeholders, developers, and project managers to ensure a clear understanding of the system's objectives, functionalities and constraints.
1.2 scope of this Document
This document covers the overall objectives of the HMS, including guest management, room reservations, billing and reporting features. It will provide value to customers by streamlining operations, enhancing guest experiences, and improving management efficiency. The estimated development cost and timeline are also included.
1.3 Overview
The Hotel management system is designed to automate and manage the day-to-day operations of a hotel. It will facilitate online booking, check-in/check-out processes, and maintain guest records, enhancing operational efficiency and customer satisfaction.

2. General Description

The HRMS will serve hotel staff and guests by offering features such as online reservations, room management, invoicing, and reporting. users include hotel managers, front desk staff, and guests. The system will provide easy navigation, quick access to information, and integration with existing software

3) Functional Requirements

User registration: users can create and manage their accounts

~~Catalog management: Librarians can~~

Room management: staff can add, update and delete room information including status

Booking systems: to book rooms

Billing and invoicing: Generate and manage invoices for guests, including additional services

4) Interface Requirements

User Interface: web-based and mobile interfaces for guests and hotel staff

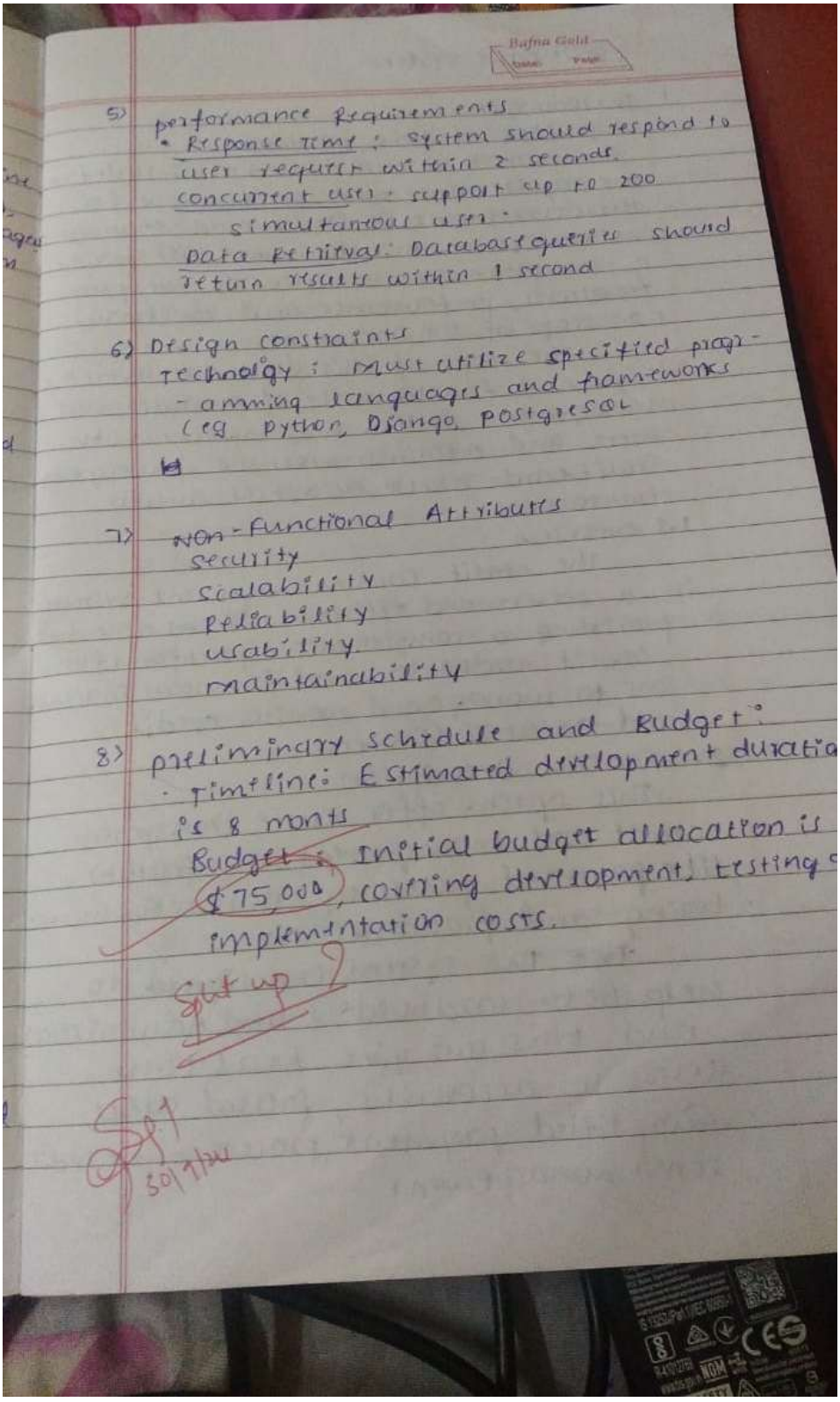
Database Interface: interaction with a backend database for room and guest data management

API Interfaces: Integration with third-party services, such as payment gateways and travel agencies

5)

6)

7)


Bafna Gold
5) performance Requirements
• Response Time: System should respond to user requests within 2 seconds.
concurrent users: support up to 200 simultaneous users.
Data Retrival: Database queries should return results within 1 second
6) Design constraints
Technology: Must utilize specified programming languages and frameworks (eg Python, Django, postgresql
7) Non-Functional Attributes
security
scalability
Reliability
usability
maintainability
8) preliminary schedule and Budget:
· Timeline: Estimated development duration is 8 months
Budget: Initial budget allocation is $75,000, covering development, testing & implementation costs.
Set up
Set 1
30/7/24

Class Diagram

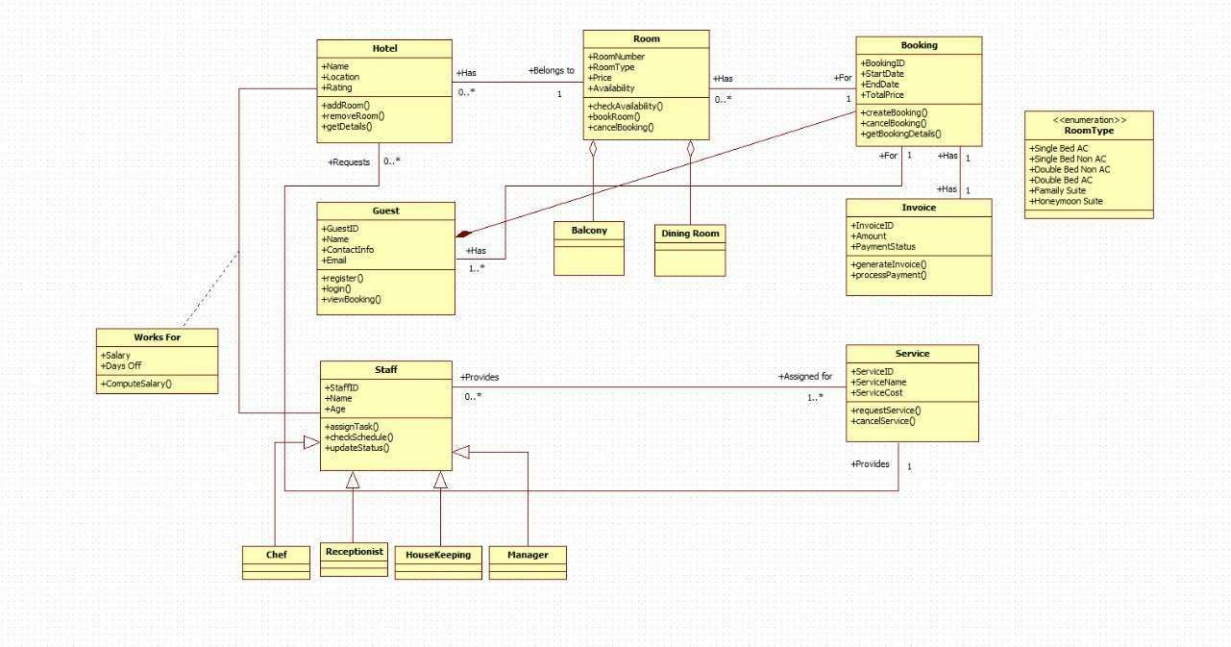


Figure 4.4: Class Diagram

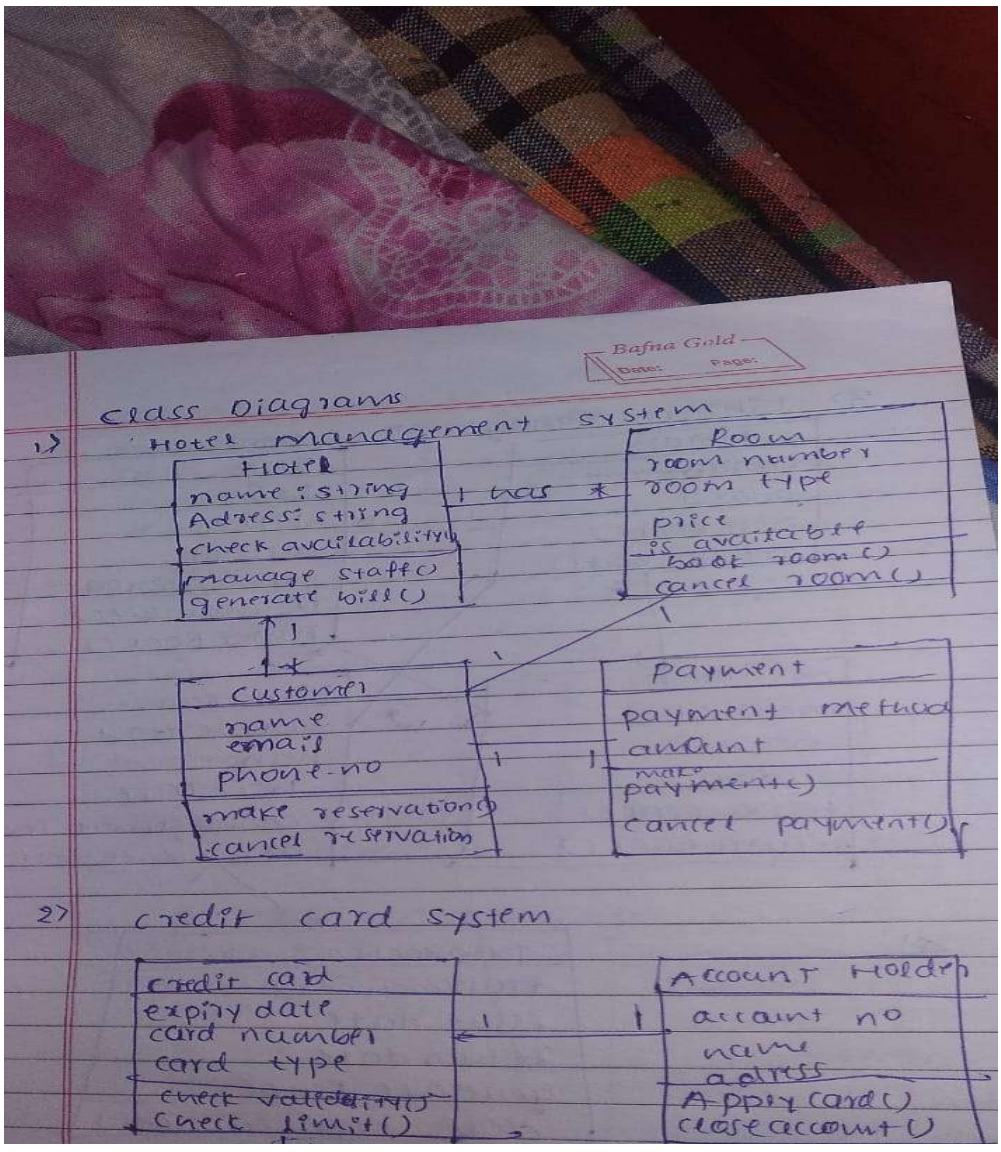
1. Room: Attributes include roomID, roomType, price, status, and amenities. Methods include check(), clean(), and assign(). The room is linked to bookings and reservations.
2. Customer: Attributes include customerID, name, contact, checkInDate, checkOutDate, and roomPreference. Methods include book(), pay(), and requestService(). Customers are linked to bookings and payments.
3. Booking: Attributes include bookingID, customer details, room, checkInDate, and checkOutDate. Methods include creatingBooking() and cancelBooking(). Connected to customers and rooms.
4. Reservation: Attributes include reservation date, time, and customer details. Methods include selectRoom(). Linked to rooms and customers.
5. Payment: Attributes include paymentID, amount, method, and status. Methods include process() and refund(). Payments are associated with customers.
6. Employee: Attributes include employeeID, name, role, salary, and shiftTiming. General methods include performDuty() and report(). Subtypes include:
    - Clerk: handleCheckTime(), manageRecords().

- Receptionist: greetCustomer(), assignRoom().
- Chef: provideFood(), manageKitchen().
- Bellboy: carryLuggage().

Relationships depict:

- Customers making bookings, reservations, and payments.
- Employees providing various services.
- Rooms being occupied and maintained.

Figure 5.5: Class Diagram Obs bk



State Diagram

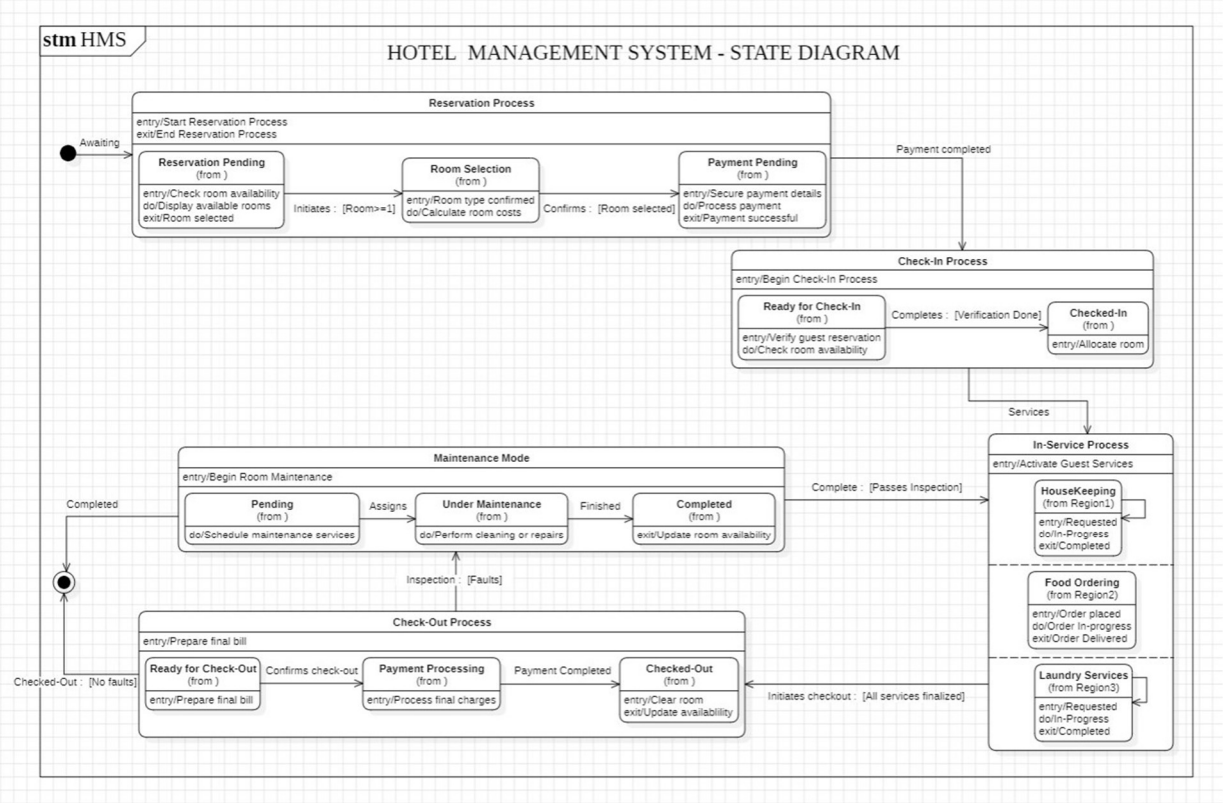


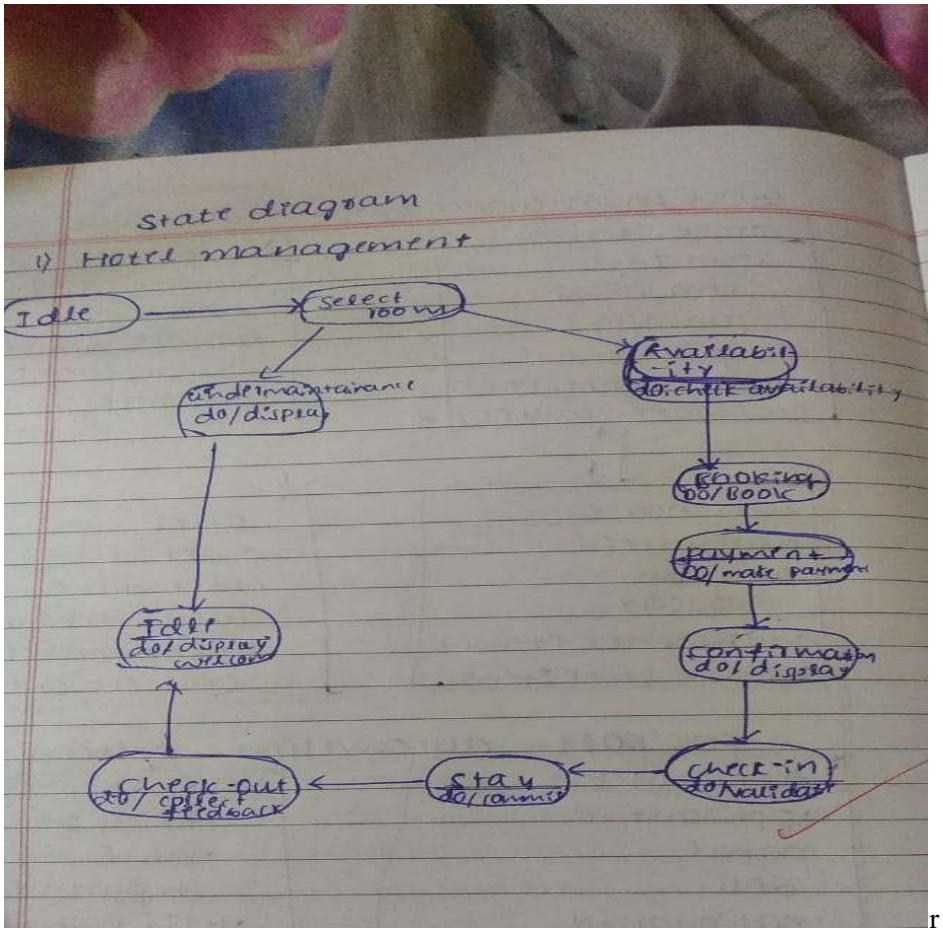
Figure 6.6: State Diagram

1. Booking:
    - States: Idle, Choice Selection, Check Availability, Confirm Booking, Confirmed, Failed.
    - Transitions: Booking starts from Idle, progresses through choice selection and availability checks, and ends in either Confirmed or Failed.
2. Room Occupied:
    - State: Represents the guest occupying the room after a successful booking.
    - Trigger: Booking is confirmed, and the guest checks in.
3. Ordering Room Service:
    - States: Choose Service, In Process, Completed.
    - Transitions: Customers choose a service, it gets processed, and then completed, with the cost added to the bill.
4. Billing:
    - States: Billing Initiated, Total Calculated, Added to Bill.
    - Transitions: Begins when a customer exits or orders additional services, calculates the total, and updates the bill.
5. Bill Calculation:

- States: Calculating, Adding Service and Tax, Return Tax.
- Transitions: Starts with calculating charges, adds applicable taxes or services, and adjusts for refunds if needed.

Relationships depict:

- Bookings transitioning through various states until confirmation or failure.
- Room occupancy dependent on booking confirmation.
- Room service ordered by guests affects the billing process.
- Billing integrates calculations



r room charges, services, and taxes.

Figure 7.7: State Diagram Obs bk

Use - Case Diagram

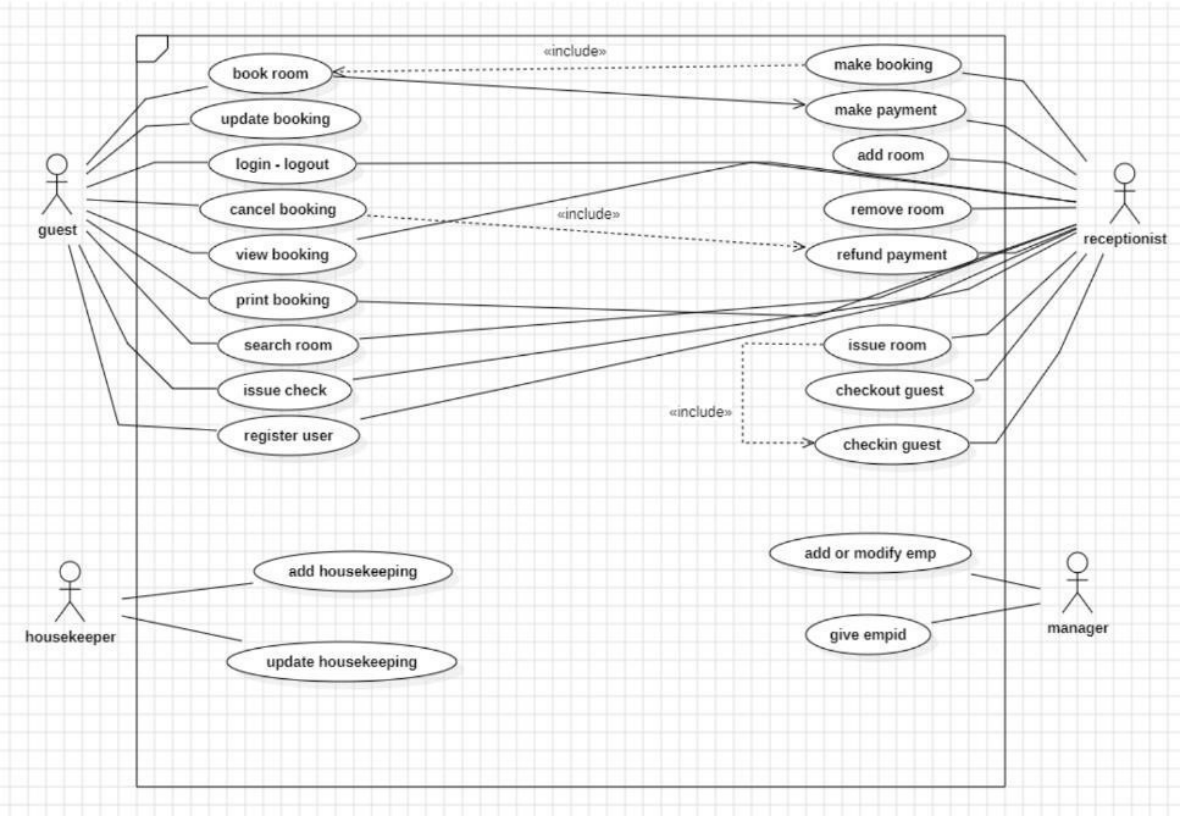
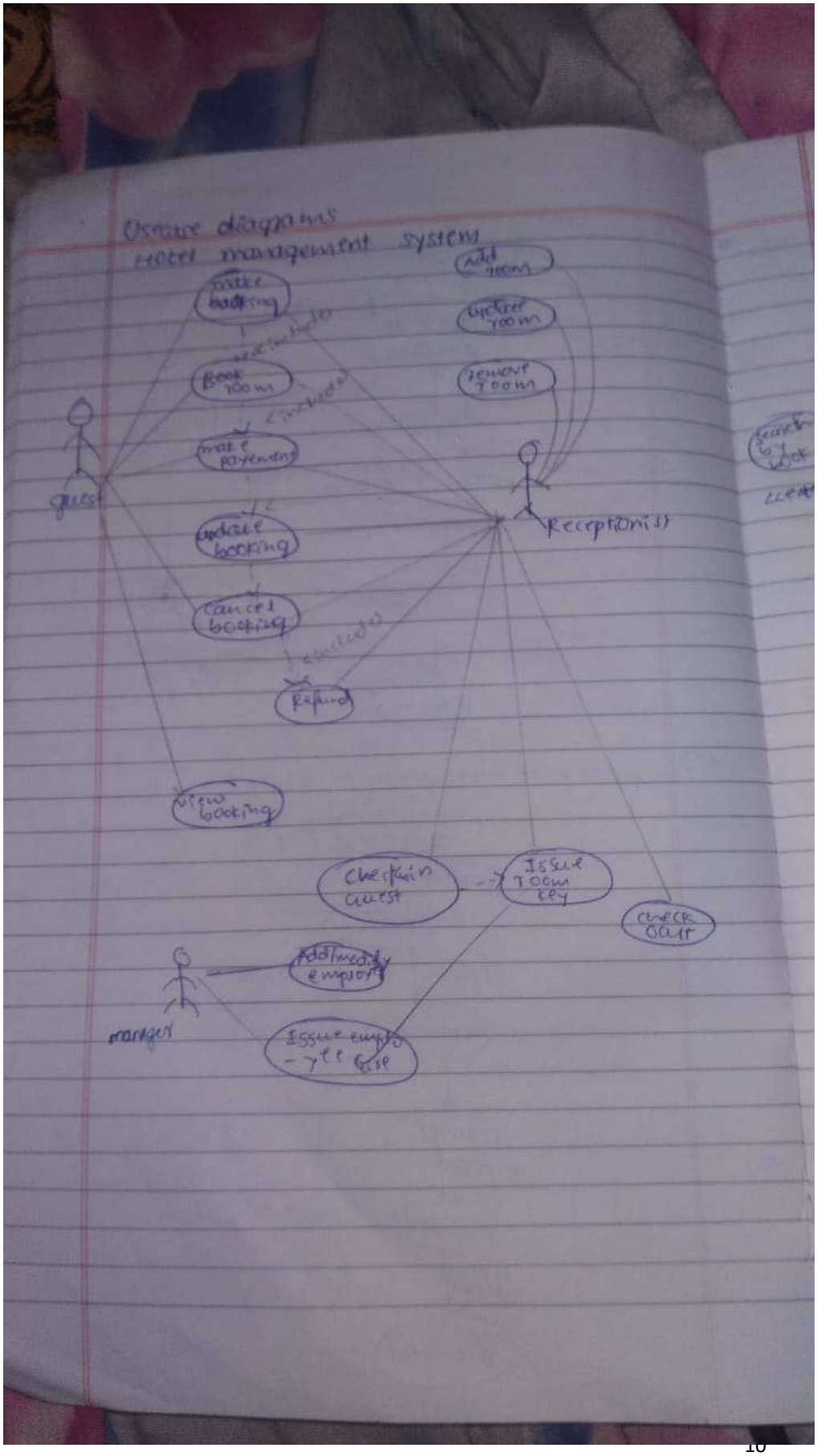


Figure 8.8: Use Case Diagram

1. Guest:
   - Actions:
     - Book Room, Update Booking, Login-Logout, Cancel Booking, View Booking, Print Booking, Search Room, Issue Check, Register User.
   - Relationships:
     - Includes: Make Booking (shared with the Receptionist).
     - The guest interacts with the system to manage their reservations and performs actions related to booking and viewing details.
2. Receptionist:
   - Actions:
     - Make Booking, Make Payment, Add Room, Remove Room, Refund Payment, Issue Room, Check-in Guest, Checkout Guest.
   - Relationships:
     - Includes: Shared tasks with the Guest (e.g., Make Booking) and Check-in/Check-out functionality.

- 
Usecase diagrams
Hotel management system
Add room
Update room
Remove room
Book room
make payement
guest
Receptionist
update booking
cancel booking
Refund
view booking
Check in guest
Issue room key
Check out
Add/modify employee
manager
Issue empl - yee pass

The receptionist facilitates customer interactions, processes payments, and manages room logistics.

3. Housekeeper:
   - Actions:
     - Add Housekeeping, Update Housekeeping.
   - Relationships:
     - Focused on maintaining room cleanliness and managing housekeeping schedules.
4. Manager:
   - Actions:
     - Add or Modify Employee, Assign Employee ID.
   - Relationships:
     - Oversees employee management and is responsible for assigning roles and duties within the hotel.

Figure 9.9: Use Case Diagram Obs bk

Sequence Diagram

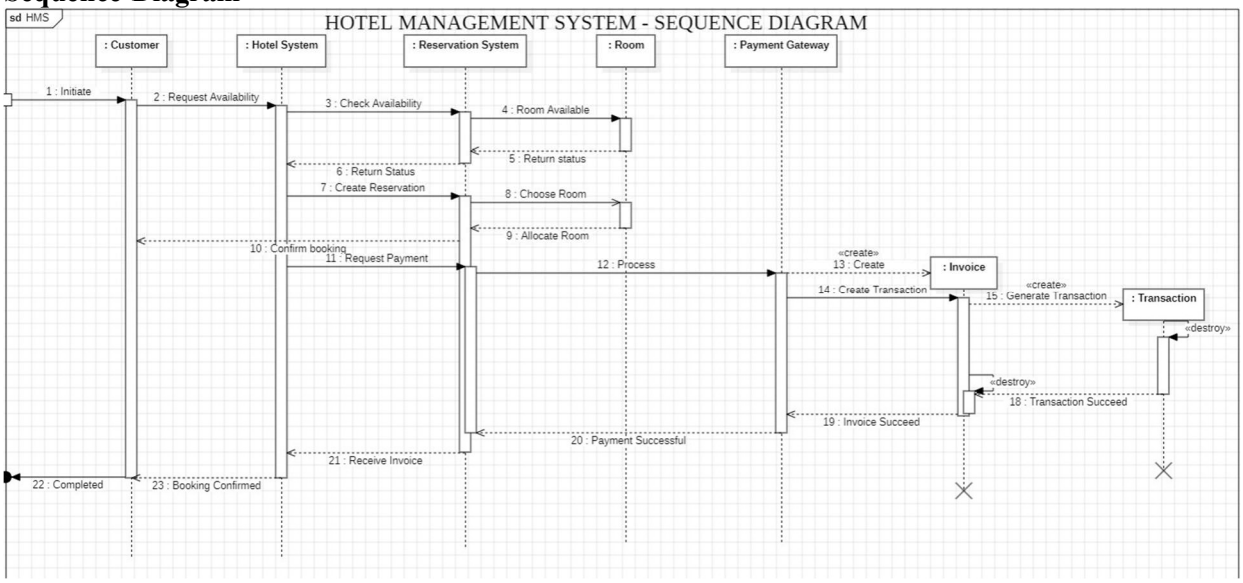


Figure 10.10: Sequence Diagram

Actors/Entities:

1. Customer: Initiates the process by interacting with the system to place an order.
2. Menu: Provides item details to the customer.
3. Order: Handles order validation and management.
4. Chef: Prepares the order after validation.
5. Server: Delivers the prepared order to the customer and manages notifications.

Interactions and Processes:

1. Customer views items (view_items):
    - The customer interacts with the menu to view available items.
    - Menu responds with the item_list.
2. Customer places an order (placeOrder):

- The customer sends order details to the Order system.

3. Order validation (validateOrder):
    - The Order system validates the order details.
    - If invalid, the invalidOrder method is triggered, and the customer is notified of a failure (orderFailedNotify).
    - If valid, the system proceeds to preparation.
4. Order preparation (prepareOrder):
    - The validated order is sent to the Chef.
    - The preparation takes 15 to 20 minutes (indicated with a timing note).
5. Order readiness (orderReady):
    - The Chef notifies the Order system when the preparation is complete.
6. Notification to Server (notifyReady):
    - The Order system notifies the Server about the order's readiness.
7. Order delivery (deliverOrder):
    - The Server delivers the order to the customer.
8. Customer feedback (orderFeedback):
    - The customer provides feedback on the order.

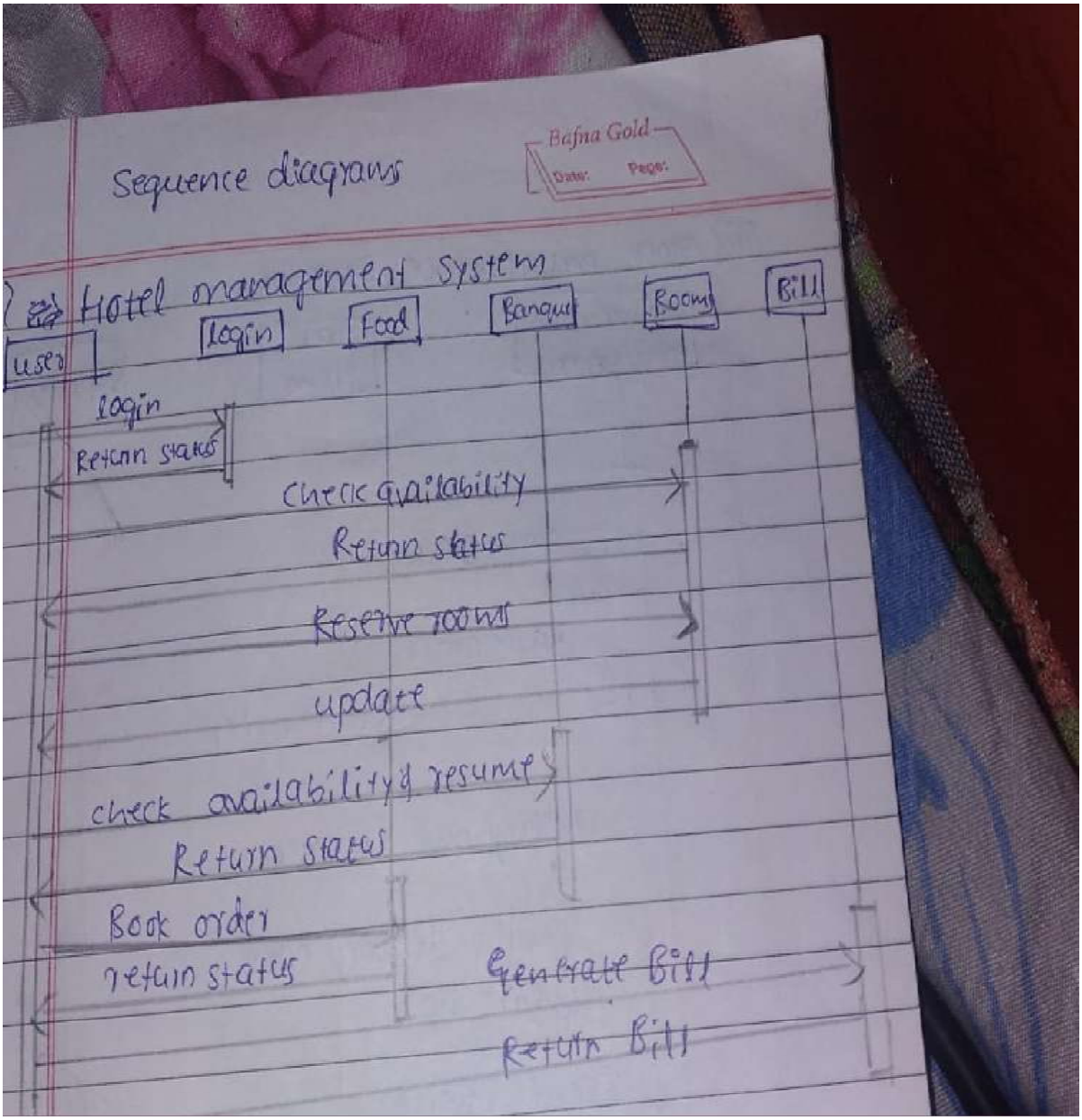


Figure 11.11: Sequence Diagram Obs bk

Activity Diagram

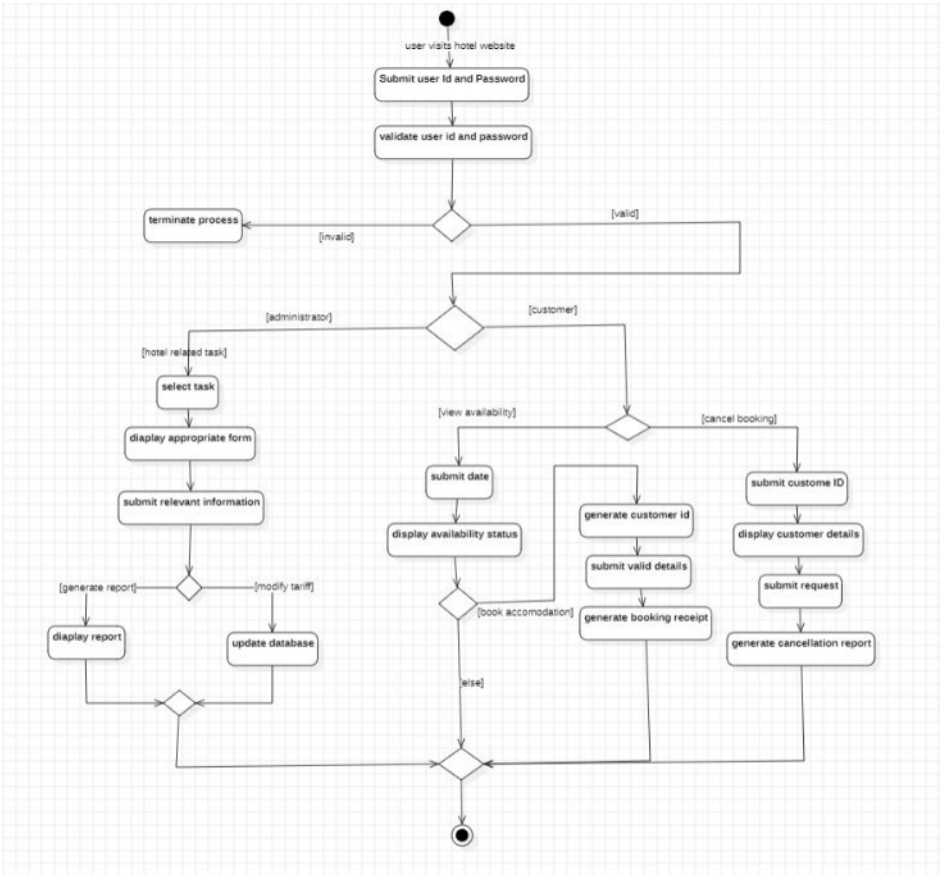


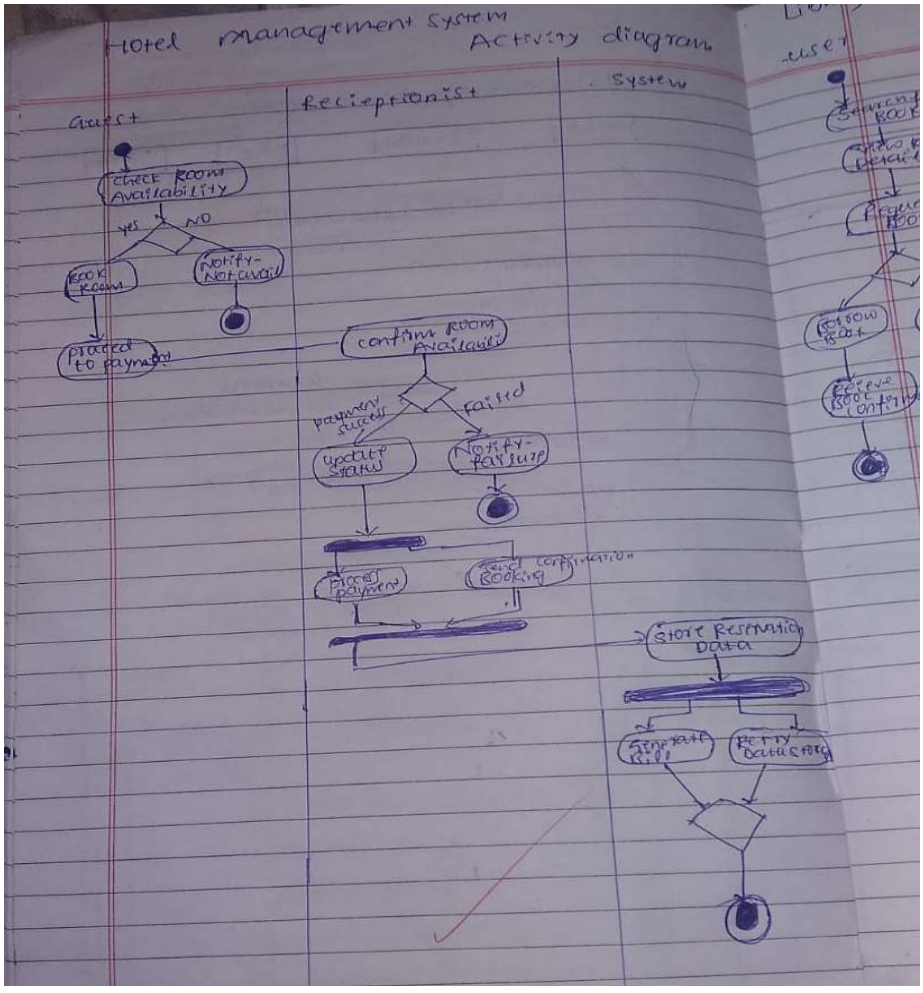
Figure 12.12: Activity Diagram

1. View Items:
    - The Customer interacts with the Menu to view available items (view_items()).
    - The Menu responds with the list of items (item_list()).
2. Place Order:
    - The Customer places an order by providing details (placeOrder(order details)).
3. Validate Order:
    - The Order object validates the received order (validateOrder()).
    - If the order is invalid, an error is triggered (invalidOrder()), and the customer is notified (orderFailedNotify()).
    - If the order is valid, the system proceeds to the next step.
4. Prepare Order:

- The Order sends the request to the Chef to prepare the order (prepareOrder()).
- The preparation process takes between 15–20 minutes.

5. Order Ready:
   - Once the preparation is complete, the Chef marks the order as ready (orderReady()).
   - The Order notifies the Server (notifyReady()).
6. Deliver Order:
   - The Server delivers the order to the Customer (deliverOrder()).
7. Feedback:
   - After receiving the order, the Customer provides feedback (orderFeedback()).

Figure 13.13: Activity Diagram Obs bk



2. Credit Card Processing

Problem Statement

The growing reliance on credit cards for transactions demands a robust, secure, and efficient system for processing payments. Traditional credit card processing systems often face challenges such as slow transaction speeds, security vulnerabilities, limited support for multiple payment methods, and difficulty in handling disputes or errors. These issues lead to customer dissatisfaction, financial losses, and non-compliance with industry standards like PCI DSS (Payment Card Industry Data Security Standard).

Additionally, businesses need real-time insights into their transactions, seamless integration with existing accounting systems, and support for modern payment technologies like mobile wallets and contactless payments.

SRS – Software Requirements Specification

Figure 2.1: SRS Document Observation Bk (1)

Credit card system

1 Introduction

1.1 purpose of this document

The purpose of this document is to define the requirements and specifications for the development of credit card management system. This document will provide a comprehensive description of the systems features, performance and constraints

1.2 Scope of the Document

The scope of this document covers the design and functionality of a credit card management system that allows both users and Administrators to manage credit card-related activities such as transactions

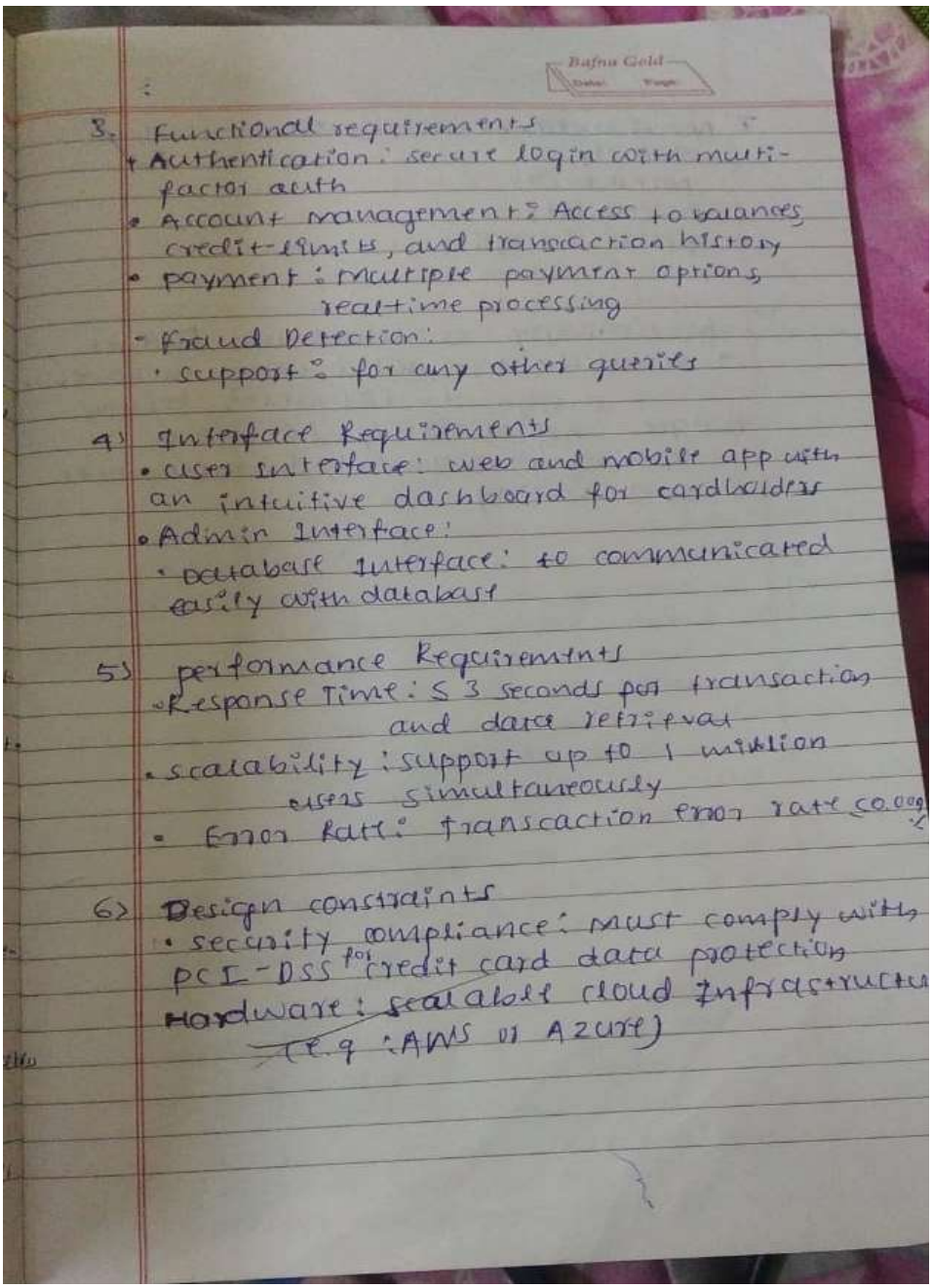
1.3 overview

The credit card management system is a secure and efficient system aimed at providing a comprehensive platform for credit cardholders and financial institutions to manage and monitor credit card-related services

2. General Description

This system offer secure, transparent credit card management with features like payment processing, transaction monitoring and fraud detection.

This system is designed to help both cardholders and Administrators and this can give Real-time access to accounts, fraud alerts, simplified payment process and credit limit management

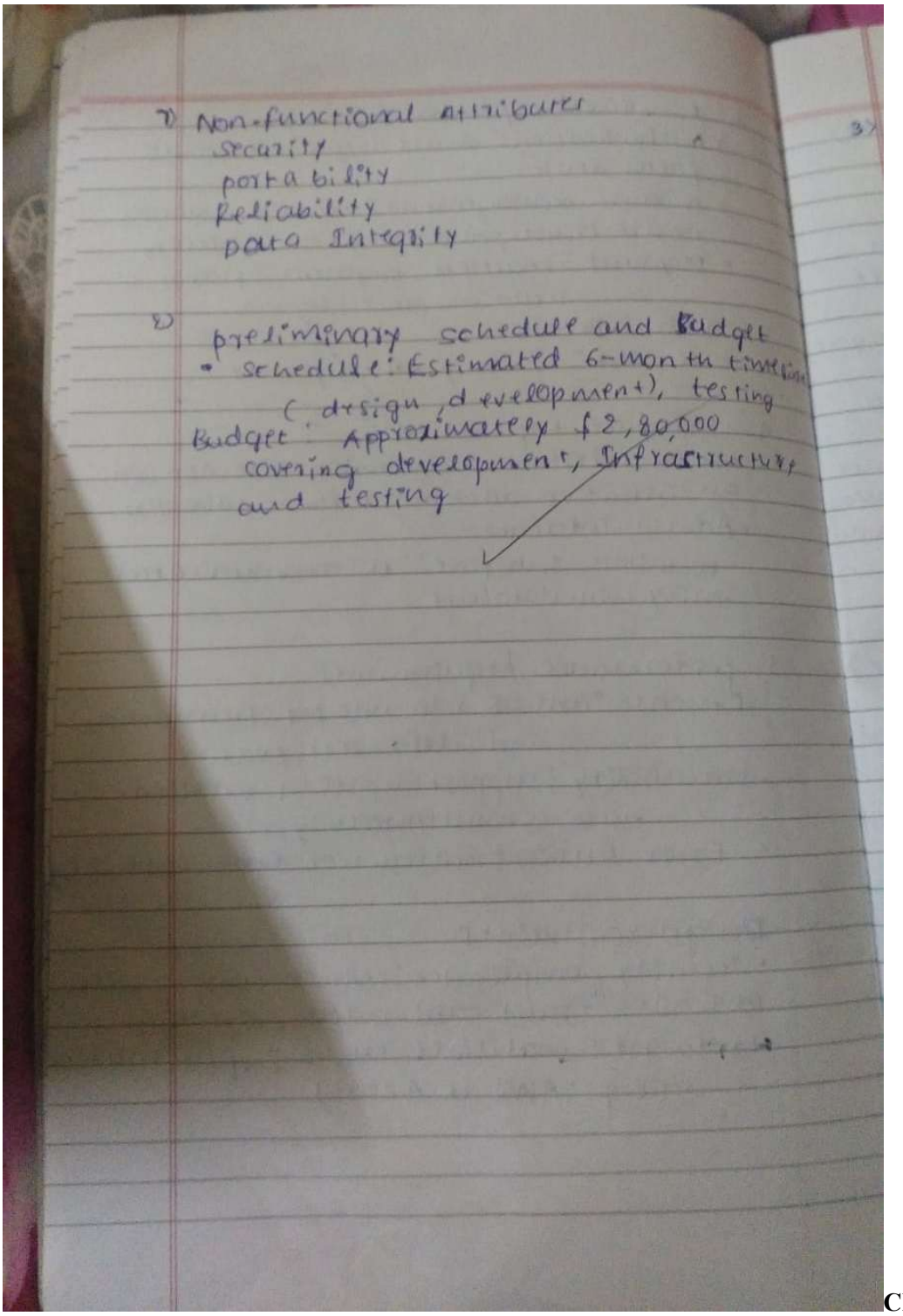


3. Functional requirements
- Authentication: secure login with multi-factor auth
- Account management: Access to balances, credit limits, and transaction history
- payment: multiple payment options, real-time processing
- Fraud Detection:
- support: for any other queries

4) Interface Requirements
- user interface: web and mobile app with an intuitive dashboard for cardholders
- Admin Interface:
- Database Interface: to communicated easily with database

5) performance Requirements
- Response Time: ≤ 3 seconds for transaction and data retrieval
- scalability: support up to 1 million users simultaneously
- Error Rate: transaction error rate ≤ 0.00%

6) Design constraints
- security compliance: must comply with PCI-DSS for credit card data protection
- Hardware: ~~scalable~~ cloud infrastructure (e.g: AWS or AZURE)



Class Diagram

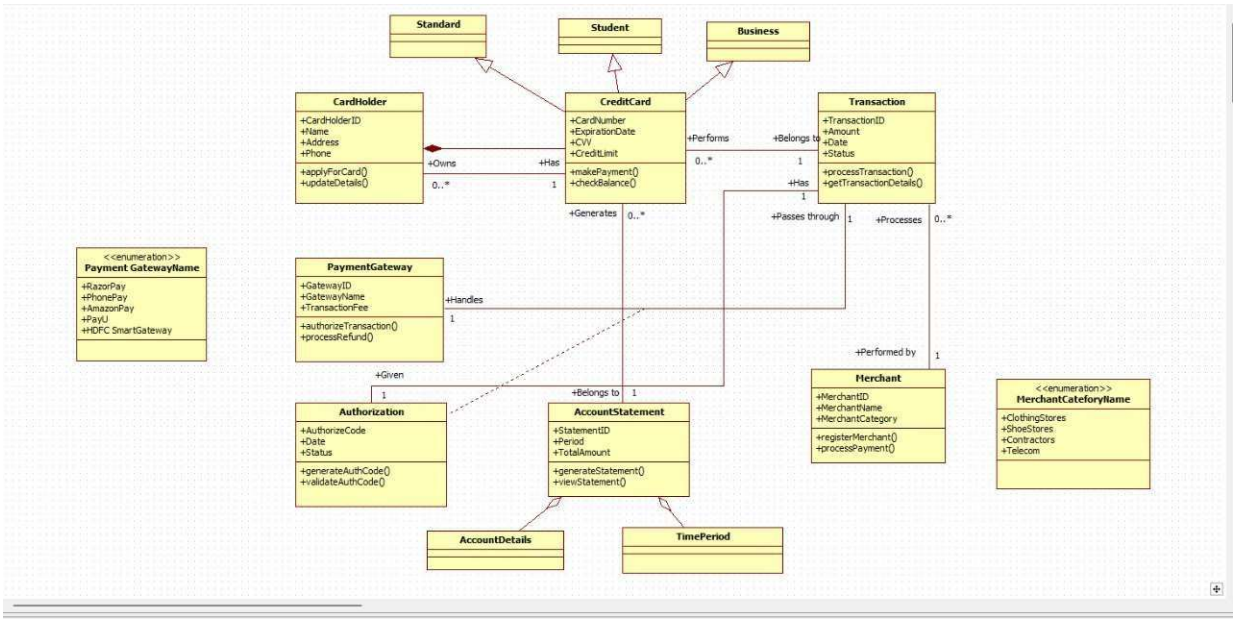


Figure 2.4: Class Diagram

Customer: Attributes include custID, name, contact, balance. Methods include pay(), viewTrans(), updateInfo(), requestCard(). Customers make bookings, reservations, and payments.

Statement: Attributes include statementID, issueDate, totalDue. Methods include generateStatement(). Customers have statements.

Transaction: Attributes include transID, amount, date, status. Methods include initiate(), confirmTrans(), cancelTrans(), refundTrans(). Customers generate transactions. Customers participate in transactions. Transactions include credit cards.

CreditCard: Attributes include cardNo, cvv, expireDate, amount. Methods include validateCard(), checkBalance(), updateBalance(), blockCard(). Credit cards belong to banks.

Bank: Attributes include bankID, name, branch, ifscCode. Methods include generateStatement(), authorizeTransaction(), openAccount().

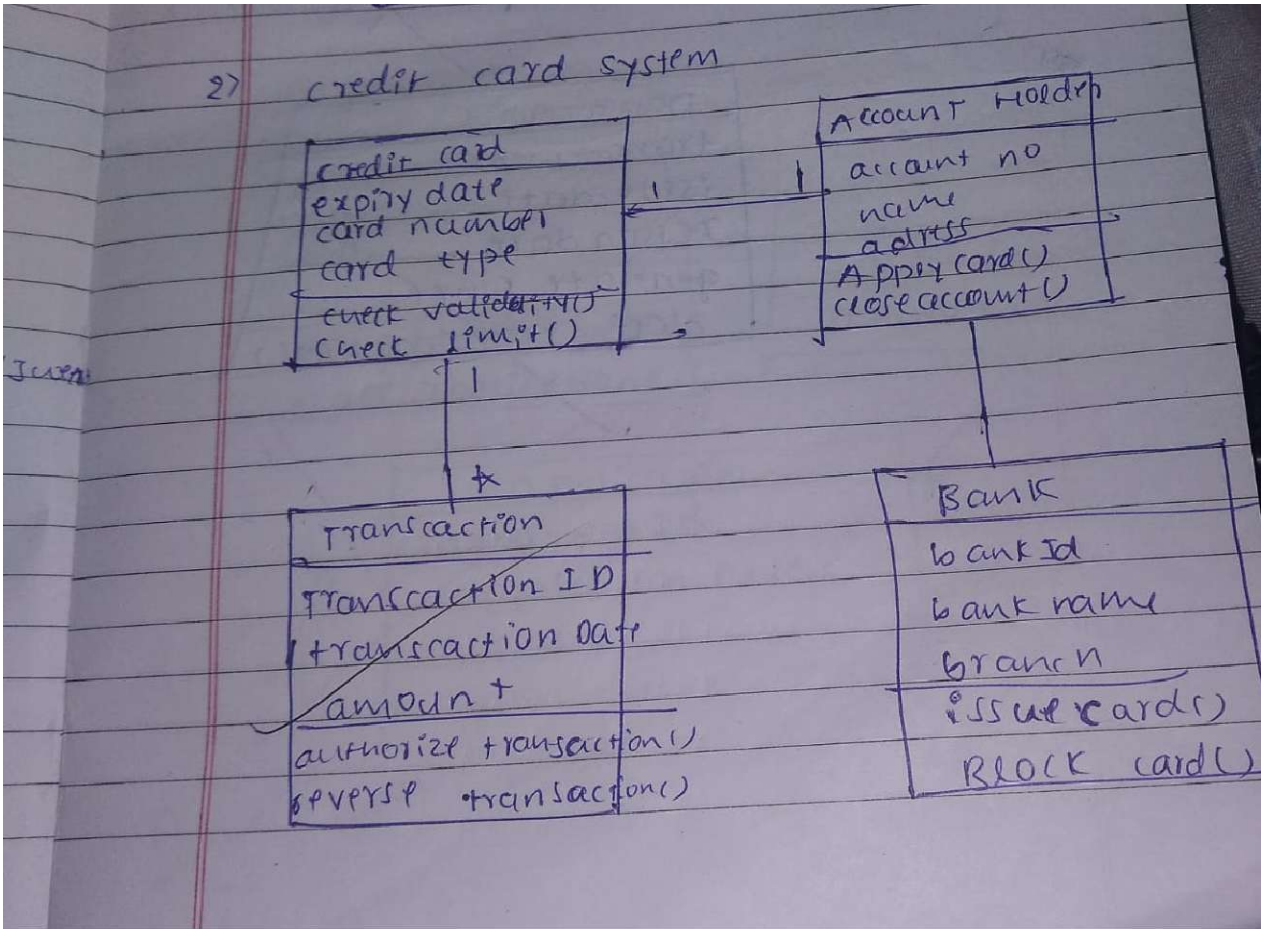
Platinum: (Subtype of CreditCard) Attributes include cardNo, limit. Methods include validate(), block(), update().

Diamond: (Subtype of CreditCard) Attributes include cardNo, limit. Methods include validate(), block(), update().

Relationships depict:

- Customers generate transactions.
- Customers participate in transactions.
- Customers have statements.
- Transactions include credit cards.
- Credit cards belong to banks.

Figure 2.5: Class Diagram Obs bk



State Diagram

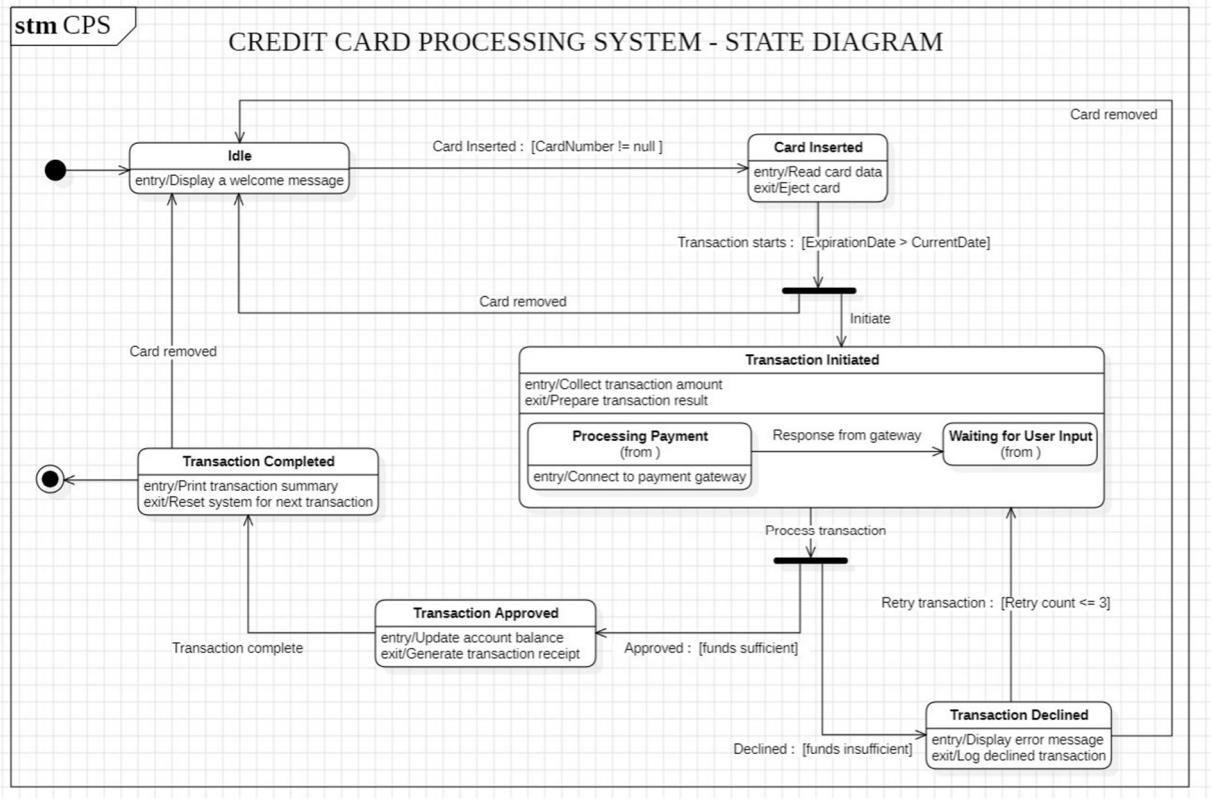


Figure 2.6: State Diagram

States:

- Initialized: The system starts here.
- Validate: Card and PIN are checked.
- Transaction: Amount is entered and processed.
- Failure: Errors occur during processing.

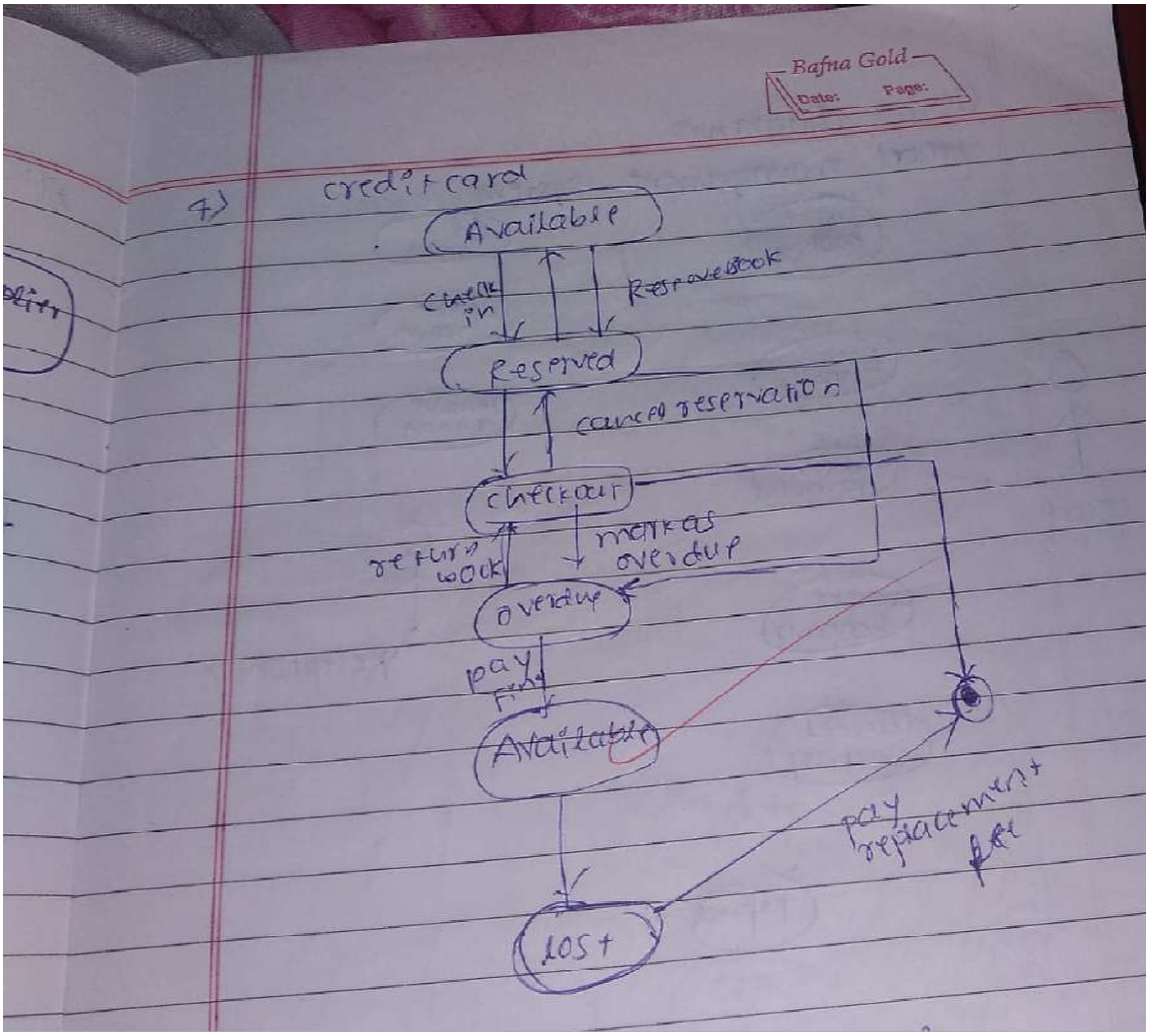
Transitions:

- The system moves from Initialized to Validate upon card insertion.
- If validation succeeds, it transitions to Transaction.
- If validation fails, it goes to Failure.
- Transaction can also lead to Failure if issues arise.
- The system can be reset from any state.

Actions:

- Card matching, PIN verification, amount input, and balance updates are performed during the process.

Figure 2.7: State Diagram Obs bk



Use - Case Diagram

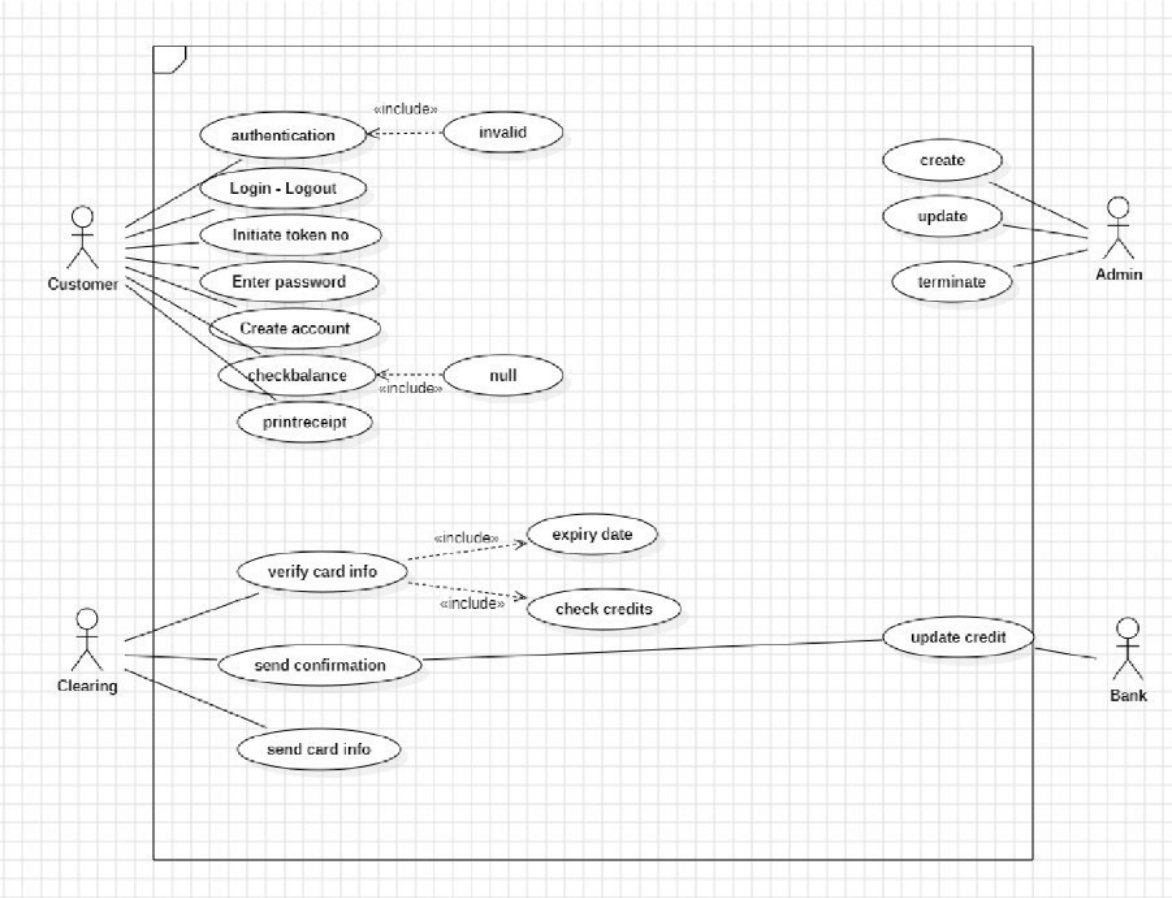


Figure 2.8: Use Case Diagram

Use Cases:

- Customers can login, create accounts, check balances, and initiate transactions.
- The system verifies card information and sends transaction details to clearing.
- Admins can manage accounts and system settings.
- The bank updates credit information.

Relationships:

- Authentication is included in login/logout.
- Other use cases may have optional extensions.

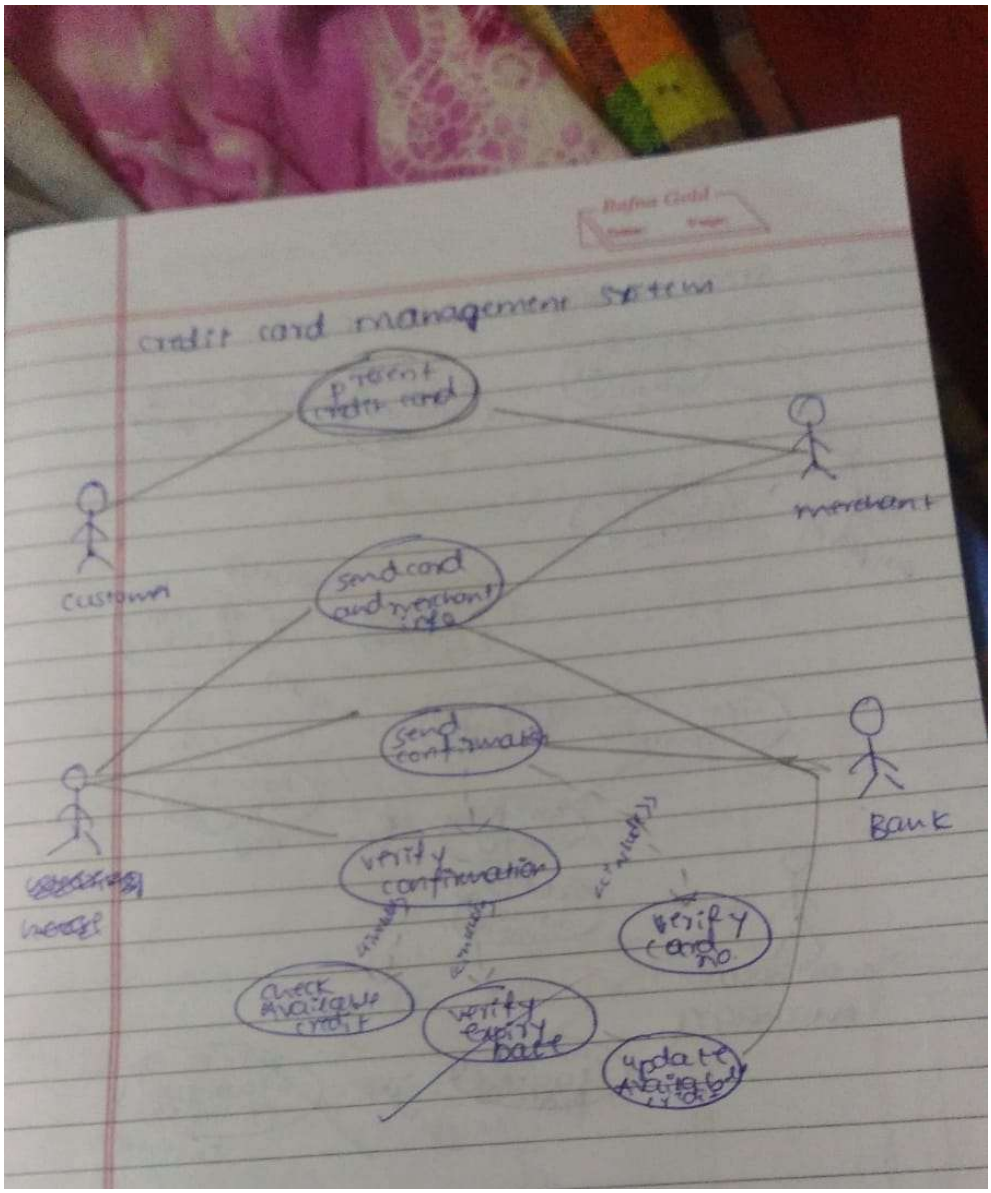


Figure 2.9: Use Case Diagram Obs bk

Sequence Diagram

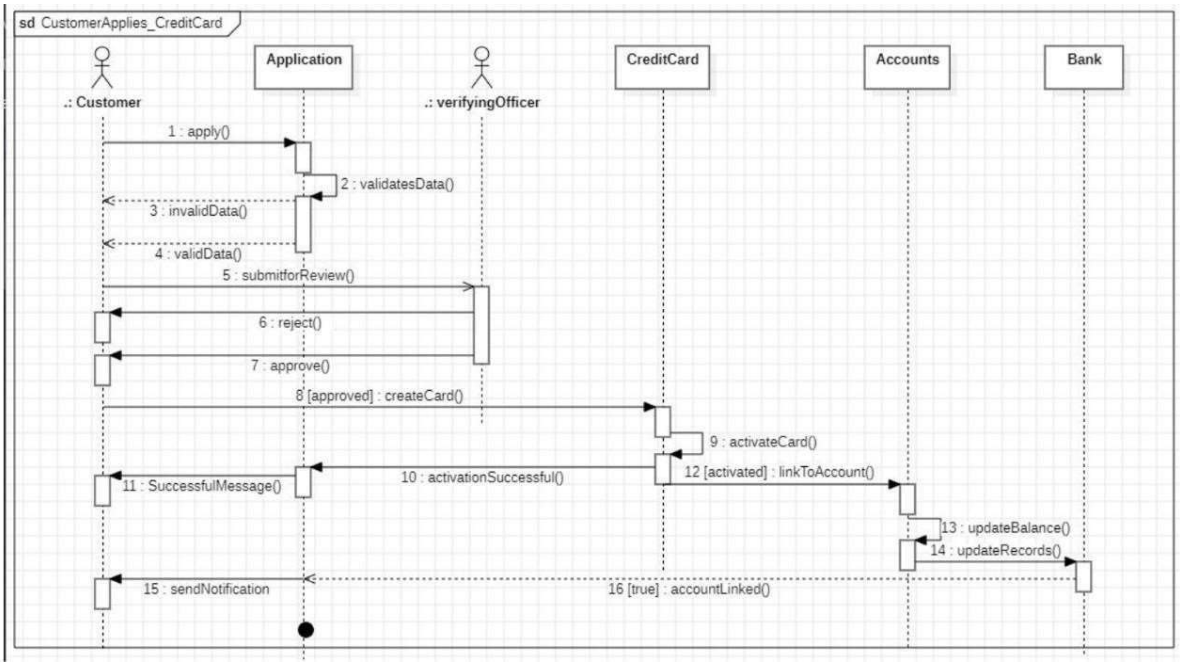
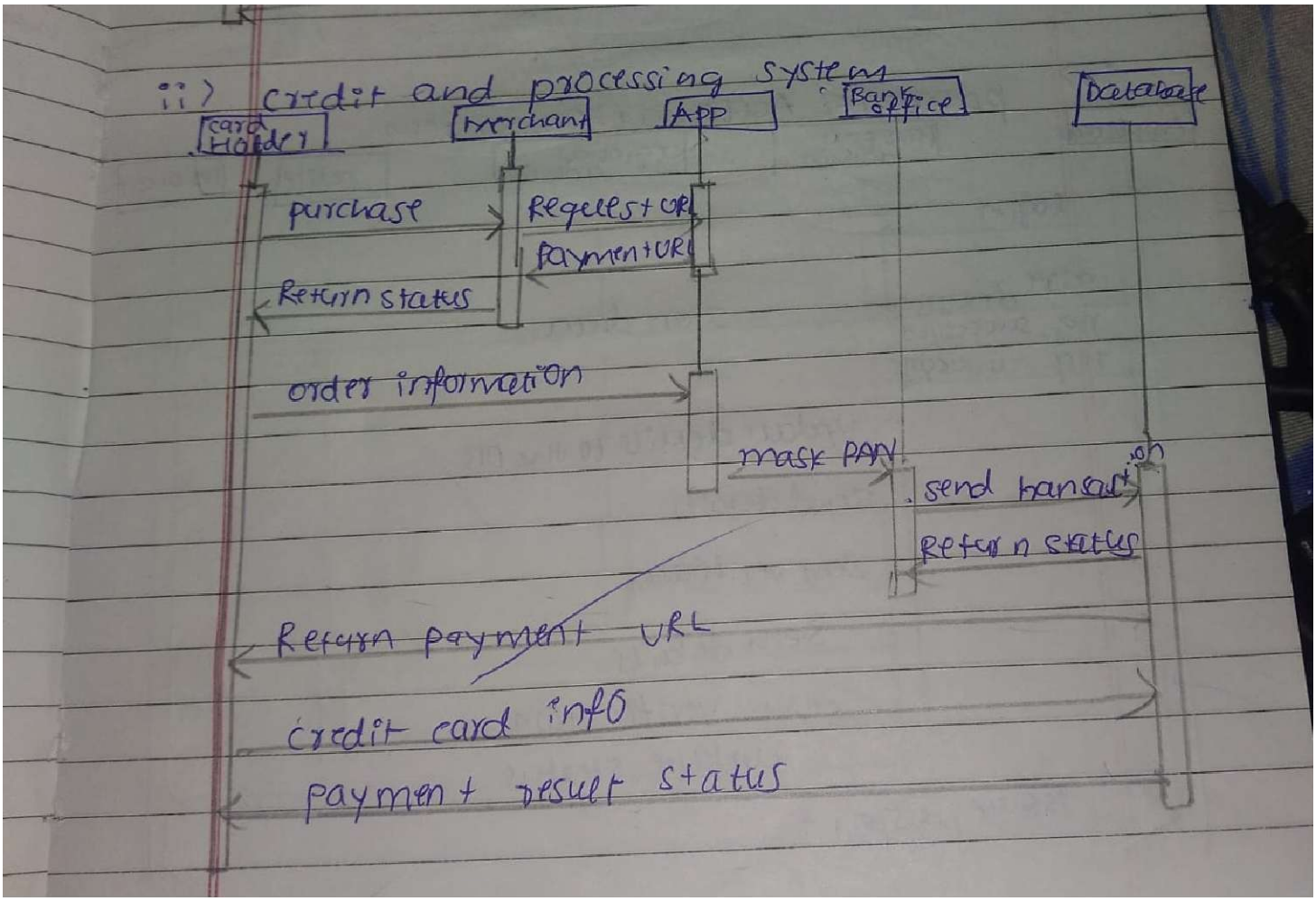


Figure 2.10: Sequence Diagram

1. Customer applies(): The customer initiates the process by applying for a credit card.
2. validateData(): The application object validates the data provided by the customer.
3. invalidData(): If the data is invalid, an error message is sent to the customer.
4. validData(): If the data is valid, the application is submitted for review.
5. submitForReview(): The application object submits the application to the verifying officer.
6. reject(): If the verifying officer rejects the application, the customer is notified.
7. approve(): If the verifying officer approves the application, a credit card is created.
8. createCard(): The credit card object is created.
9. activateCard(): The credit card is activated.
10. activationSuccessful(): The customer is notified of the successful activation.
11. SuccessfulMessage(): A success message is displayed to the customer.
12. linkToAccount(): The credit card is linked to the customer's account.
13. updateBalance(): The customer's account balance is updated.
14. updateRecords(): The bank's records are updated.
15. sendNotification(): A notification is sent to the customer.
16. accountLinked(): The system confirms that the credit card has been linked to the account.

Figure 2..11: Sequence Diagram Obs bk



Activity Diagram

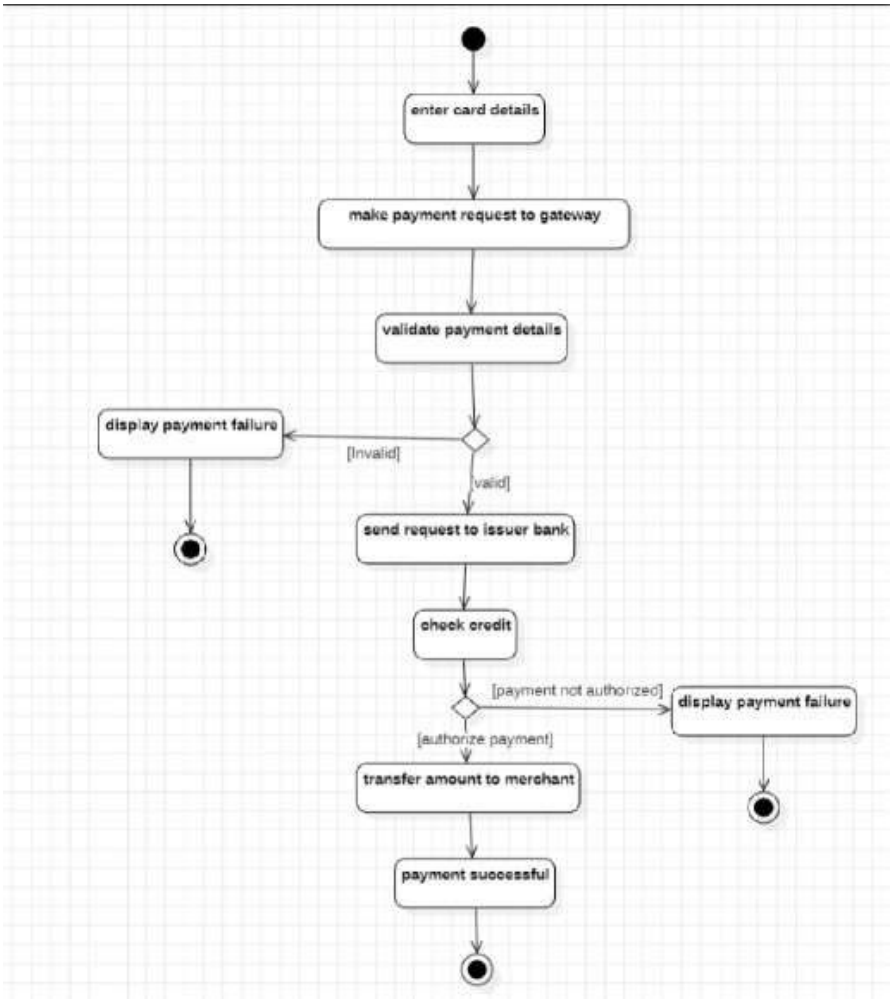


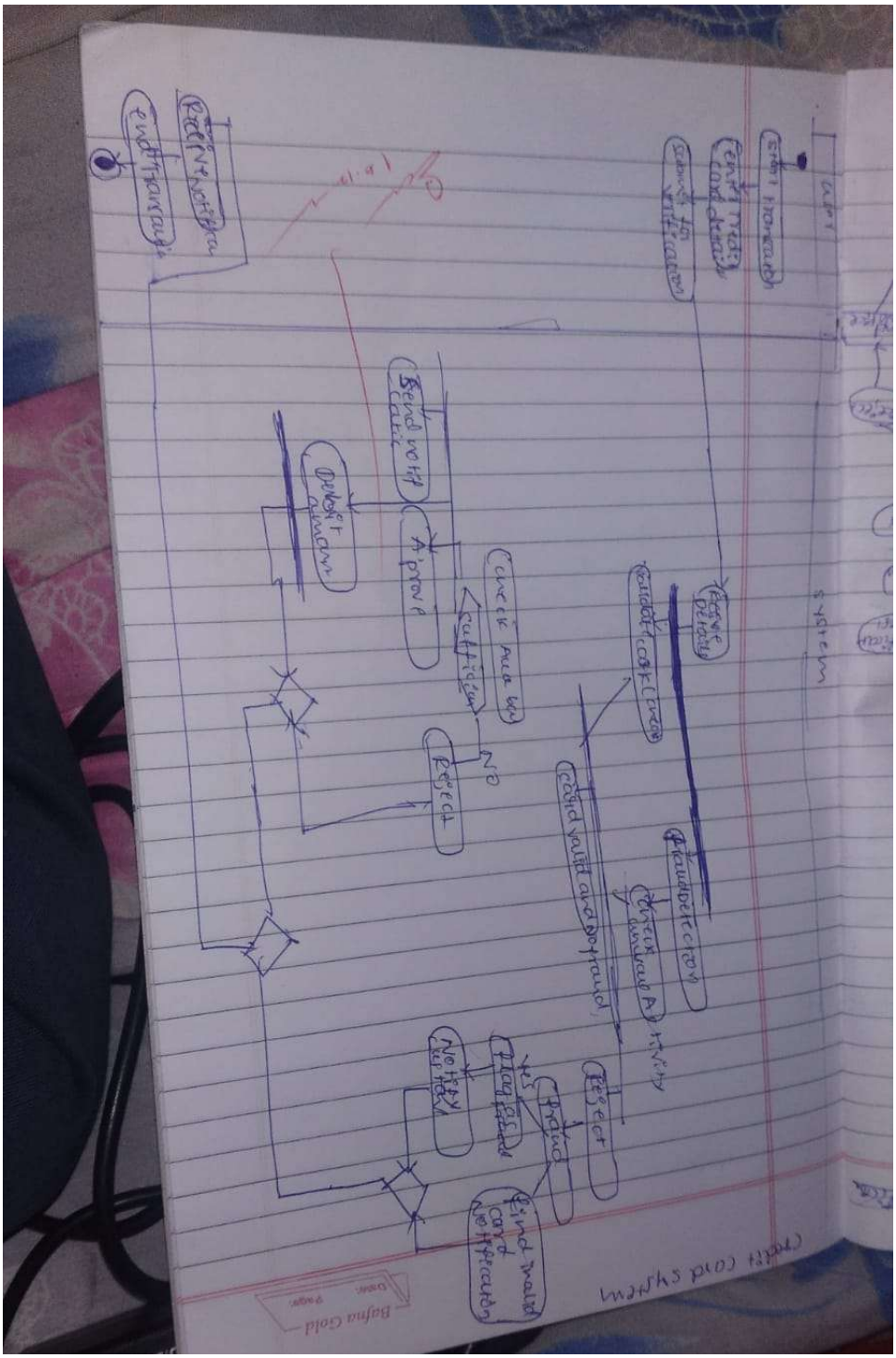
Figure 2.12: Activity Diagram

1. Enter Card Details: The user enters their card details, such as card number, expiration date, and CVV.
2. Make Payment Request to Gateway: The system sends a payment request to the payment gateway.
3. Validate Payment Details: The payment gateway validates the entered card details.
    - Invalid: If the details are invalid, the process ends with a "payment failure" message.
    - Valid: If the details are valid, the process proceeds to the next step.
4. Send Request to Issuer Bank: The payment gateway sends a request to the issuer bank to authorize the transaction.
5. Check Credit: The issuer bank checks the user's credit limit and account balance.

- Payment Not Authorized: If the payment is not authorized (e.g., insufficient funds), the process ends with a "payment failure" message.
- Authorize Payment: If the payment is authorized, the process proceeds to the next step.

6. Transfer Amount to Merchant: The issuer bank transfers the payment amount to the merchant's account.
7. Payment Successful: The payment process is completed successfully.

Figure 2.13: Activity Diagram Obs bk



3. Library Management System

Problem Statement

Traditional library management systems often rely on manual processes, leading to inefficiencies in tracking books, managing member records, and handling transactions such as book borrowing and returns. These methods result in time-consuming tasks, data inaccuracies, limited accessibility for users, and challenges in generating meaningful insights for library staff. A lack of integration with digital technologies further hampers the library's ability to meet modern user expectations for convenience and efficiency.

SRS – Software Requirements Specification

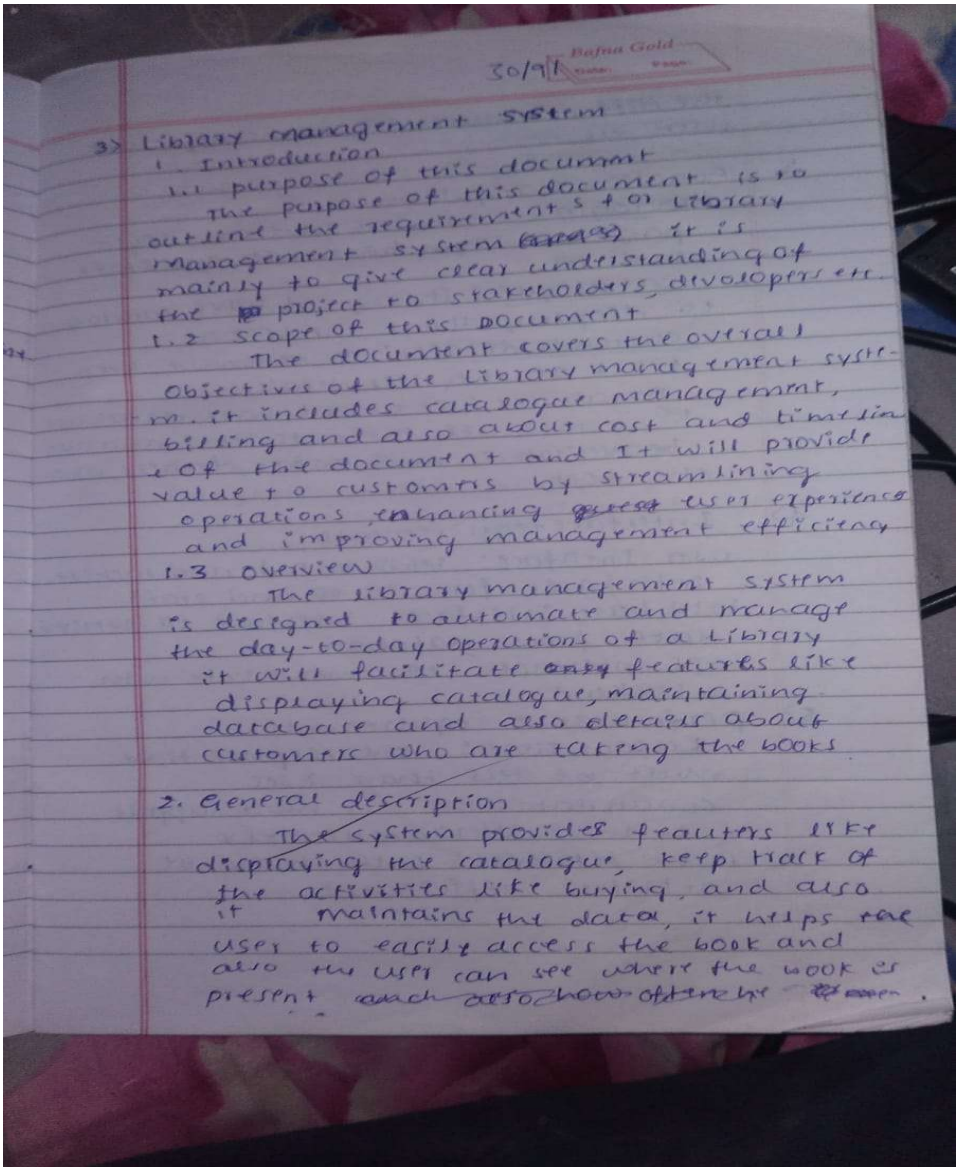
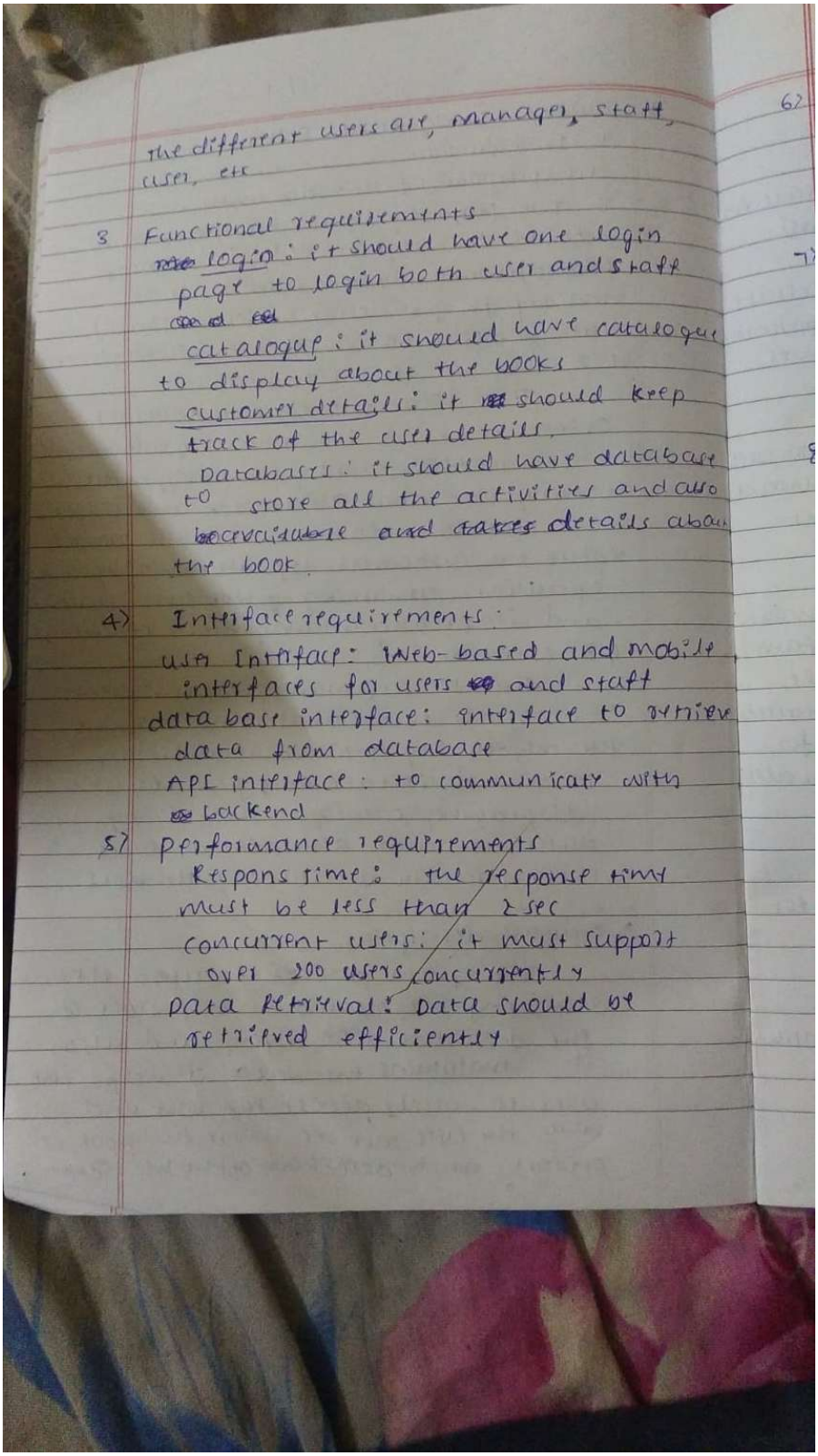


Figure 3.1: SRS Document Observation Bk (1)



the different users are, manager, staff, user, etc

3 Functional requirements
login: it should have one login page to login both user and staff
catalogue: it should have catalogue to display about the books
customer details: it should keep track of the user details.
Database: it should have database to store all the activities and also ~~b~~ ~~r~~ details about the book.

4) Interface requirements:
user Interface: Web-based and mobile interfaces for users and staff
data base interface: interface to retrieve data from database
API interface: to communicate with backend

5) performance requirements
Respons time: the response time must be less than 2 sec
concurrent users: it must support over 200 users concurrently
Data Retrival: Data should be retrieved efficiently

Bafna Gold
Date: Page:

6) Design constraints:
It must use programming languages and frameworks (e.g python, Django, postgreSQL)

7) Non-functional Attributes
security
scalability
Reliability
usability
maintainability

8) preliminary
Timeline: Estimated development duration is 8 months
Budget: Initial budget allocation is $75,000, covering development, testing and implementation costs

Class Diagram

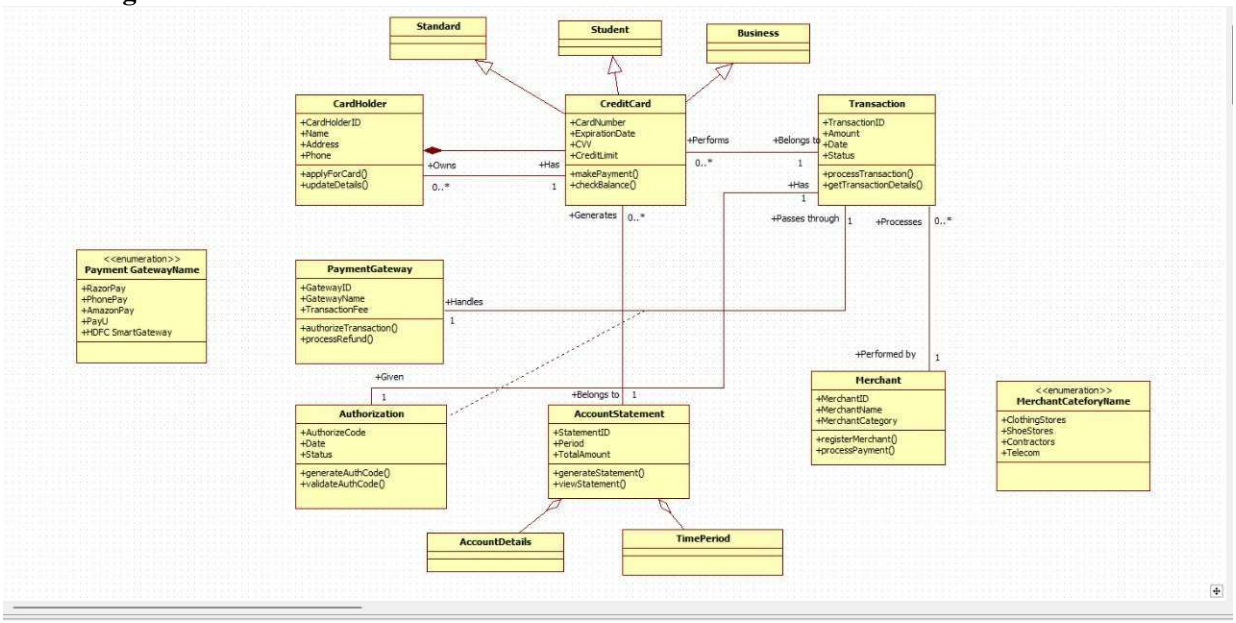


Figure 3.4: Class Diagram

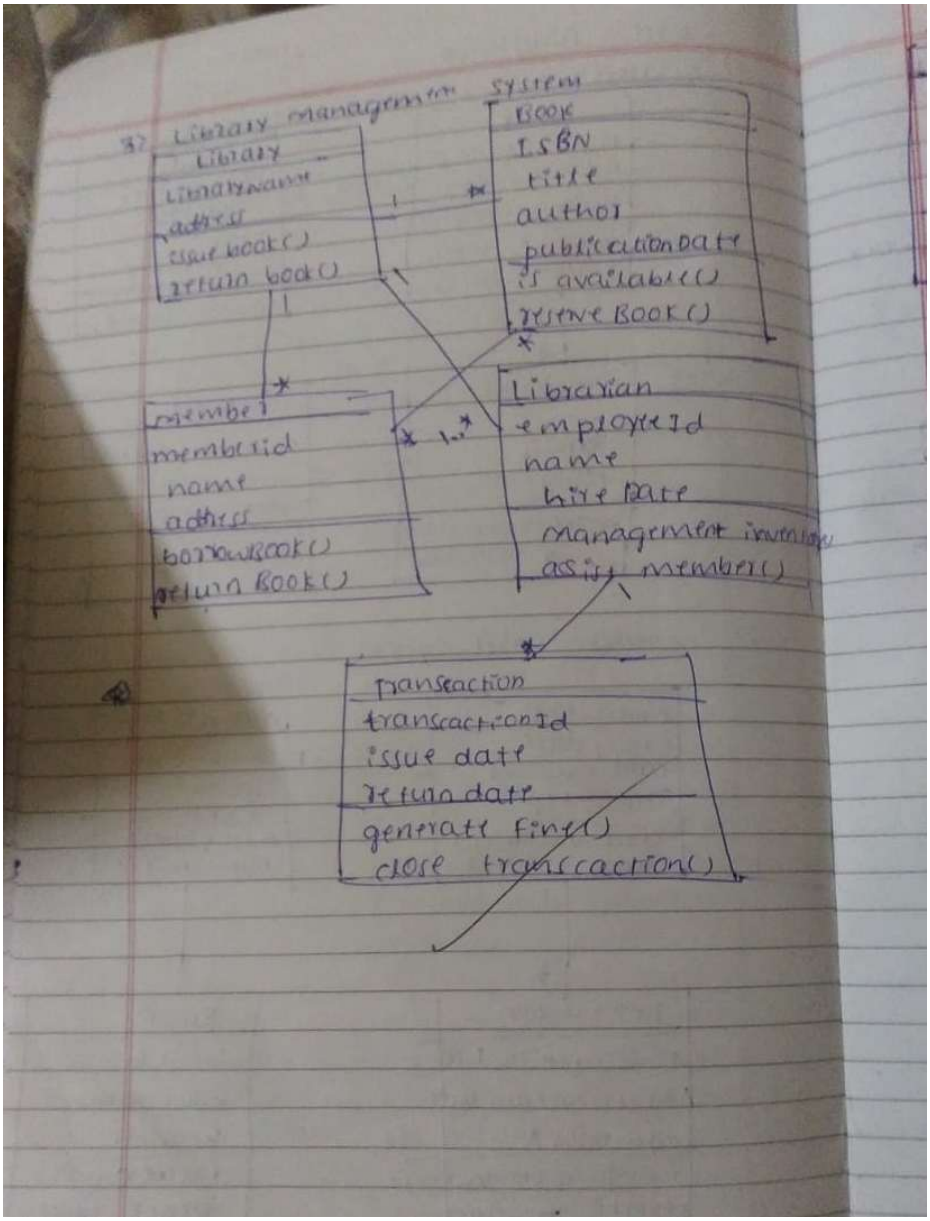
1. Library: Manages books.
2. Book: Has title, author, ISBN, publication year, availability. Can be issued and returned.
3. Magazine, Newspaper, Fiction: Subtypes of Book with specific attributes.
4. Librarian: Manages library, registers members, fines members.
5. Member: Borrows and returns books. Owns a library card. Can have fines.
6. FacultyMember, StudentMember: Subtypes of Member.
7. LibraryCard: Has ID, issue date, expiration date. Can be renewed.
8. Transaction: Tracks borrowing and returning dates, fines.
9. Fine: Has amount and due date. Can be paid.
10. Author: Writes books.

Relationships:

- A Library has many Books.
- A Book is written by one Author.
- A Book can be borrowed by zero or one Member.

- A Member owns one LibraryCard.
- A Member can have zero or more Fines imposed on them.
- FacultyMember and StudentMember are subclasses of Member.
- Magazine, Newspaper, and Fiction are subclasses of Book.

Figure 3.5: Class Diagram Obs bk



State Diagram

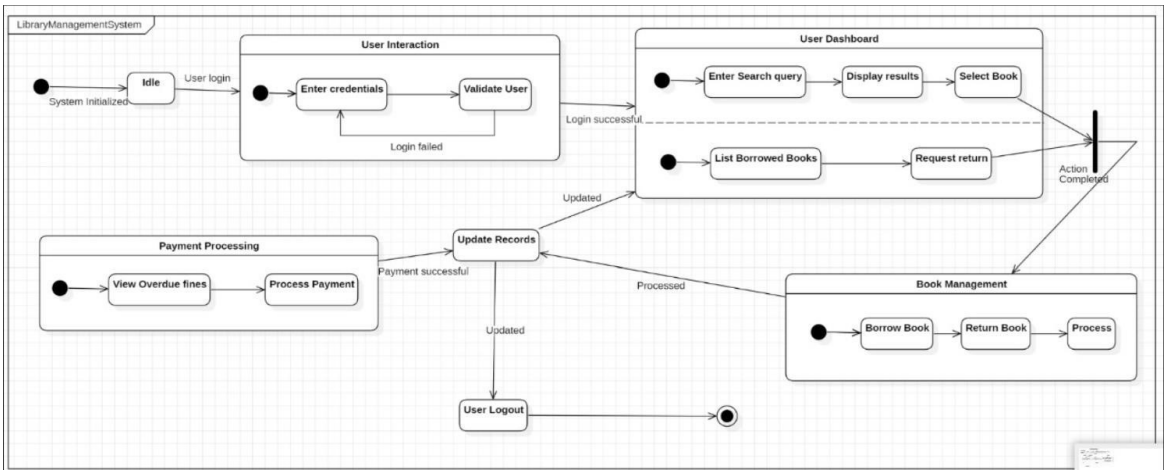


Figure 3.6: State Diagram

- User Interaction:
  - User Login: The user enters their credentials to log into the system.
  - Validate User: The system validates the user's credentials.
  - Enter Search Query: The user enters a search query to find books.
  - Display Results: The system displays the search results to the user.
  - Select Book: The user selects a book from the results.
  - List Borrowed Books: The system displays a list of books borrowed by the user.
  - Request Return: The user requests to return a borrowed book.
  - User Logout: The user logs out of the system.
- Book Management:
  - Borrow Book: The system processes the borrowing of a book by a user.
  - Return Book: The system processes the return of a book by a user.
- Payment Processing:
  - View Overdue Fines: The system displays the overdue fines for the user.
  - Process Payment: The system processes the payment of overdue fines.

Relationships:

- The diagram shows a sequential flow of activities, with some activities branching based on conditions (e.g., successful login, valid payment).
- The "Action Completed" state represents the end of a successful transaction.

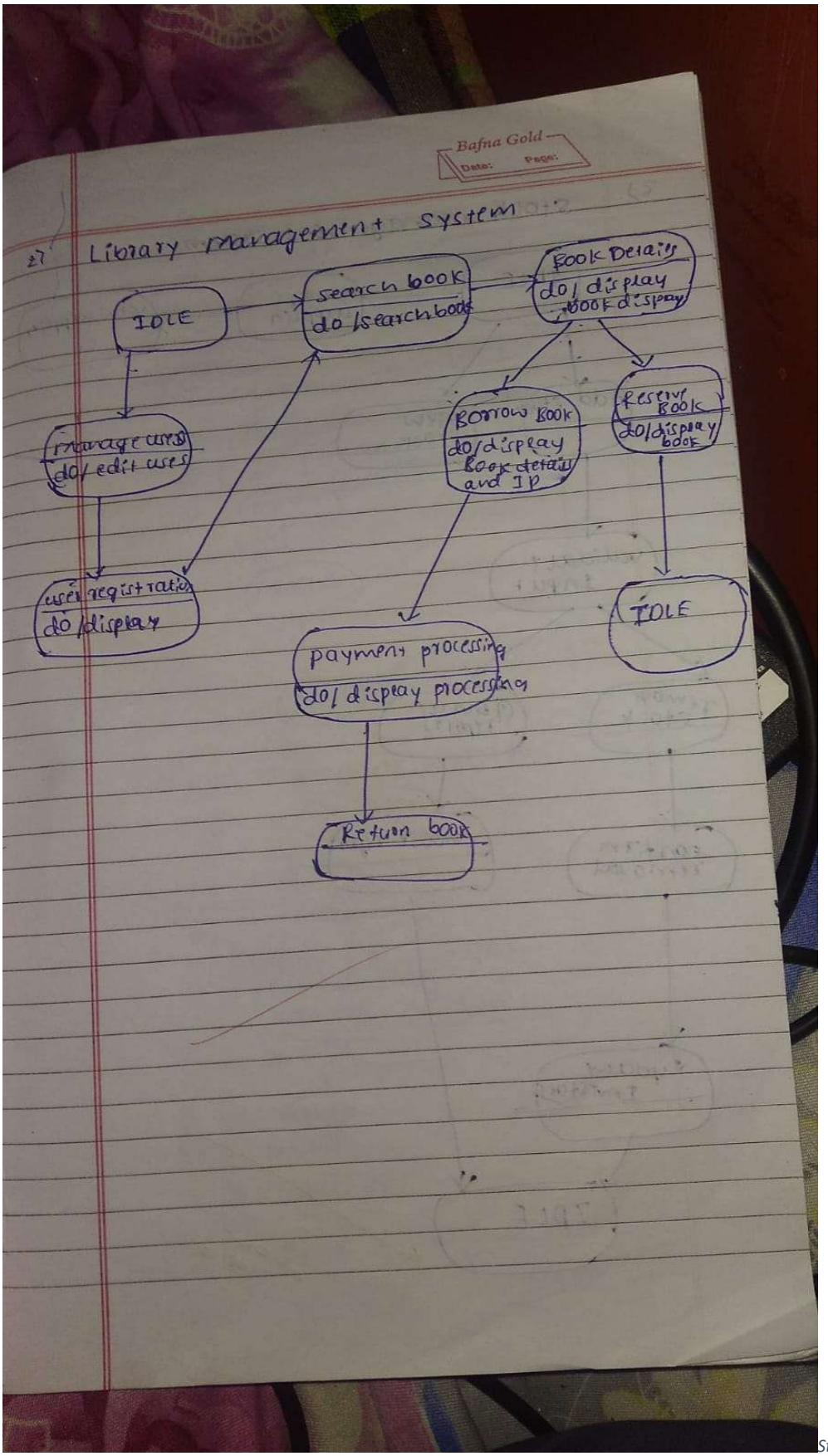


Figure 3.7: State Diagram Obs bk

Use - Case Diagram

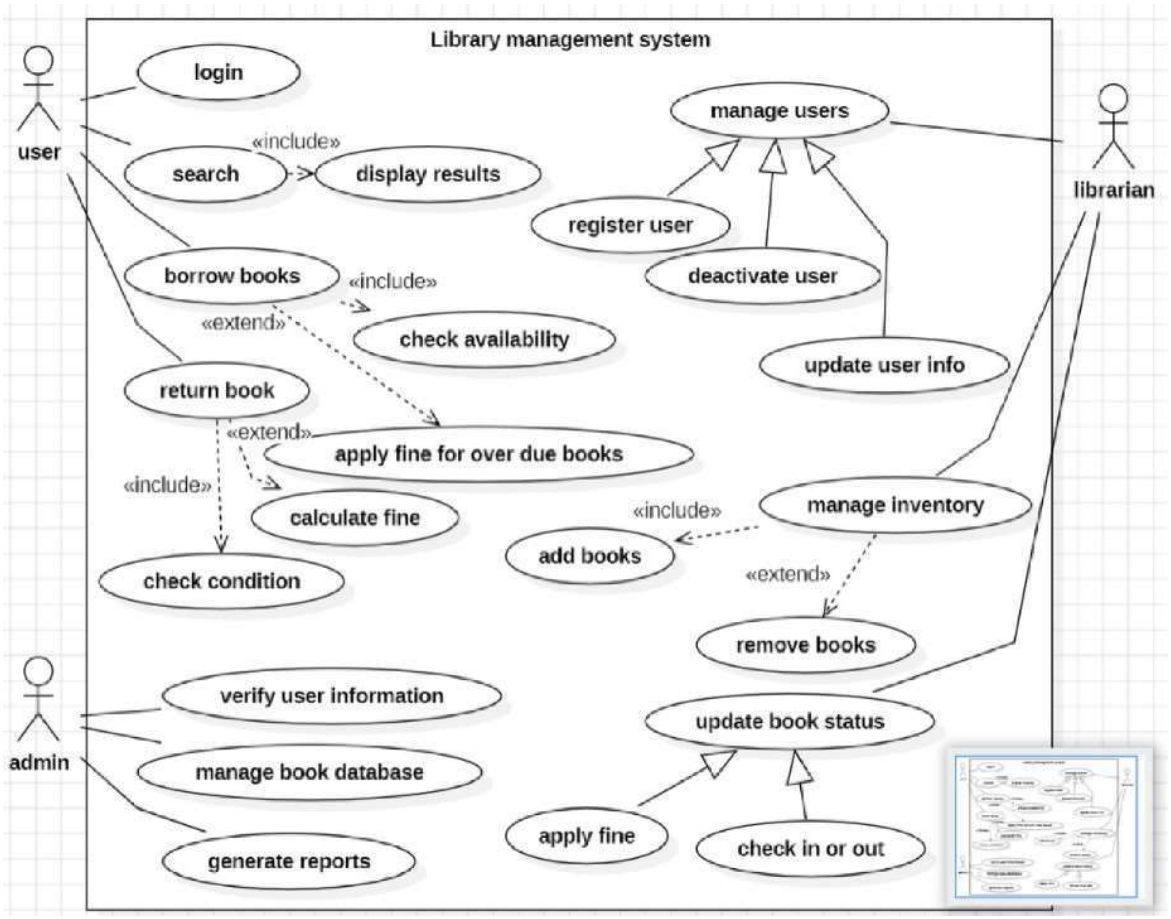


Figure 3.8: Use Case Diagram

Use Cases:

- Login: The process of authenticating a user to access the system.
- Search: Enables users to search for books in the library catalog.
- Display Results: Presents the search results to the user.
- Borrow Books: The process of issuing books to users.
- Return Book: The process of returning books to the library.
- Check Availability: Verifies the availability of a book before issuing.
- Apply Fine for Overdue Books: Calculates and applies fines for overdue books.
- Calculate Fine: Determines the amount of fine based on the overdue period.
- Check Condition: Checks the condition of returned books.

- Register User: Creates new user accounts in the system.
- Deactivate User: Deactivates user accounts.
- Update User Info: Updates user information, such as contact details.
- Manage Inventory: Manages the library's book collection.
- Add Books: Adds new books to the library catalog.
- Remove Books: Removes books from the library catalog.
- Update Book Status: Updates the status of books (e.g., available, checked out, reserved).
- Apply Fine: Applies fines to users for overdue books.
- Check In/Out: Processes book check-in and check-out operations.
- Verify User Information: Verifies the accuracy of user information.
- Manage Book Database: Performs administrative tasks related to the book database.
- Generate Reports: Generates various reports, such as usage statistics and overdue book reports.

Relationships:

- Include: Indicates that one use case includes another use case as a sub-step. For example, "Search" includes "Display Results."
- Extend: Indicates that a use case can be extended with optional behavior. For example, "Borrow Books" can be extended with "Check Availability."

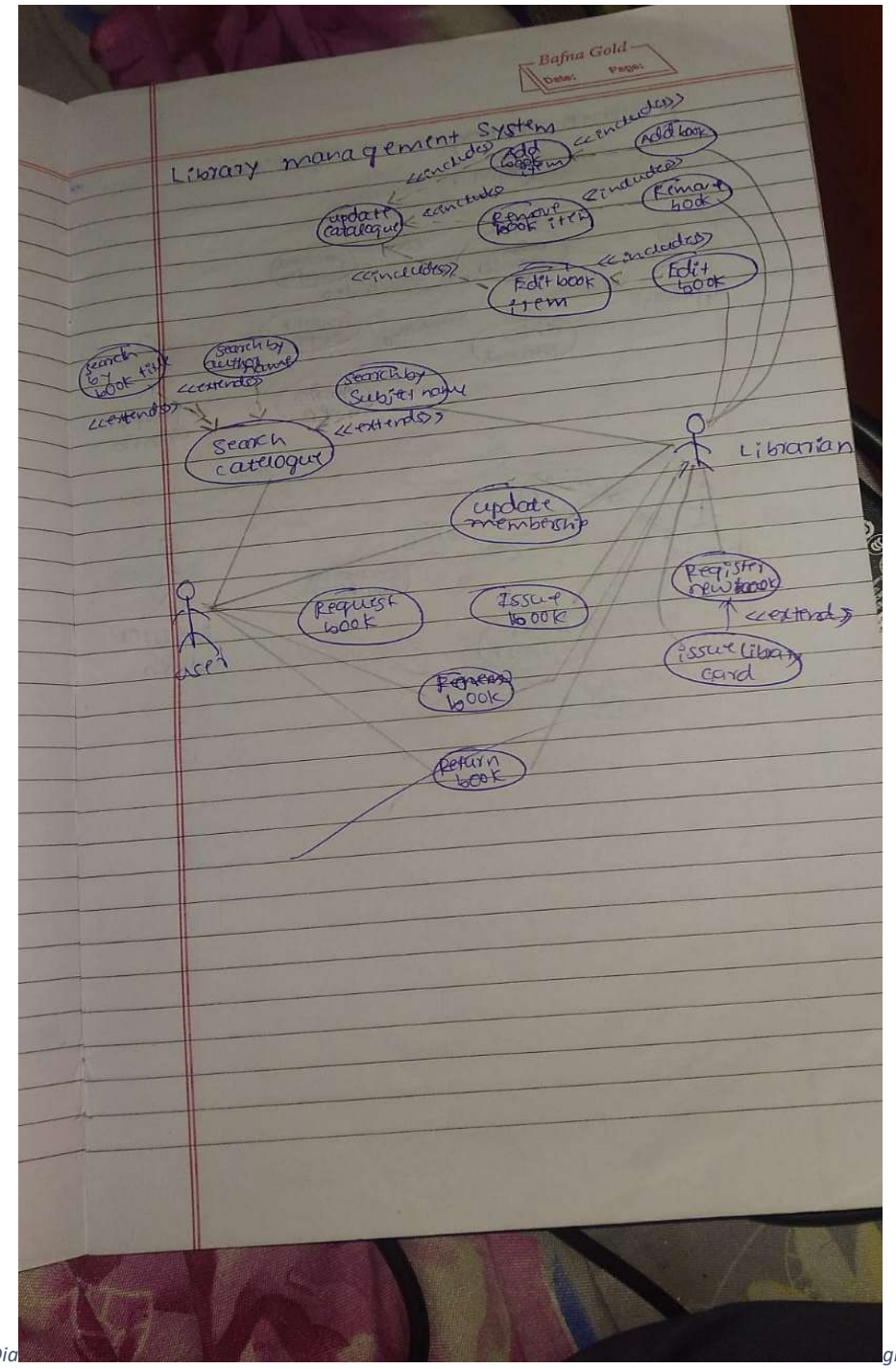


Figure 3.9: Use Case Diagram Obs

bk

Sequence Diagram

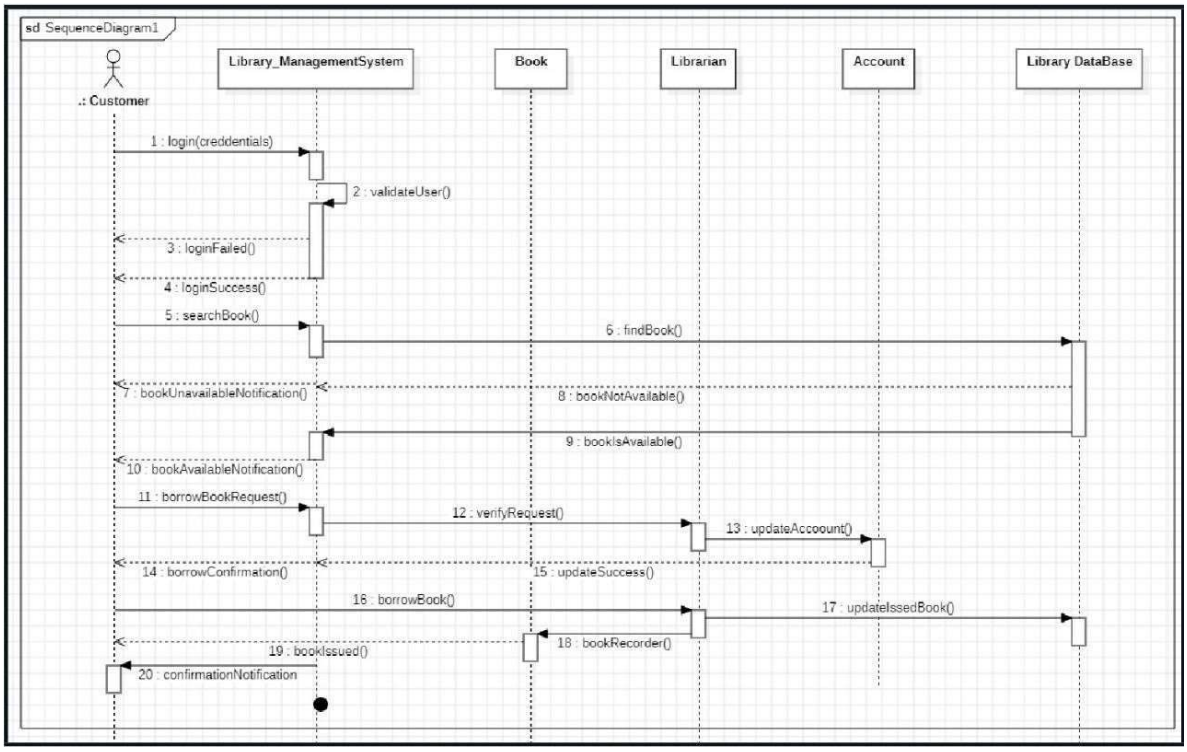
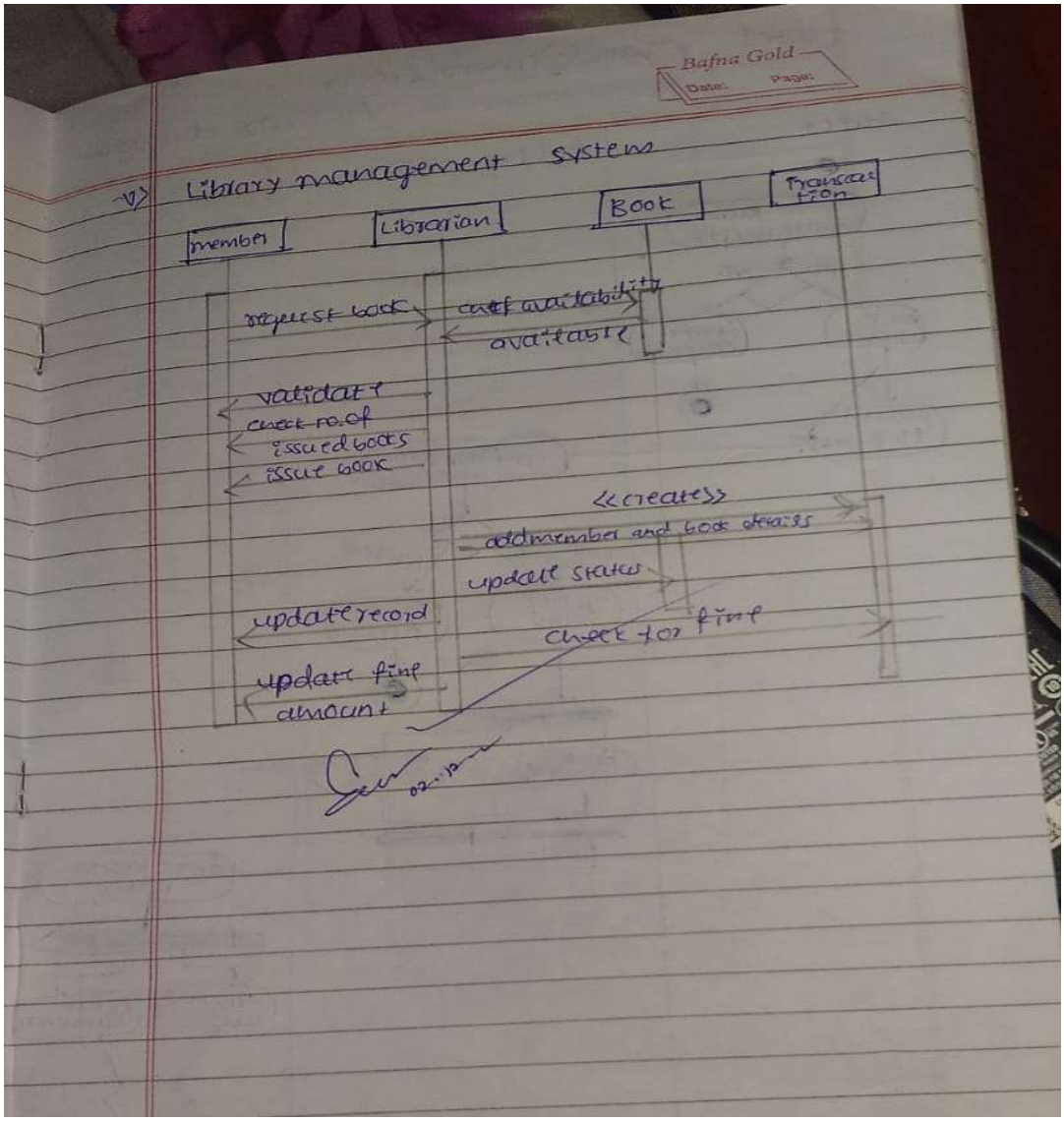


Figure 3.10: Sequence Diagram

1. **login(credentials):** The customer initiates the process by logging into the system with their credentials.
2. **validateUser():** The Library Management System validates the customer's credentials.
3. **loginFailed():** If the credentials are invalid, an error message is sent to the customer.
4. **loginSuccess():** If the credentials are valid, the customer is successfully logged in.
5. **searchBook():** The customer searches for a book in the system.
6. **findBook():** The Library Management System searches for the requested book in the Library Database.
7. **bookUnavailableNotification():** If the book is not available, a notification is sent to the customer.
8. **bookIsAvailable():** If the book is available, the system proceeds to the next step.
9. **bookAvailableNotification():** A notification is sent to the customer indicating that the book is available.
10. **borrowBookRequest():** The customer requests to borrow the available book.
11. **verifyRequest():** The Library Management System verifies the borrowing request.
12. **updateAccount():** The system updates the customer's account to reflect the borrowed book.

13. **borrowConfirmation():** A confirmation message is sent to the customer.
14. **borrowBook():** The Library Management System records the book as borrowed.
15. **updateIssuedBook():** The Library Database updates the status of the book to "issued."
16. **bookRecorded():** The system records the borrowing transaction in the Library Database.
17. **confirmationNotification():** A confirmation notification is sent to the customer.

Figure 3..11: Sequence Diagram Obs bk



Activity Diagram

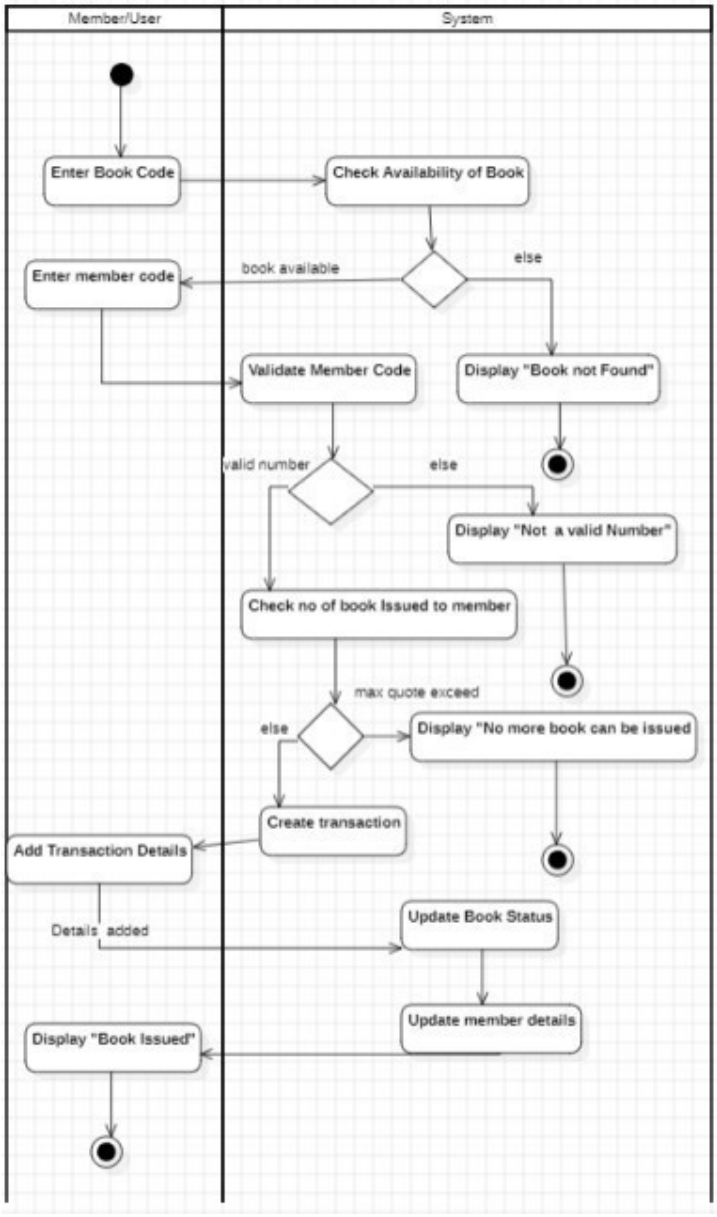


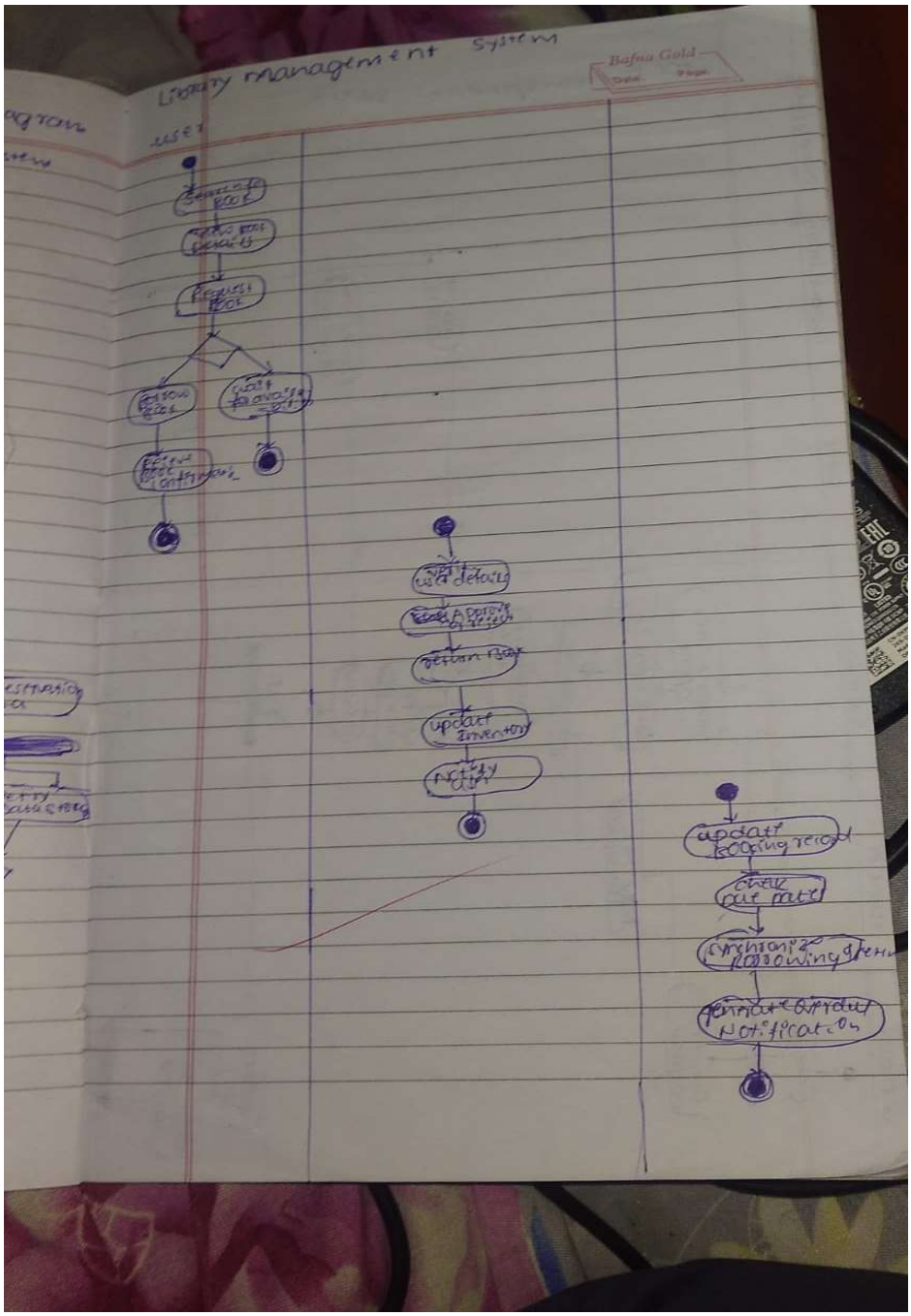
Figure 3.12: Activity Diagram

1. Enter Book Code: The user enters the code of the book they wish to borrow.
2. Check Availability of Book: The system checks if the book is available for borrowing.
   - Book Available: If the book is available, the process proceeds to the next step.
   - Book Not Found: If the book is not found in the system, an error message is displayed.
3. Enter Member Code: The user enters their member code.
4. Validate Member Code: The system validates the entered member code.

- Valid Number: If the member code is valid, the process proceeds to the next step.
- Not a Valid Number: If the member code is invalid, an error message is displayed.

5. Check No of Book Issued to Member: The system checks if the member has reached the maximum number of books allowed to borrow.
    - Max Quote Exceeded: If the member has exceeded the borrowing limit, an error message is displayed.
    - Within Limit: If the member is within the borrowing limit, the process proceeds to the next step.
6. Create Transaction: A new transaction is created in the system to record the book borrowing.
7. Add Transaction Details: Details of the transaction, such as book code, member code, and issue date, are added to the transaction record.
8. Update Book Status: The status of the book is updated in the system to "issued."
9. Update Member Details: The member's borrowing history is updated in the system to reflect the new book.
10. Display "Book Issued": A success message is displayed to the user, indicating that the book has been successfully issued.

Figure 3.13: Activity Diagram Obs bk



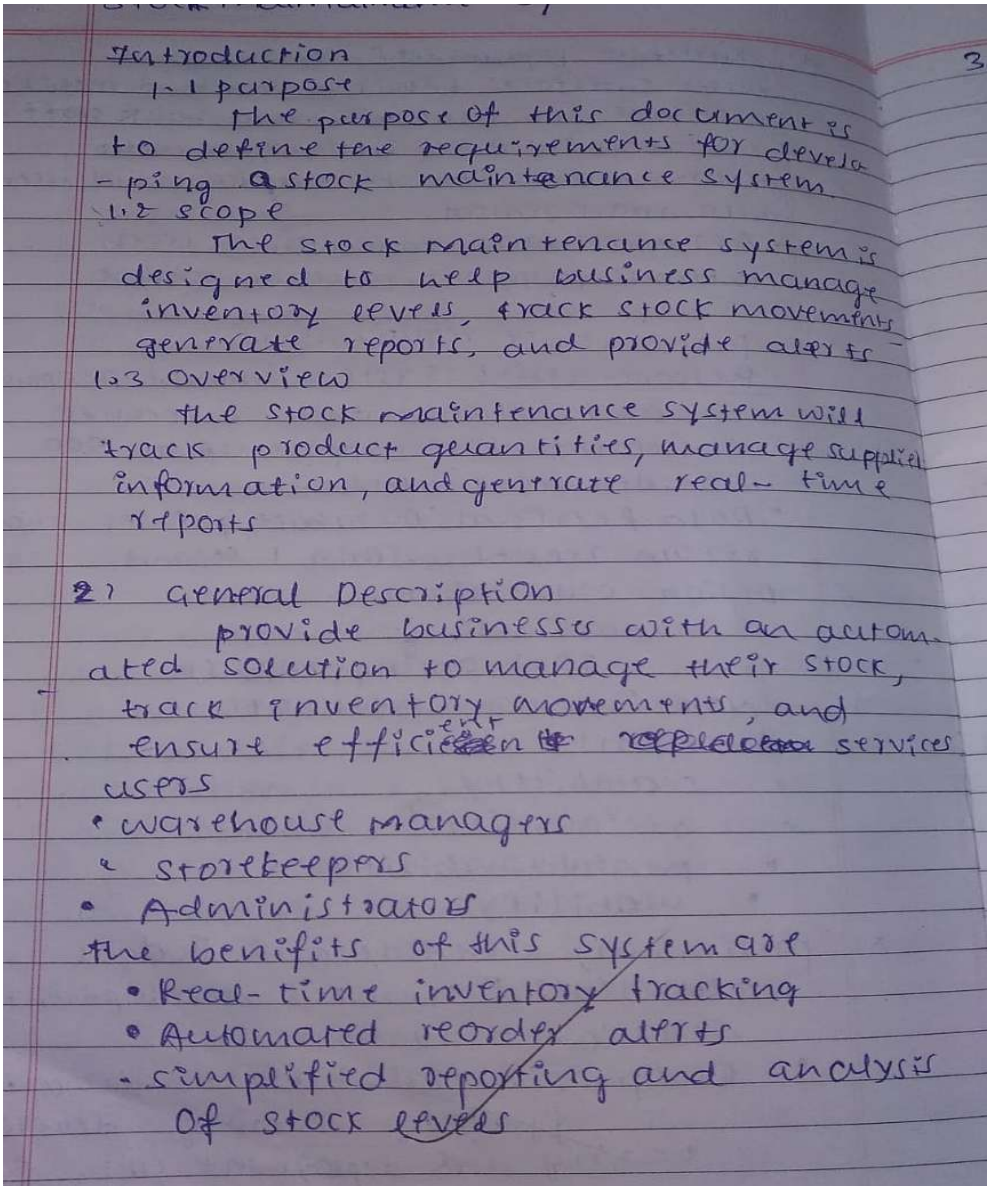
4. Stock Maintenance System

Problem Statement

Effective stock management is critical for businesses to ensure optimal inventory levels, reduce wastage, and meet customer demands. However, traditional manual or semi-automated methods of stock maintenance often result in challenges such as inaccurate inventory tracking, overstocking or understocking, delayed replenishments, and inefficient reporting. These issues can lead to increased operational costs, missed sales opportunities, and poor decision-making due to a lack of real-time data and analytics.

SRS – Software Requirements Specification

Figure 4.1: SRS Document Observation Bk (1)



Introduction 3

1.1 purpose

The purpose of this document is to define the requirements for developing a stock maintenance system

1.2 scope

The stock maintenance system is designed to help business manage inventory levels, track stock movements, generate reports, and provide alerts

1.3 overview

the stock maintenance system will track product quantities, manage supplier information, and genrate real-time reports

2) General Description

provide businesses with an automated solution to manage their stock, track inventory movements, and ensure efficient services

users

- warehouse managers
- storekeepers
- Administrators

the benifits of this system are

- Real-time inventory tracking
- Automated reorder alerts
- simplified reporting and analysis of stock levels

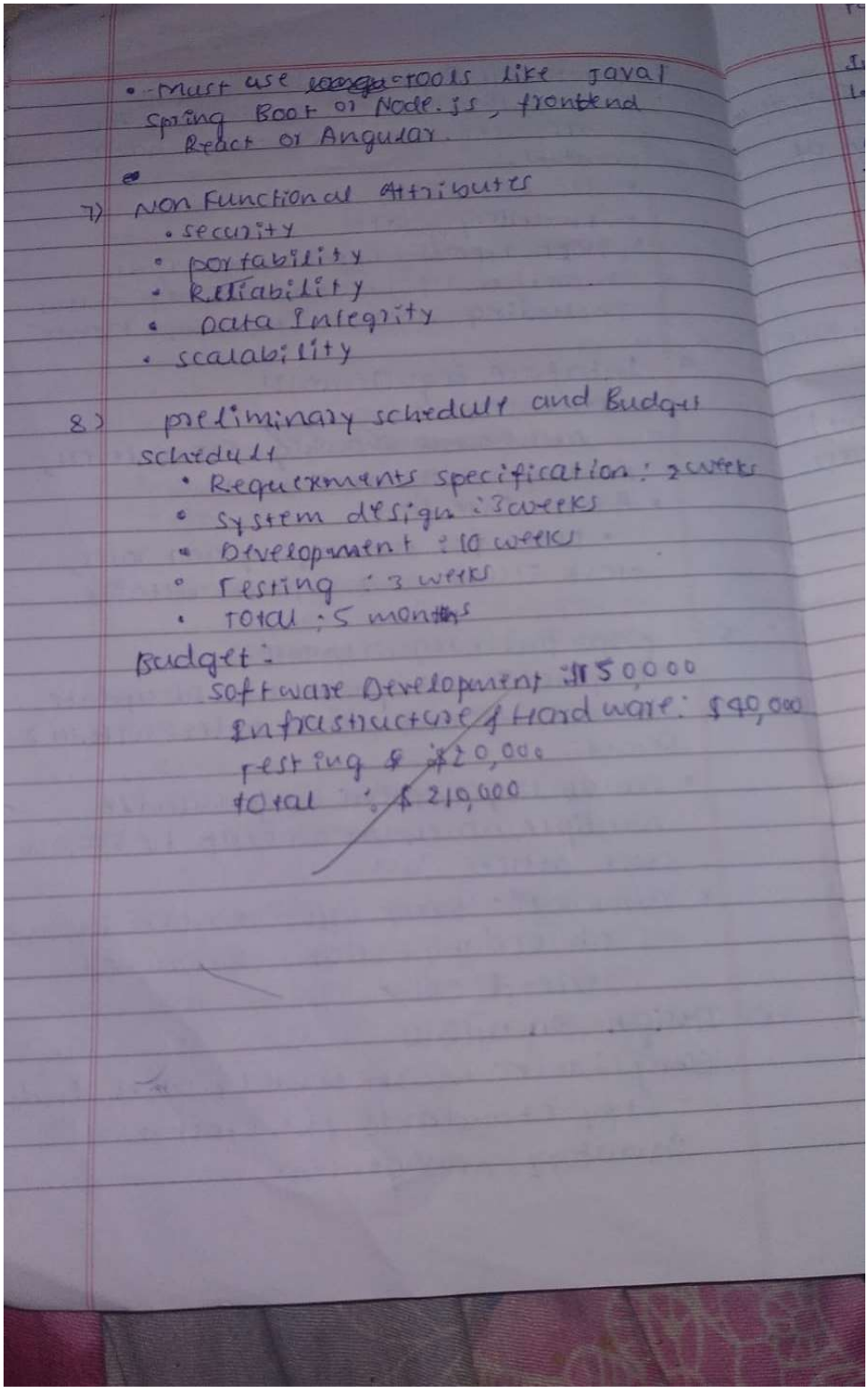
3. Functional Requirements
- User Authentication: Secure login for all users with role-based access control.
- Stock-tracking
- Inventory updates
- Stock reports: Generate daily, or monthly reports on inventory status including stock levels, usage trends.

4. Interface Requirements
- User Interface
  Dashboard showing stock levels, pending orders and alerts
- Admin Interface
  - Tools for managing user roles, stock items and supplier details

5. Performance requirements
- Response Time: System should update stock levels and process orders within 2 seconds.
- Scalability: System must handle multiple warehouse and up to 500,000 stock entries
- Error rate: Error rate for stock mismatch or duplication should not exceed 0.01%

6. Design constraints
Compliance: must comply with industry standards for stock and inventory management.

Figure 4.2: SRS Document Observation Bk (2)



- Must use ~~comp~~ tools like Java Spring Boot or Node.js, frontend React or Angular.

7) Non Functional Attributes
- security
- portability
- Reliability
- Data Integrity
- scalability

8) preliminary schedule and Budget
schedule
- Requirments specification : 2 weeks
- System design : 3 weeks
- Development : 10 weeks
- Testing : 3 weeks
- Total : 5 months

Budget :
Software Development : $50,000
Infrastructure / Hardware : $40,000
testing : $20,000
total : $ 210,000

Class Diagram

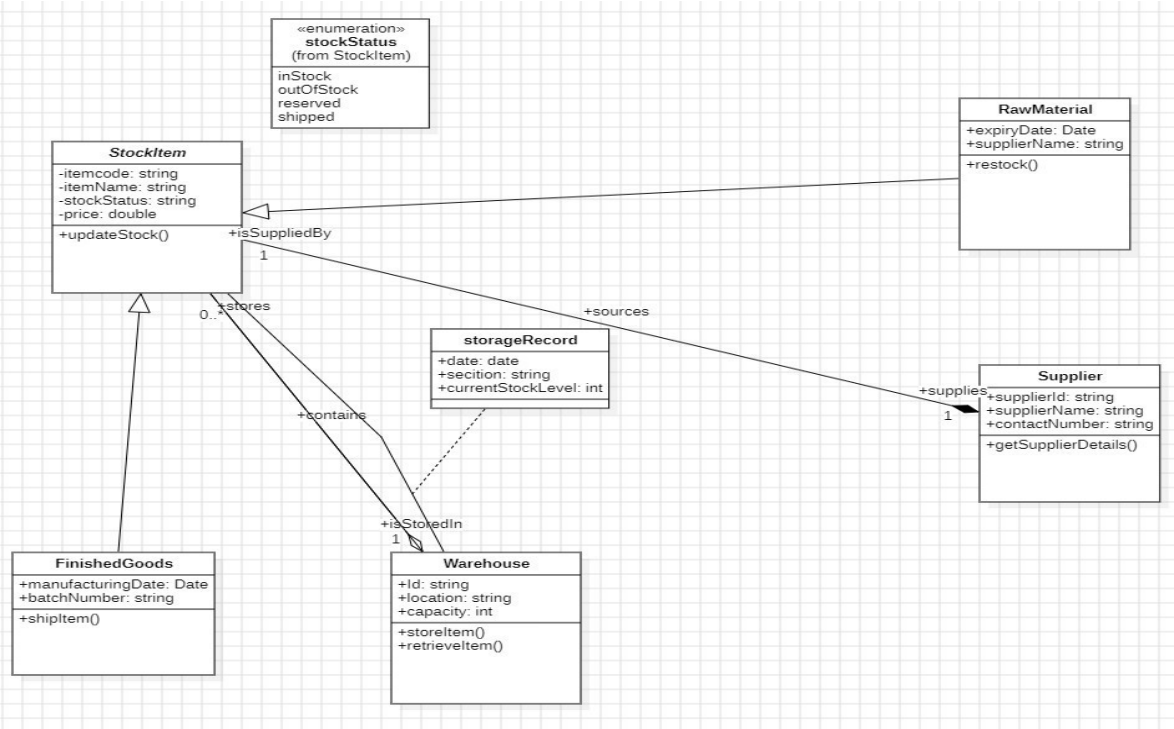


Figure 4.4: Class Diagram

- StockItem has attributes for item code, name, stock status, and price. It can be updated.
- RawMaterial is a type of StockItem with additional attributes for expiry date and supplier name. It can be restocked.
- FinishedGoods is another type of StockItem with attributes for manufacturing date and batch number. It can be shipped.
- Supplier has attributes for supplier ID, name, and contact number. It can provide supplier details.
- Warehouse has attributes for ID, location, and capacity. It can store and retrieve items.
- StorageRecord stores information about the storage of a StockItem, including date, section, and current stock level.

Relationships:

- Each StockItem is supplied by one Supplier.
- A Supplier can supply multiple StockItems.
- A Warehouse can contain multiple StockItems.

- A StockItem can have multiple StorageRecords.
- Each StorageRecord is associated with one Warehouse.

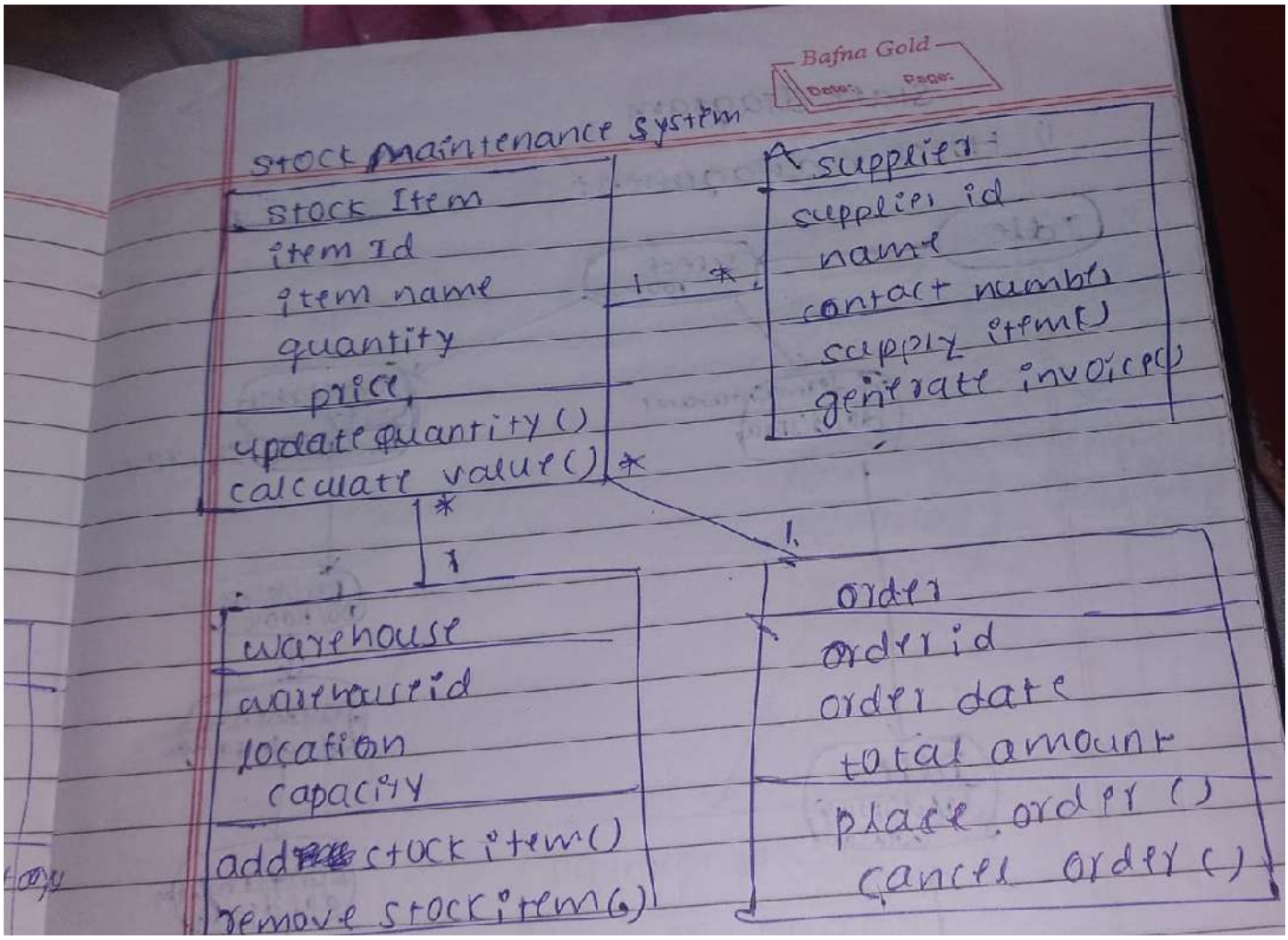


Figure 4.5: Class Diagram Obs bk

State Diagram

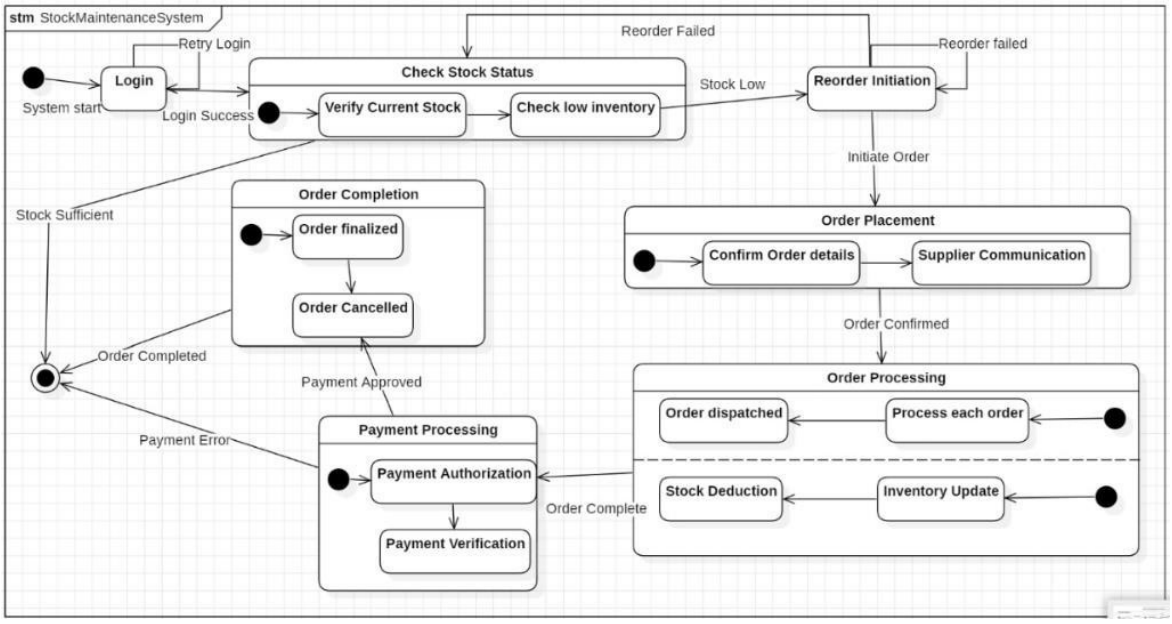
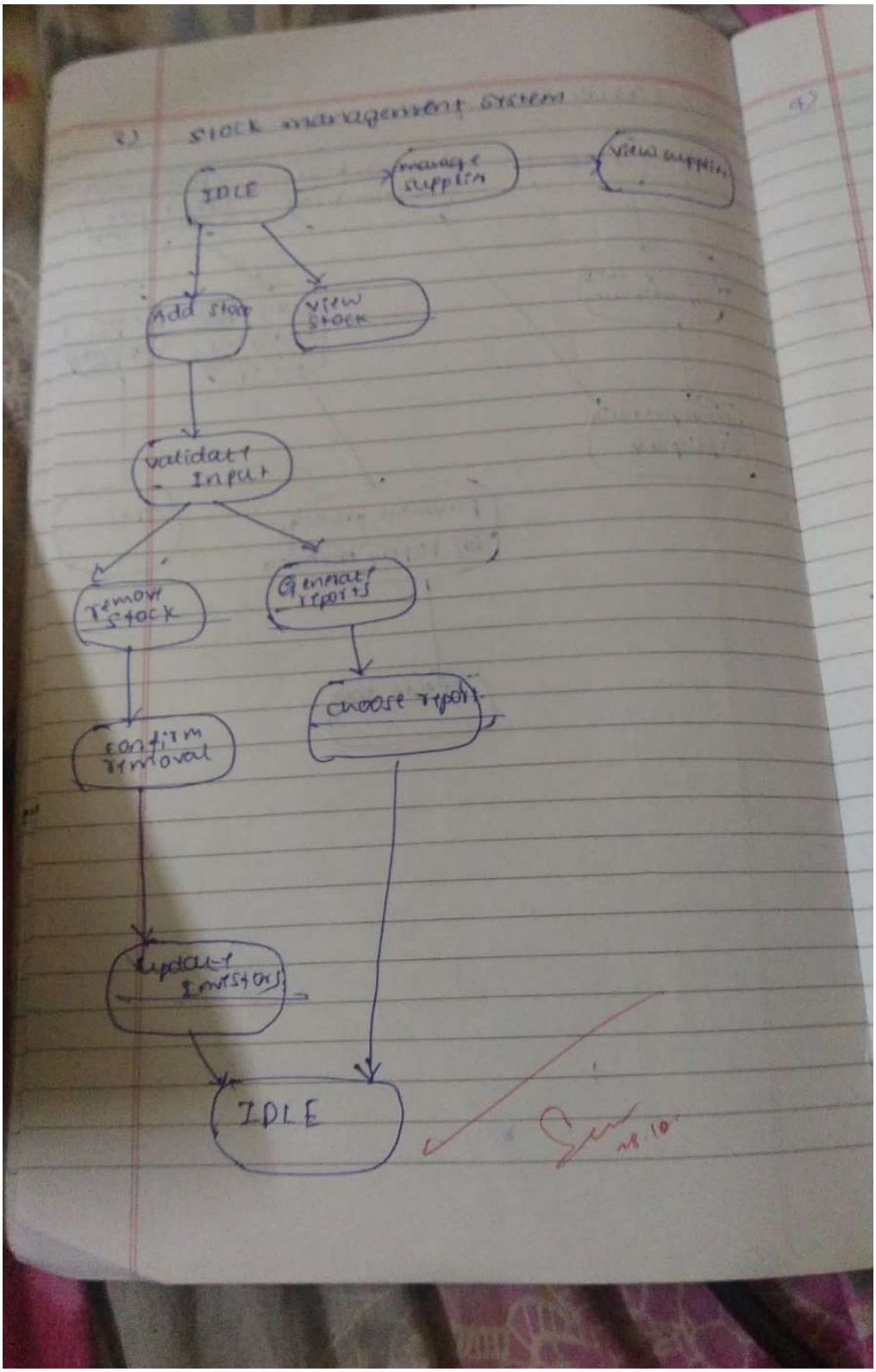


Figure 4.6: State Diagram

1. **System Start:** The initial state of the system.
2. **Login:** The state where the user attempts to log in to the system.
3. **Login Success:** The state reached after successful user authentication.
4. **Check Stock Status:** The state where the system verifies the current stock levels.
5. **Stock Sufficient:** The state indicating that stock levels are sufficient.
6. **Stock Low:** The state indicating that stock levels are below the reorder point.
7. **Reorder Initiation:** The state where the system initiates a reorder request.
8. **Order Placement:** The state where the order is being placed with the supplier.
9. **Order Confirmed:** The state indicating that the order has been confirmed by the supplier.
10. **Order Processing:** The state where the supplier is processing the order.
11. **Order Dispatched:** The state indicating that the order has been dispatched by the supplier.
12. **Stock Deduction:** The state where the system deducts the ordered quantity from the stock.
13. **Inventory Update:** The state where the inventory levels are updated to reflect the received order.
14. **Order Completed:** The final state indicating that the entire order process has been completed successfully.
15. **Order Cancelled:** The state indicating that the order has been canceled.
16. **Payment Processing:** The state where the payment for the order is being processed.
17. **Payment Authorization:** The state where the payment for the order is being authorized.

18. **Payment Verification:** The state where the payment for the order is being verified.
19. **Payment Error:** The state indicating that a payment error has occurred.
20. **Reorder Failed:** The state indicating that the reorder process has failed.

Figure 4.7: State Diagram Obs bk



Use - Case Diagram

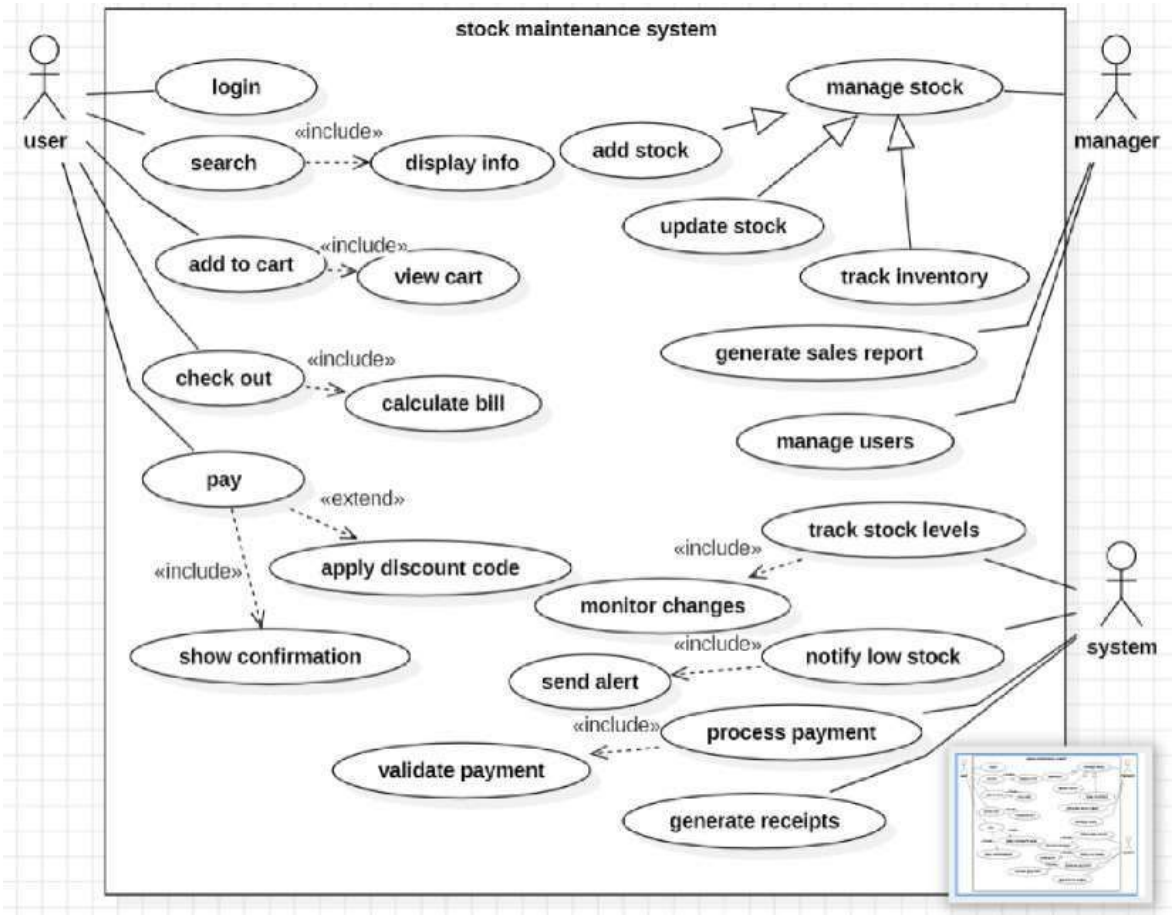


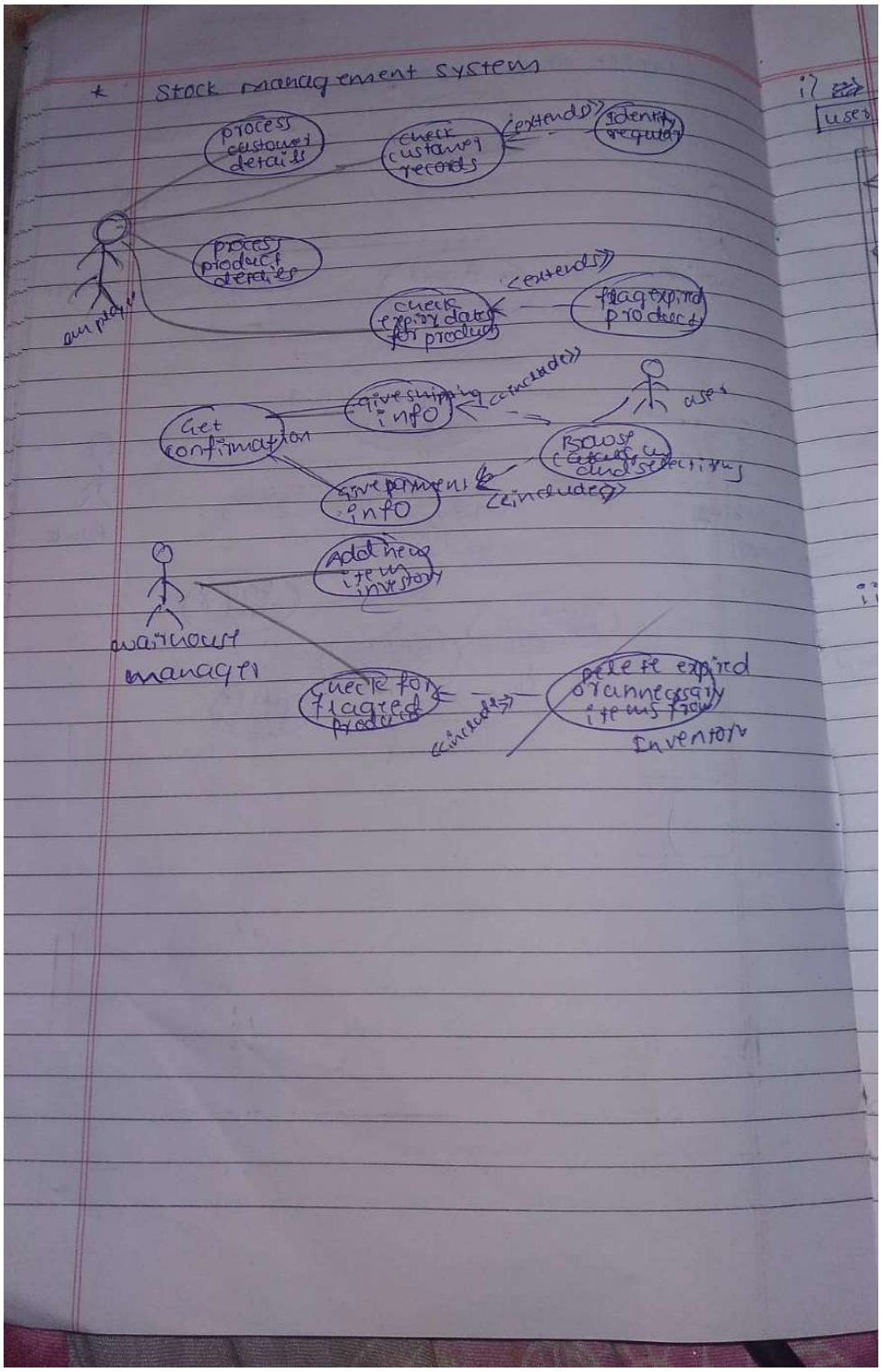
Figure 4.8: Use Case Diagram

Use Cases:

- **Login:** The process of authenticating a user to access the system.
- **Search:** Enables users to search for products in the inventory.
- **Display Info:** Displays detailed information about a product.
- **Add to Cart:** Allows users to add products to their shopping cart.
- **View Cart:** Displays the contents of the user's shopping cart.
- **Check Out:** The process of initiating the checkout process.
- **Calculate Bill:** Calculates the total cost of items in the shopping cart.
- **Pay:** The process of making a payment for the order.
- **Apply Discount Code:** Allows users to apply discount codes to their orders.

- **Show Confirmation:** Displays a confirmation message after a successful order.
- **Manage Stock:** Includes use cases for adding, updating, and tracking stock levels.
- **Add Stock:** Allows managers to add new products to the inventory.
- **Update Stock:** Allows managers to update product information and stock levels.
- **Track Inventory:** Monitors stock levels and generates reports.
- **Generate Sales Report:** Generates reports on sales data.
- **Manage Users:** Allows managers to manage user accounts (e.g., create, edit, delete).
- **Track Stock Levels:** Monitors stock levels and notifies relevant personnel when stock is low.
- **Notify Low Stock:** Sends alerts when stock levels fall below a certain threshold.
- **Monitor Changes:** Monitors changes in stock levels and other relevant data.
- **Process Payment:** Processes payment transactions.
- **Validate Payment:** Validates payment information.
- **Generate Receipts:** Generates receipts for completed orders.

Figure 4.9: Use Case Diagram Obs bk



Sequence Diagram

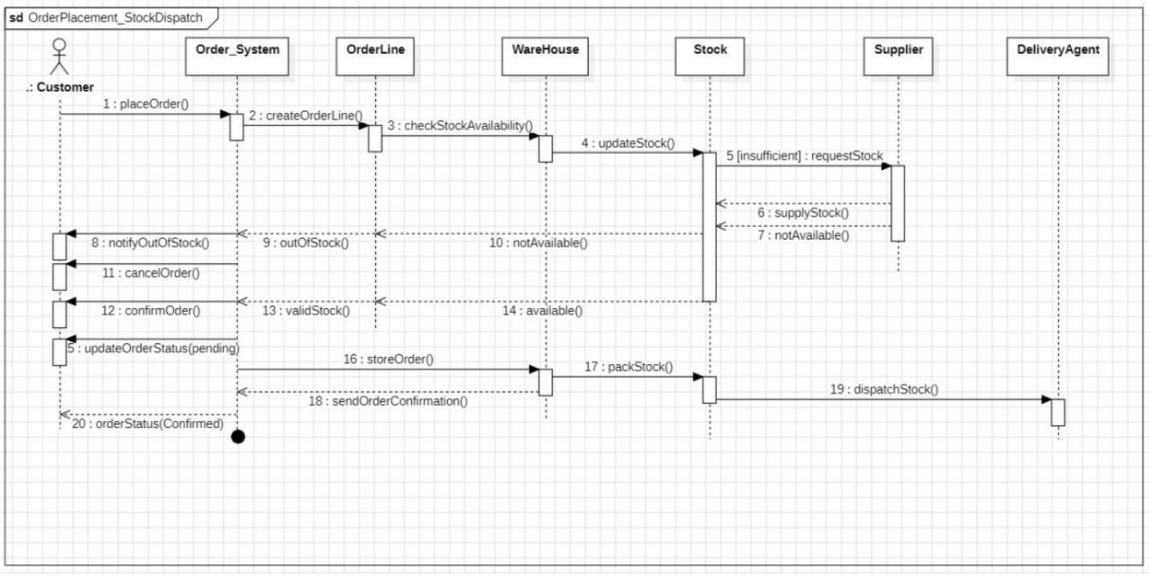
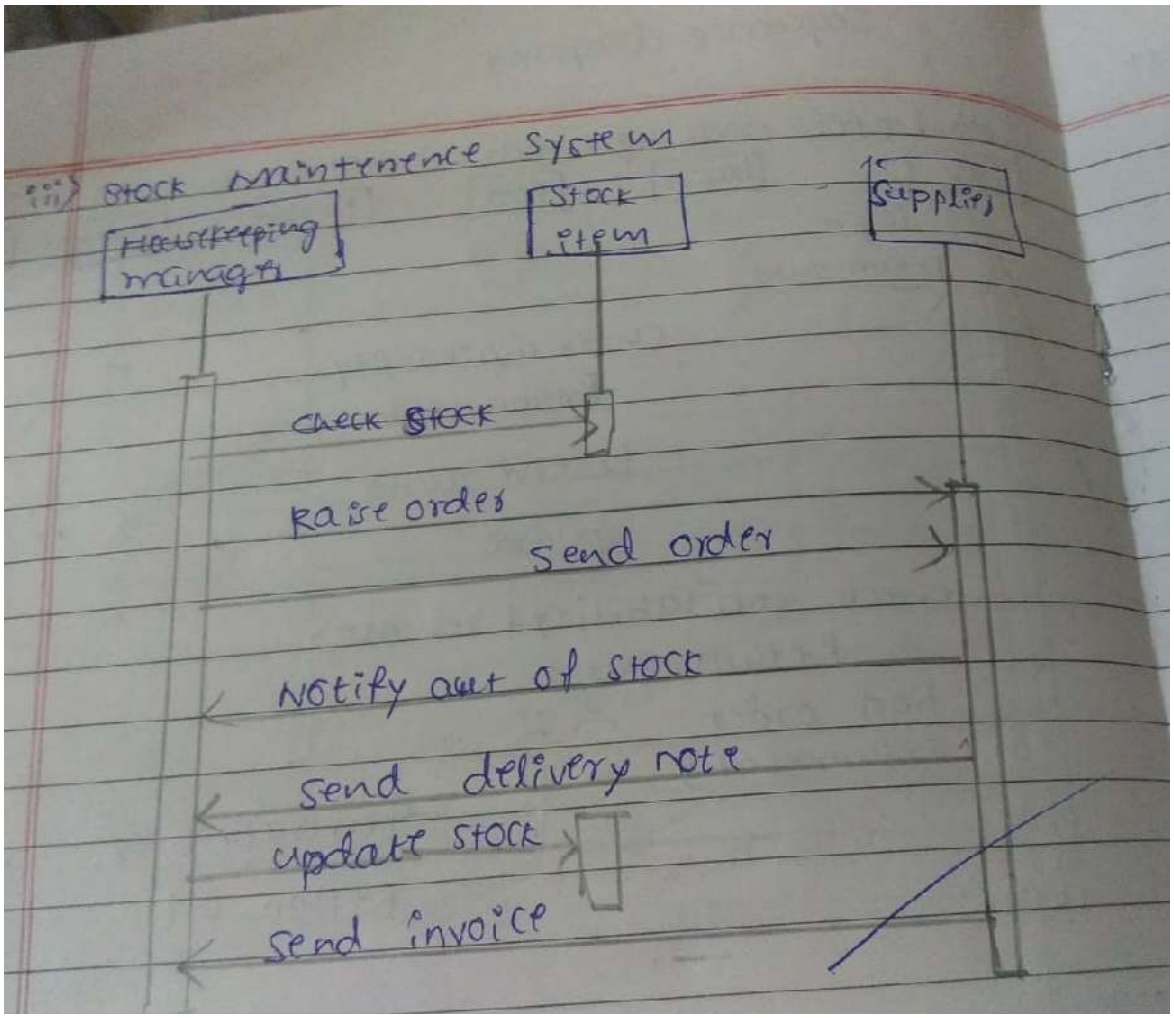


Figure 4.10: Sequence Diagram

1. placeOrder(): The customer initiates the process by placing an order.
2. createOrderLine(): The Order System creates an order line for each item in the order.
3. checkStockAvailability(): The Order Line checks the stock availability for each item in the warehouse.
4. updateStock(): The Warehouse updates the stock quantity based on the order line.
5. insufficient, requestStock(): If the stock is insufficient, the Warehouse requests more stock from the Supplier.
6. supplyStock(): The Supplier supplies the requested stock to the Warehouse.
7. notAvailable(): If the stock is not available, the system notifies the Customer.
8. notifyOutOfStock(): The Order System notifies the Customer about the out-of-stock items.
9. outOfStock(): The Order Line updates the status of the out-of-stock items.
10. notAvailable(): The Customer is notified about the unavailable items.
11. cancelOrder(): The Customer cancels the order.
12. confirmOrder(): The Order System confirms the order.
13. validStock(): The Order Line verifies that the stock is available for all items.
14. available(): The Warehouse confirms that the stock is available.
15. updateOrderStatus(pending): The Order System updates the order status to "pending."
16. storeOrder(): The Warehouse stores the order for processing.
17. packStock(): The Warehouse packs the ordered items for shipment.
18. sendOrderConfirmation(): The Order System sends an order confirmation to the Customer.

19. dispatchStock(): The Delivery Agent dispatches the order.
20. orderStatus(Confirmed): The Order System updates the order status to "Confirmed."

Figure 4.11: Sequence Diagram Obs bk



Activity Diagram

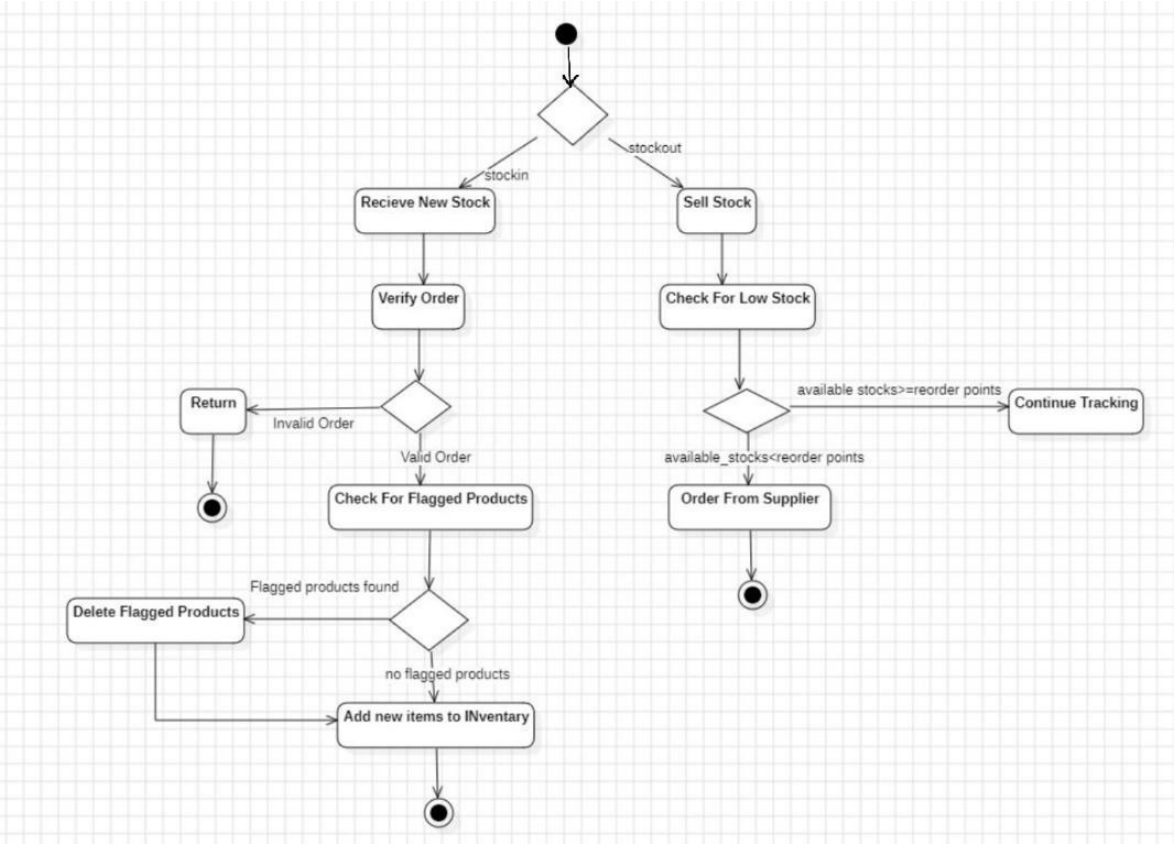
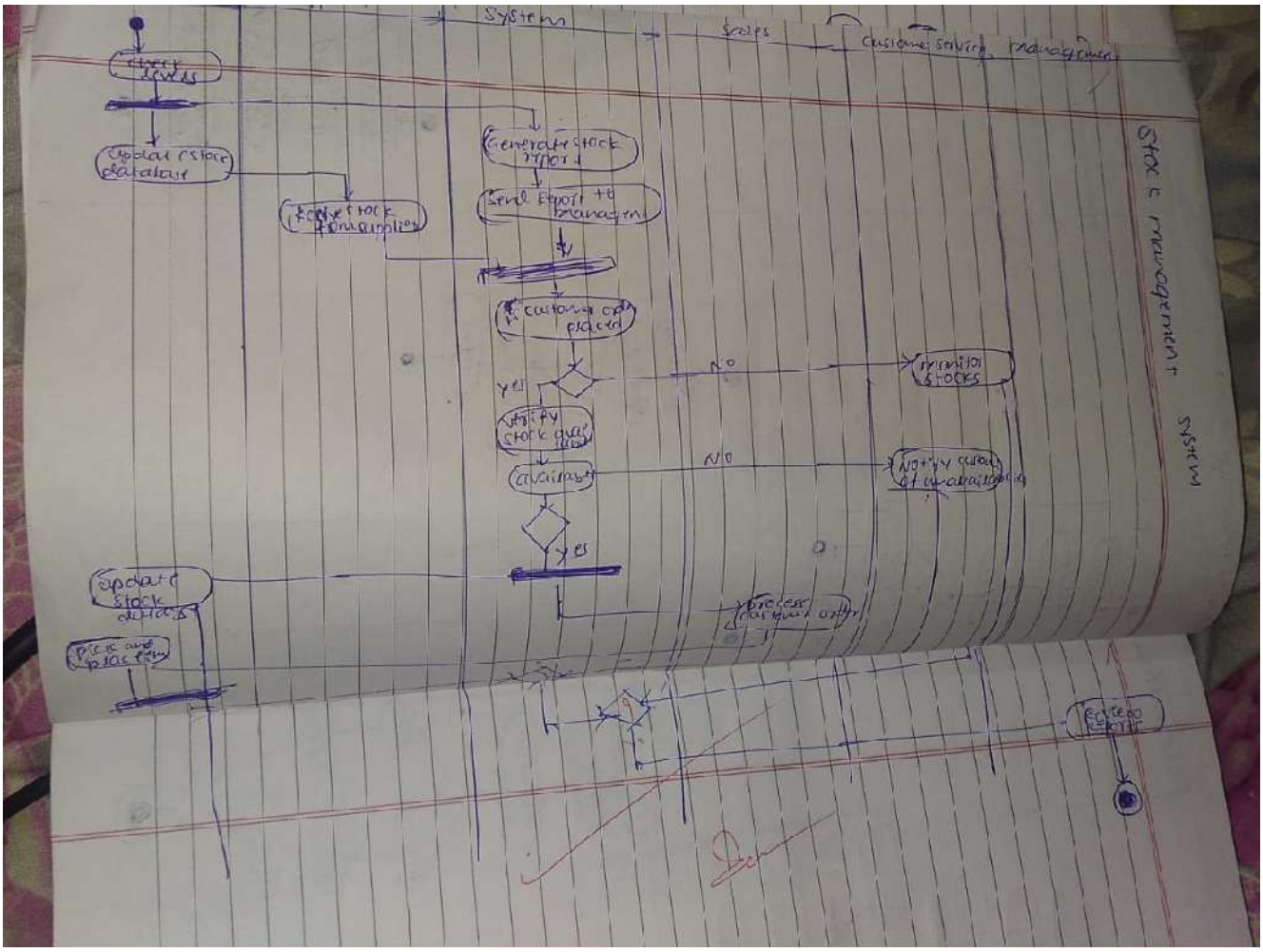


Figure 4.12: Activity Diagram

1. **Stock In/Out:** This is the initial state where the system can either receive new stock or sell stock.
2. **Receive New Stock:** This activity represents the process of receiving new stock into the inventory.
3. **Verify Order:** This activity checks the validity of an incoming order or a sales order.
4. **Check For Low Stock:** This activity checks if the stock levels of any products have fallen below the reorder point.
5. **Order From Supplier:** If stock levels are below the reorder point, this activity triggers an order to be placed with the supplier to replenish stock.
6. **Check For Flagged Products:** This activity checks if any products in the inventory have been flagged for removal or other actions.
7. **Delete Flagged Products:** If any flagged products are found, this activity removes them from the inventory.
8. **Add New Items to Inventory:** This activity adds new items to the inventory after receiving new stock or after removing flagged products.

9. **Sell Stock:** This activity represents the process of selling stock from the inventory.
10. **Continue Tracking:** This activity represents the ongoing monitoring and management of inventory levels.

Figure 4.13: Activity Diagram Obs bk



5. Passport Automation System

Problem Statement

The manual processing of passport applications is often time-consuming, prone to errors, and inefficient, leading to delays in issuance and renewal. Applicants face challenges such as lengthy queues, incomplete information, and lack of real-time updates on application status. Additionally, managing large volumes of data manually can result in errors, security concerns, and difficulty in tracking and retrieving records. These inefficiencies hinder the overall process and compromise user satisfaction and operational effectiveness.

SRS – Software Requirements Specification

Figure 5.1: SRS Document Observation Bk (1)

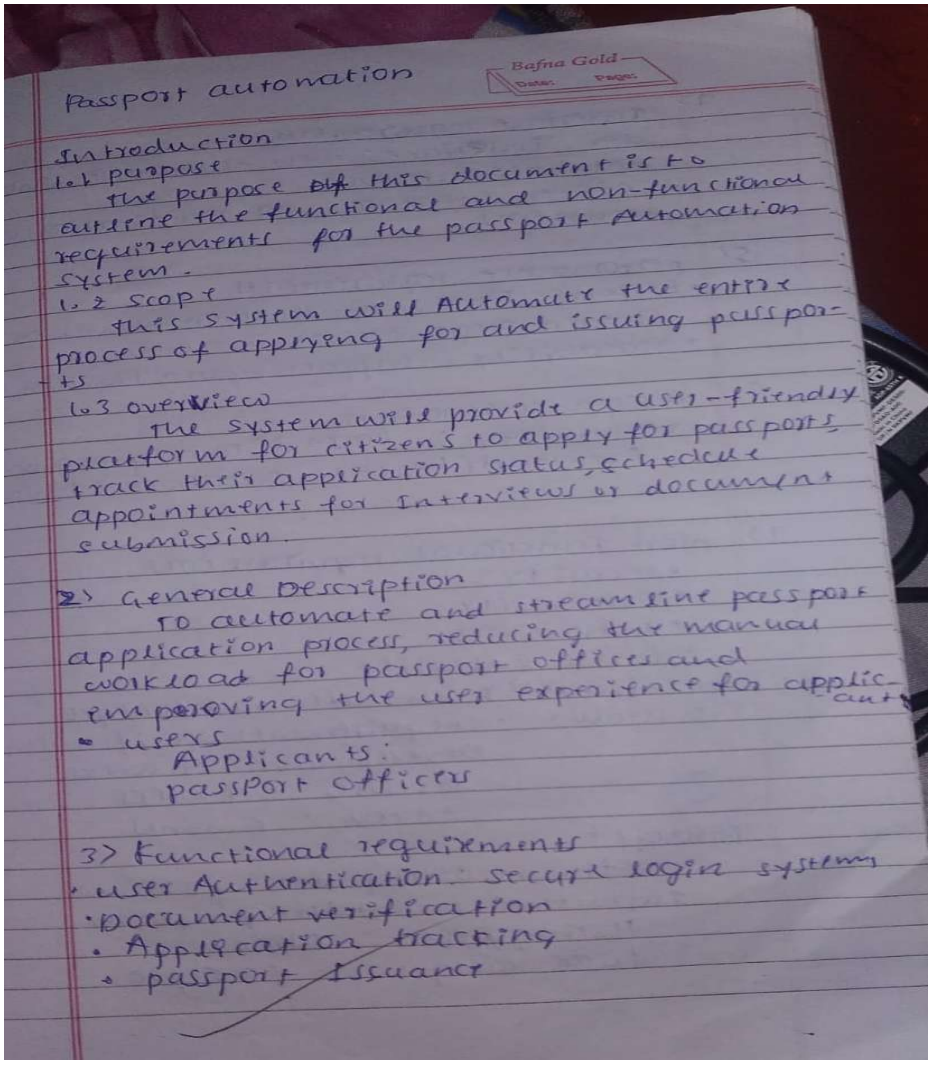
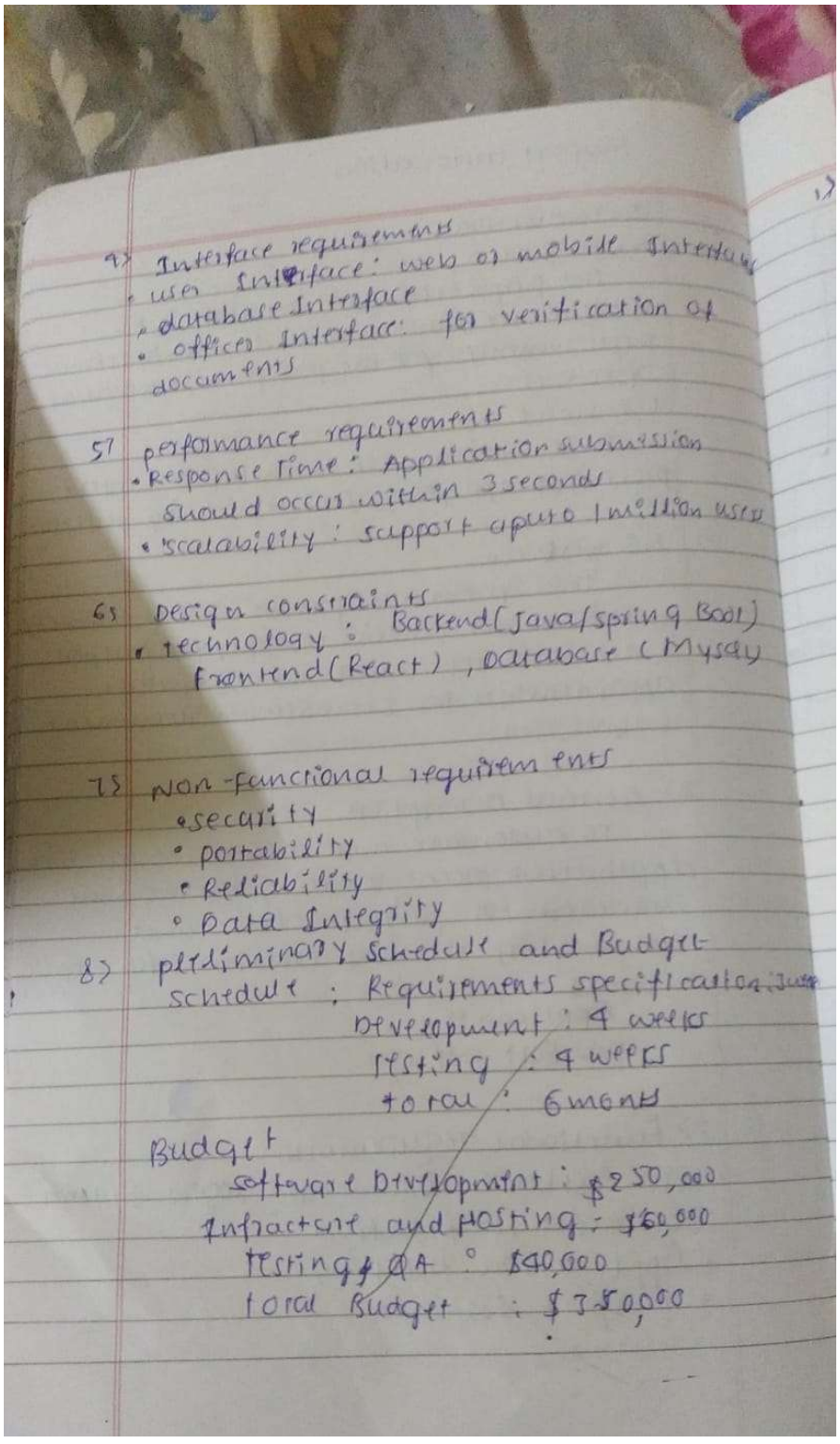


Figure 5.2: SRS Document Observation Bk (2)



Class Diagram

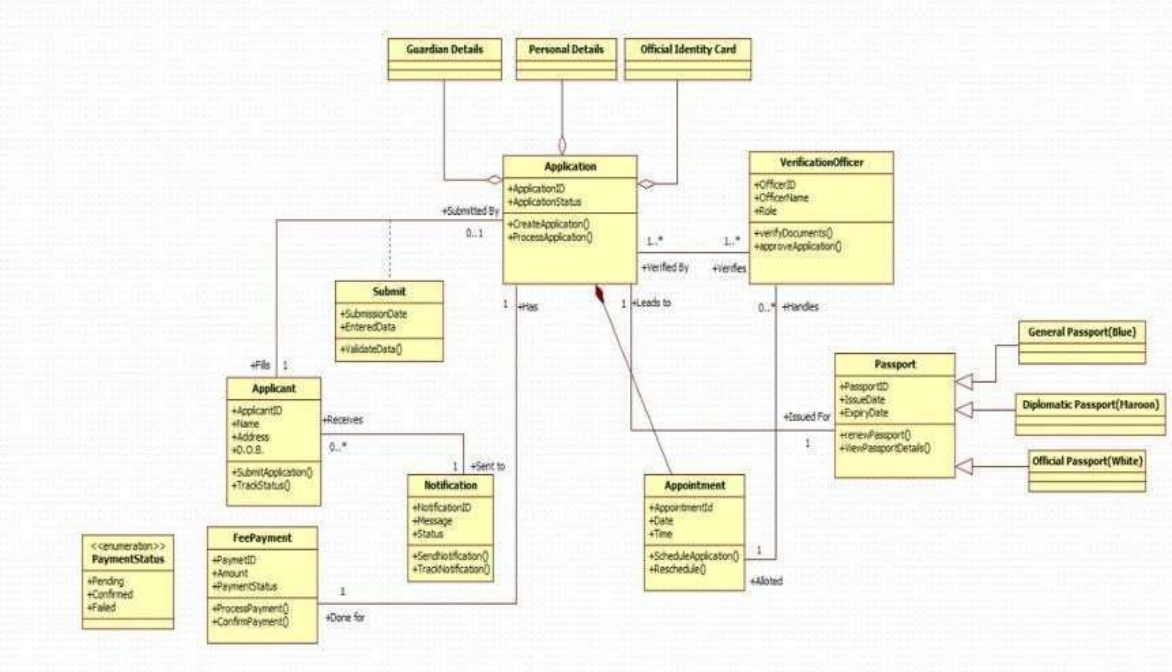


Figure 5.4: Class Diagram

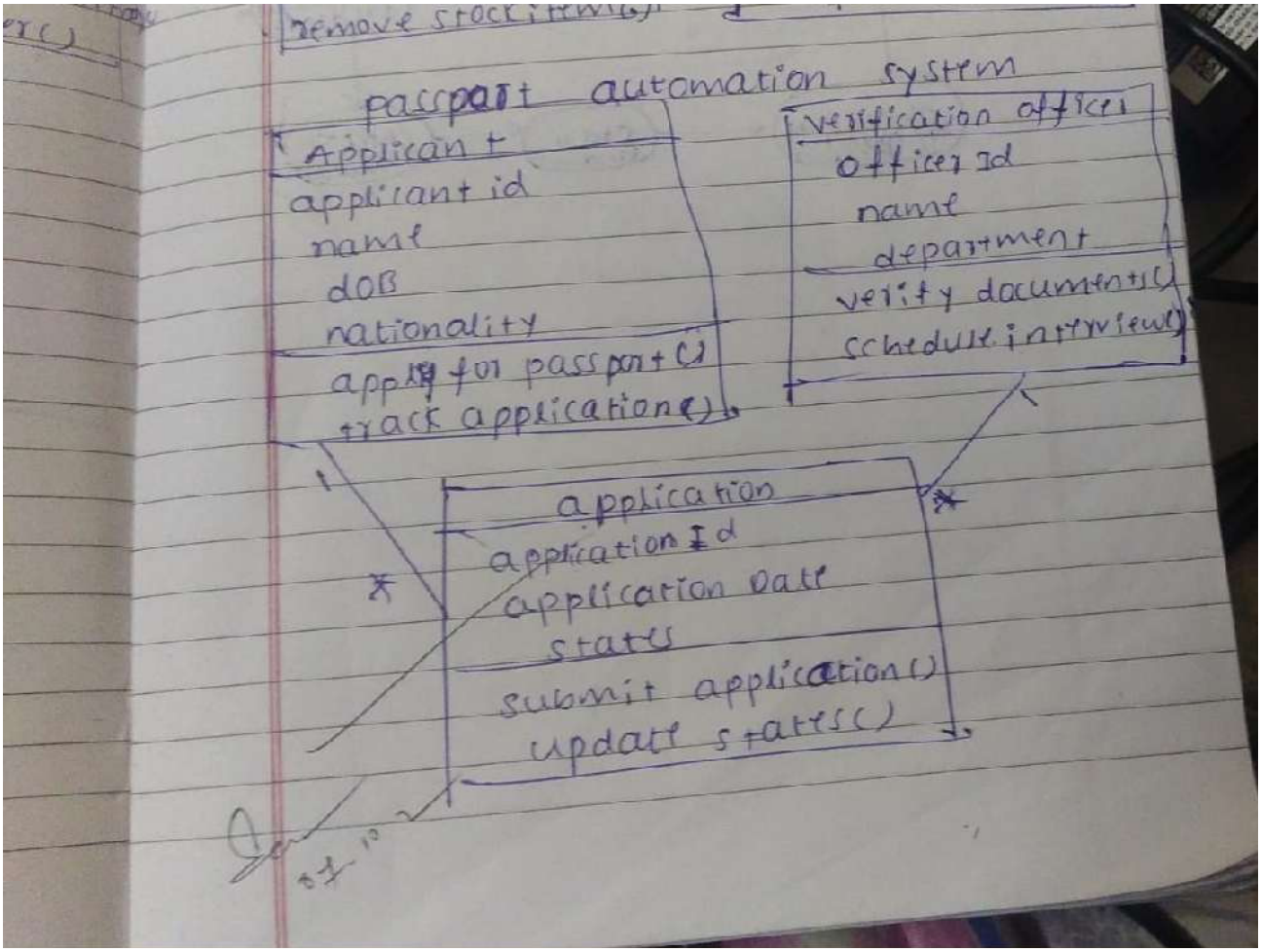
- Applicant has attributes like ID, name, address, date of birth, email, and application tracking status. They can submit applications and track their status.
- Application has an ID, status, and information about who submitted it. It can be created and processed.
- VerificationOfficer has an ID, name, and role. They can verify documents and approve applications.
- Passport has an ID, issue date, expiry date, and information about who it was issued for. It can be renewed.
- Appointment has an ID, date, time, and status. It can be scheduled and rescheduled.
- FeePayment has an ID, amount, and payment status (pending, confirmed, or failed). It can be processed and confirmed.
- Notification has an ID, message, and status. It can be sent and tracked.

Relationships:

- An Applicant can submit multiple Applications.

- A VerificationOfficer can verify multiple Applications.
- An Application can generate multiple Notifications.
- An Application can be associated with one Appointment.
- An Application has one associated FeePayment.
- General Passport is a parent class with Diplomatic Passport and Official Passport as subclasses.

Figure 5.5: Class Diagram Obs bk



State Diagram

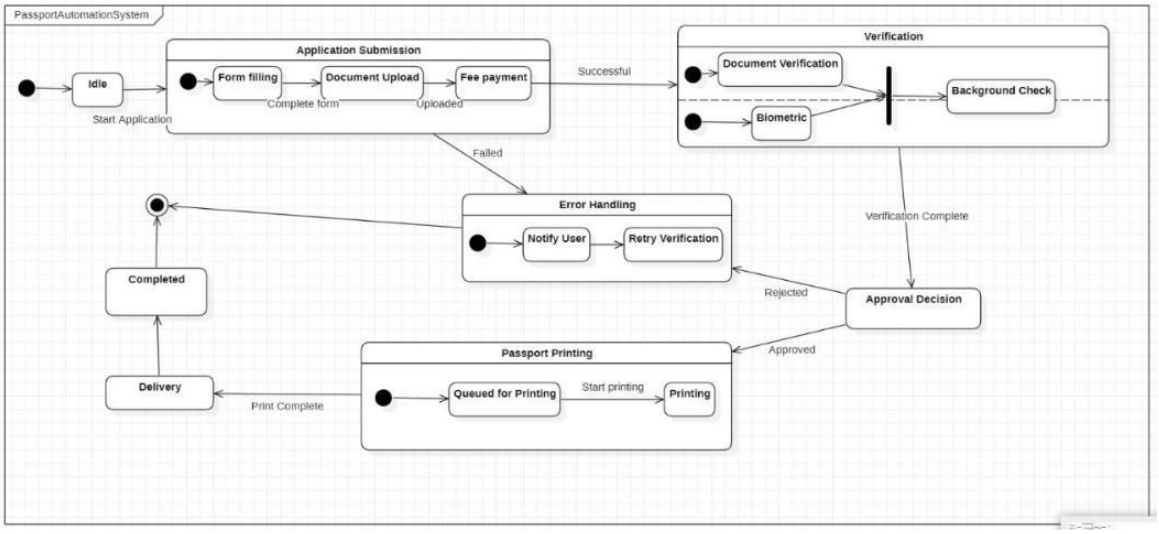


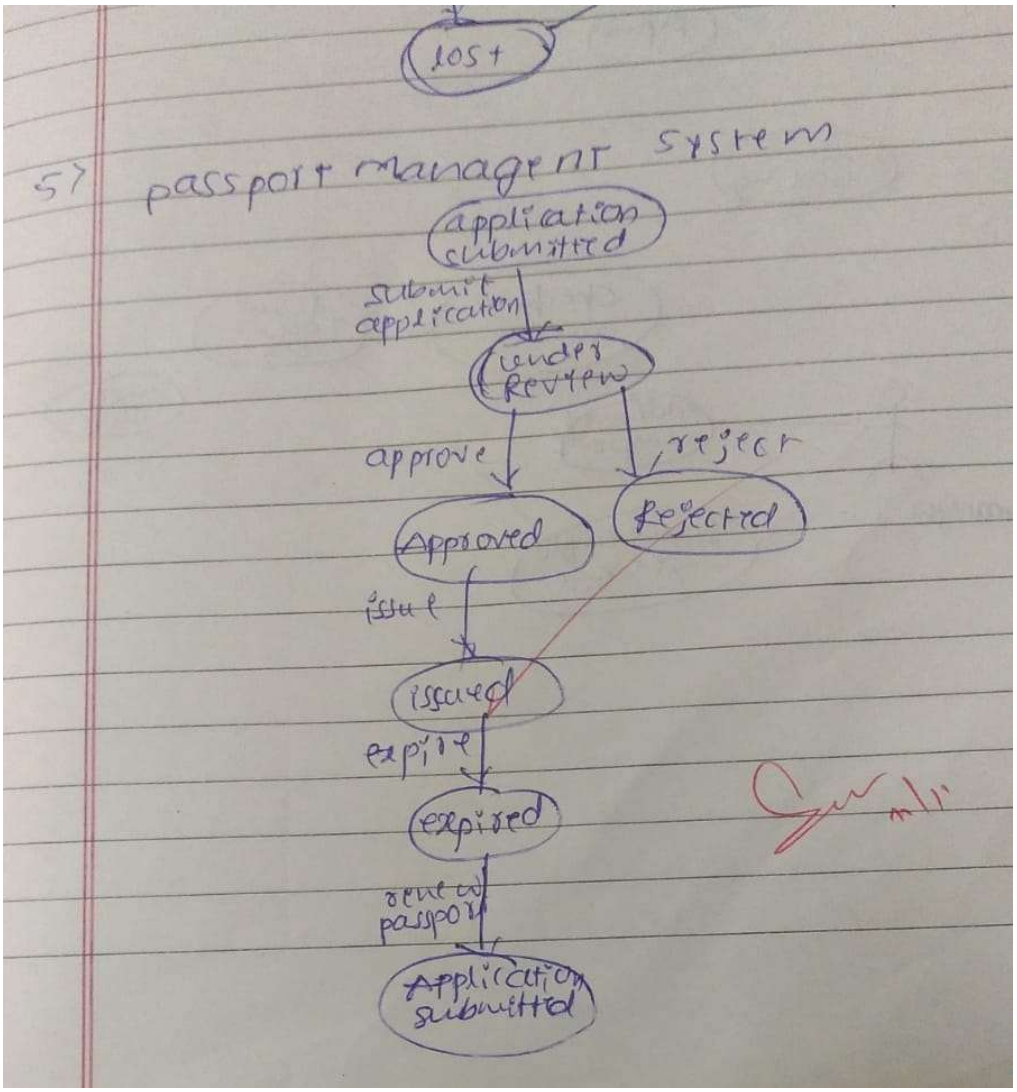
Figure 5.6: State Diagram

States:

- **Idle:** The initial state of the system.
- **Start Application:** The state where the application process begins.
- **Form Filling:** The state where the applicant is filling out the application form.
- **Document Upload:** The state where the applicant uploads supporting documents.
- **Fee Payment:** The state where the applicant pays the application fee.
- **Successful:** The state reached after successful submission of the application.
- **Document Verification:** The state where the submitted documents are being verified.
- **Biometric:** The state where biometric data (fingerprints, etc.) is being captured.
- **Background Check:** The state where the background check is being conducted.
- **Verification Complete:** The state reached after the verification process is complete.
- **Rejected:** The state indicating that the application has been rejected.
- **Approved:** The state indicating that the application has been approved.
- **Approval Decision:** The state where the decision to approve or reject the application is made.

- **Queued for Printing:** The state where the passport is queued for printing.
- **Start Printing:** The state where the passport printing process begins.
- **Printing:** The state where the passport is being printed.
- **Print Complete:** The state reached after the passport is printed.
- **Delivery:** The state where the passport is being delivered to the applicant.
- **Completed:** The final state indicating that the entire passport application process is complete.
- **Error Handling:** The state where error conditions are handled, such as failed payments or incomplete applications.

Figure 5.7: State Diagram Obs bk



Use - Case Diagram

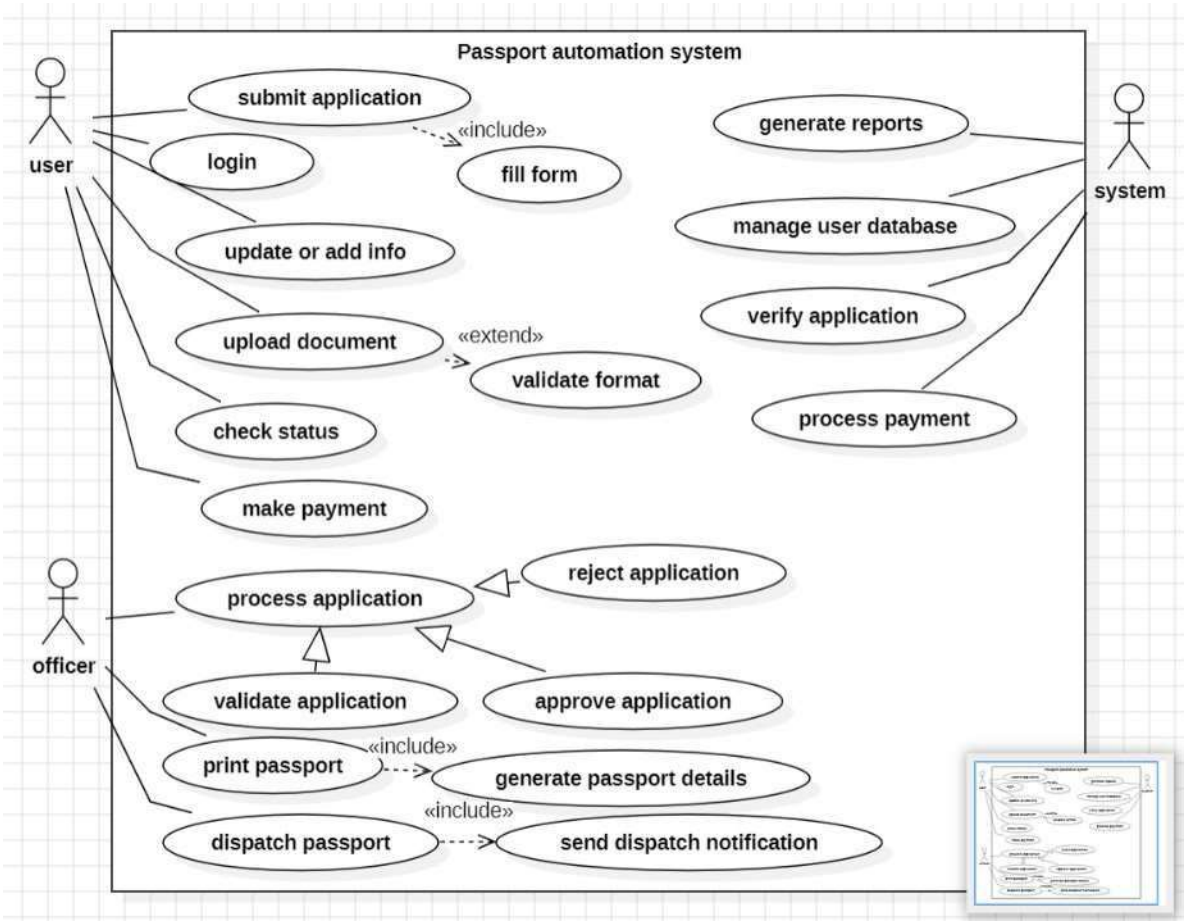


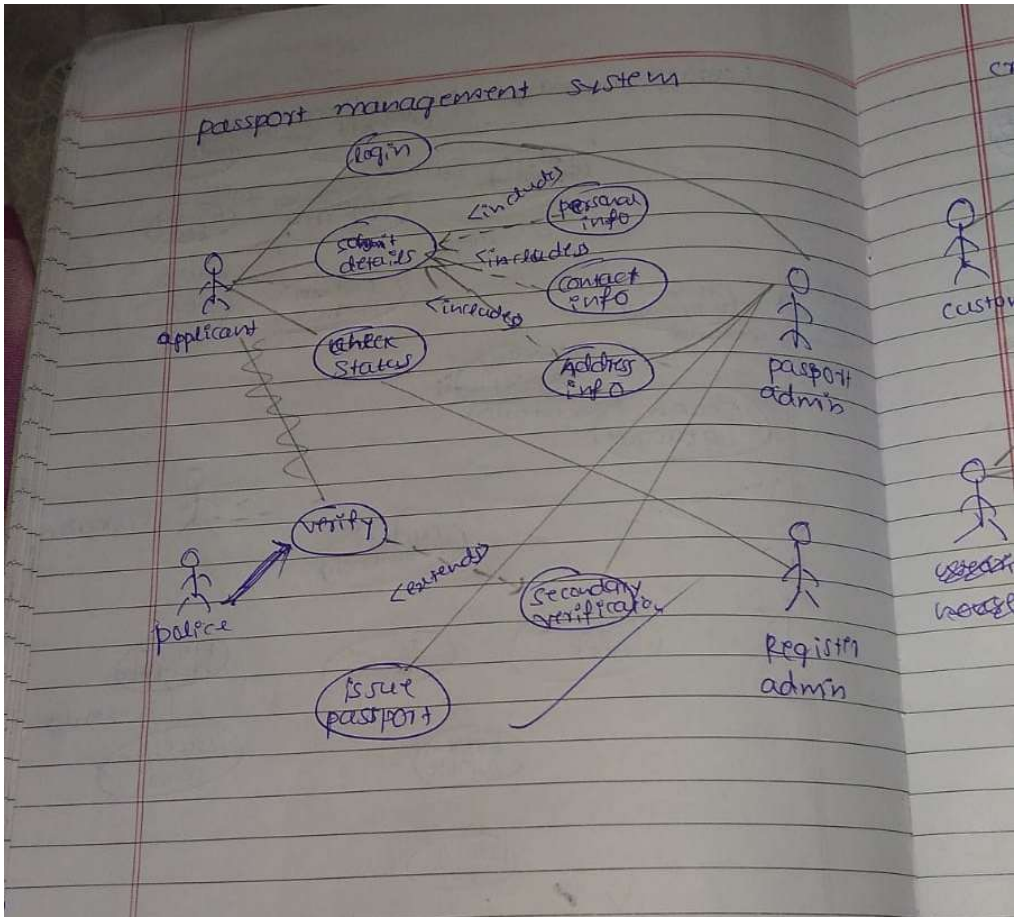
Figure 5.8: Use Case Diagram

Use Cases:

- Login: The process of authenticating a user to access the system.
- Submit Application: The process of submitting a passport application.
- Fill Form: The process of filling out the application form.
- Upload Document: The process of uploading supporting documents for the application.
- Check Status: Allows users to check the status of their application.
- Make Payment: The process of making the required payment for the application.
- Process Application: The process of reviewing and processing the application.
- Validate Application: The process of verifying the information and documents provided in the application.

- Approve Application: The process of approving an application.
- Reject Application: The process of rejecting an application.
- Print Passport: The process of printing the passport.
- Generate Passport Details: The process of generating the passport details based on the approved application.
- Dispatch Passport: The process of dispatching the passport to the applicant.
- Send Dispatch Notification: The process of sending a notification to the applicant regarding passport dispatch.
- Manage User Database: Includes tasks like adding, updating, and deleting user accounts.
- Generate Reports: Generates reports on various aspects of the passport application process.
- Verify Application: The process of verifying the authenticity of the application and supporting documents.

Figure 5.9: Use Case Diagram Obs bk


passport management system
login
<include>
personal info
submit details
<include>
contact info
<include>
applicant
check status
Address info
passport admin
verify
police
<extend>
Secondary verification
Registrn admin
issue passport

Sequence Diagram

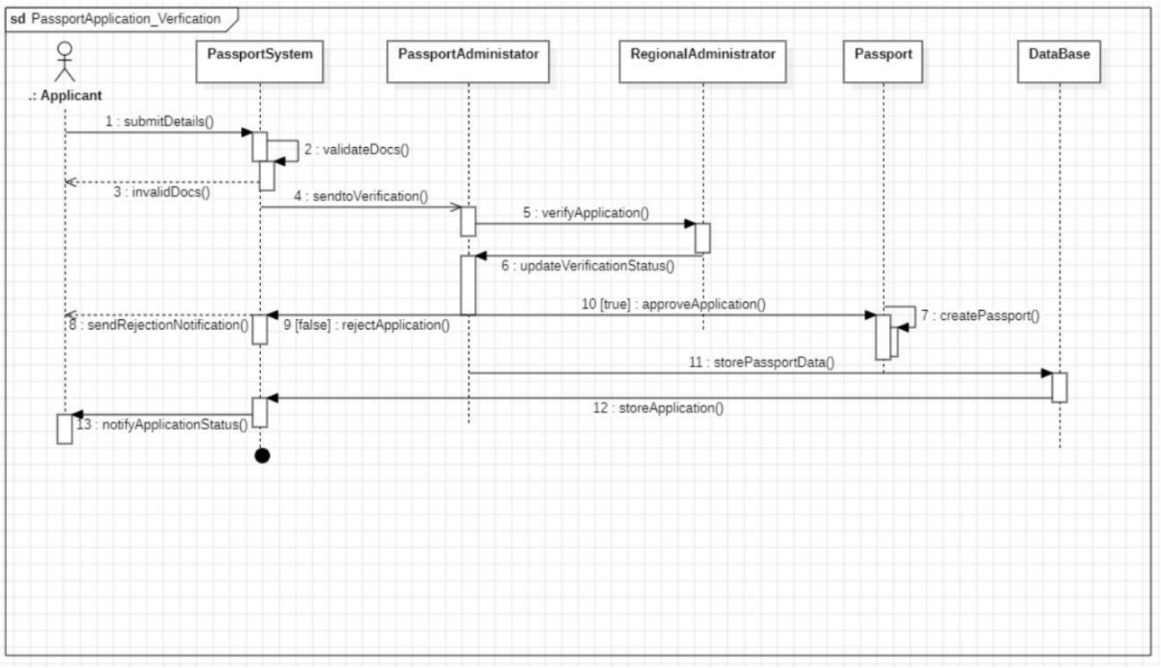
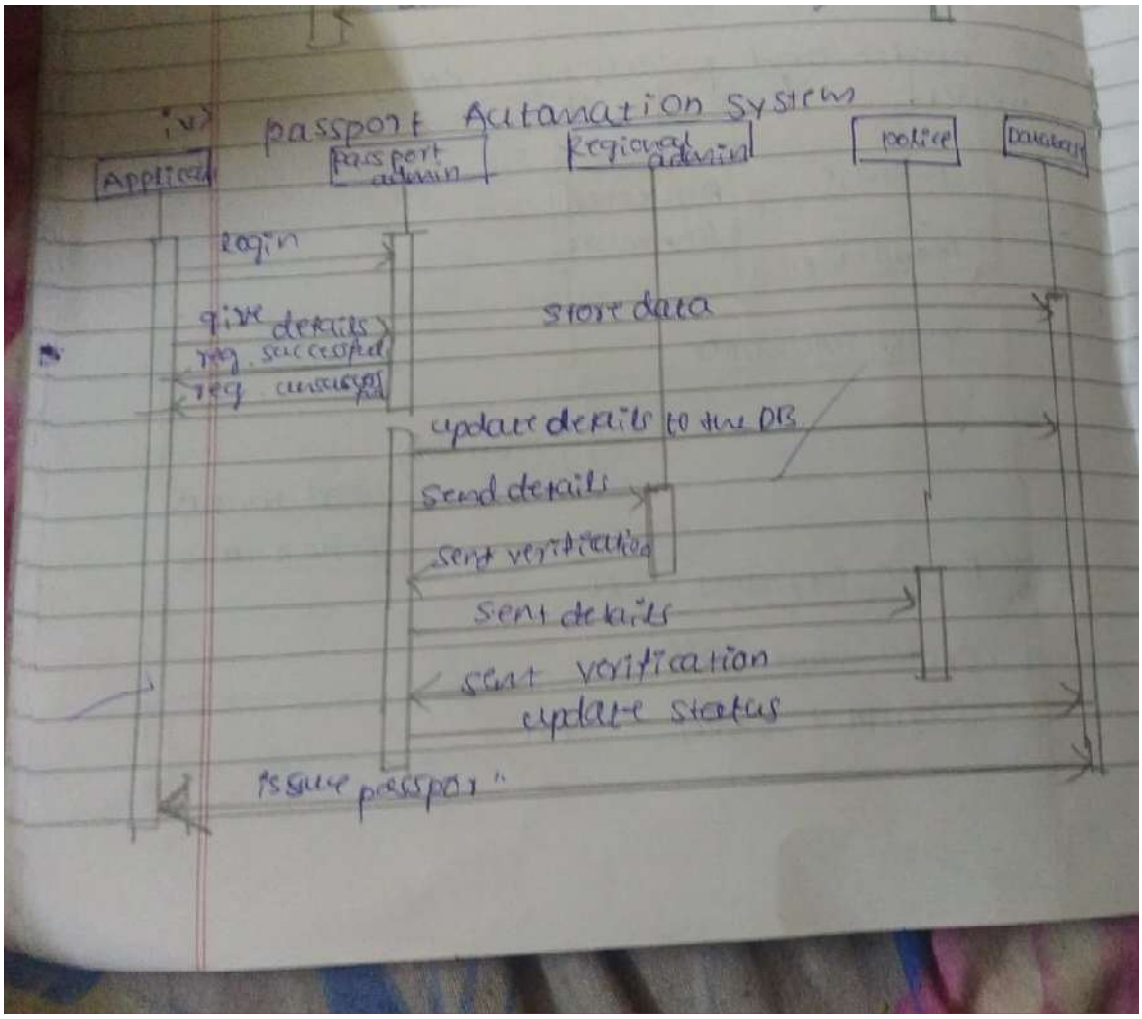


Figure 5.10: Sequence Diagram

1. submitDetails(): The Applicant initiates the process by submitting their application details to the PassportSystem.
2. validateDocs(): The PassportSystem validates the submitted documents.
3. invalidDocs(): If the documents are invalid, the PassportSystem sends an error message to the Applicant.
4. sendToVerification(): If the documents are valid, the PassportSystem sends the application to the PassportAdministrator for verification.
5. verifyApplication(): The PassportAdministrator verifies the application details.
6. updateVerificationStatus(): The PassportSystem updates the verification status of the application.
7. rejectApplication(): If the application is rejected, the PassportSystem sends a rejection notification to the Applicant.
8. approveApplication(): If the application is approved, the PassportSystem sends it to the RegionalAdministrator for further processing.
9. createPassport(): The PassportSystem creates a new Passport object.
10. storePassportData(): The PassportSystem stores the passport data in the Database.
11. storeApplication(): The PassportSystem stores the application details in the Database.

12. notifyApplicationStatus(): The PassportSystem notifies the Applicant about the status of their application.

Figure 5.11: Sequence Diagram Obs bk



Activity Diagram

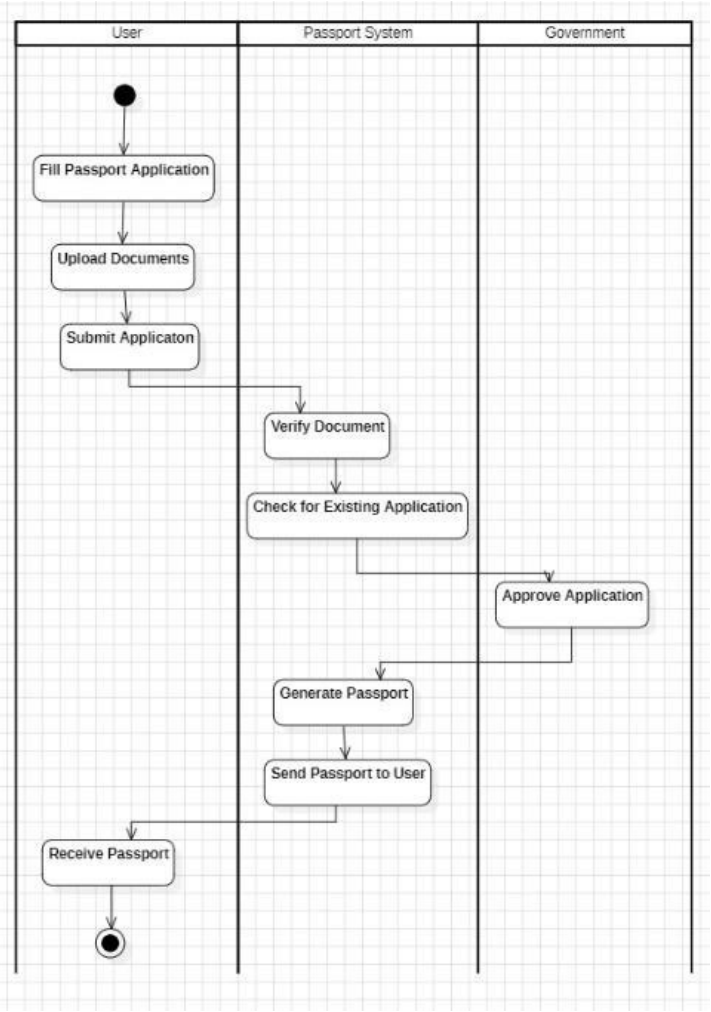
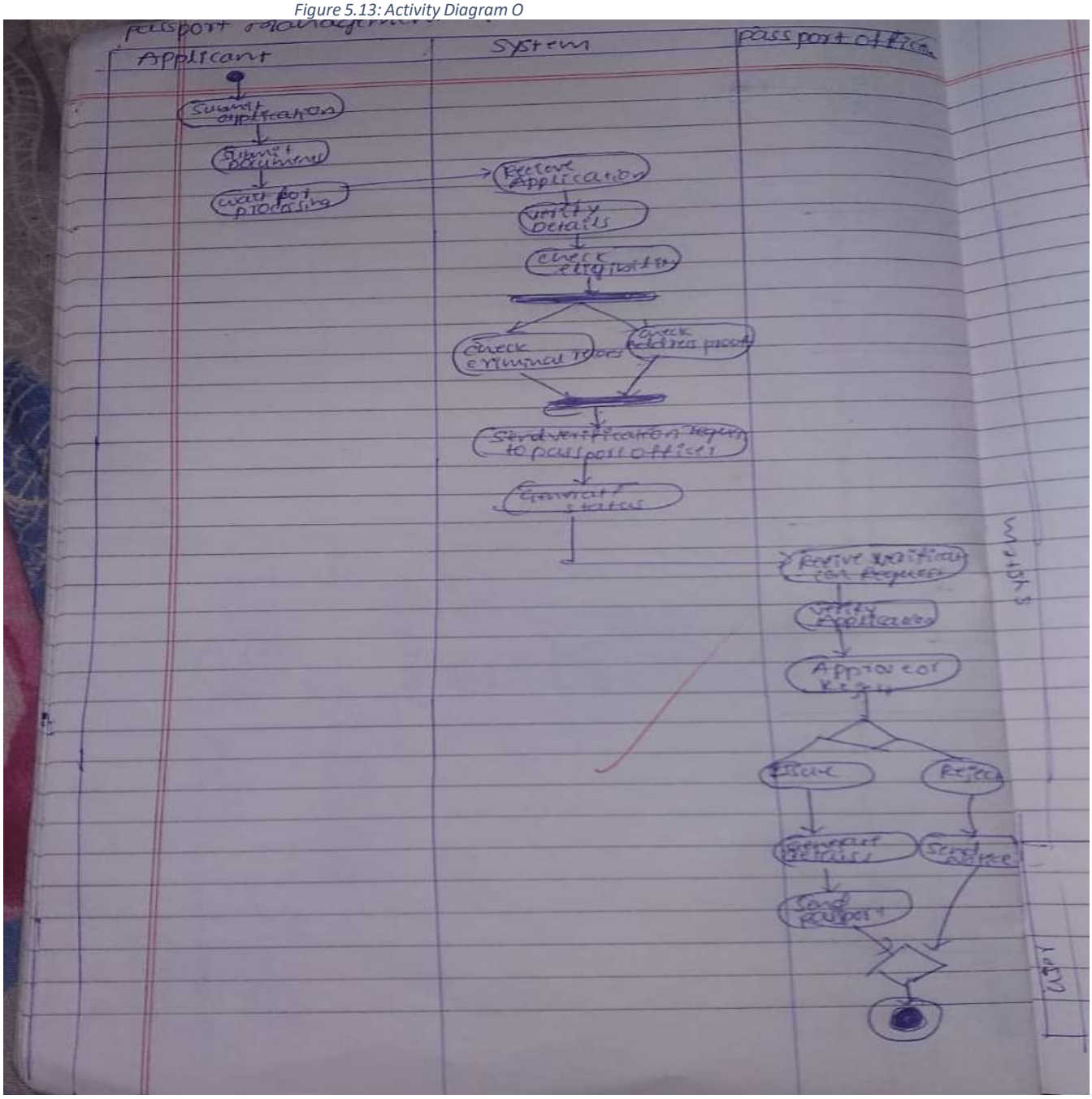


Figure 5.12: Activity Diagram

1. **Fill Passport Application:** The user starts by filling out the online passport application form.
2. **Upload Documents:** The user uploads the required supporting documents, such as proof of identity and address.
3. **Submit Application:** The user submits the completed application to the Passport System.
4. **Verify Document:** The Passport System verifies the authenticity of the uploaded documents.
5. **Check for Existing Application:** The system checks if the user already has an existing passport application.
6. **Approve Application:** If the documents are valid and no existing application is found, the Government authorities approve the application.

7. **Generate Passport:** The Passport System generates the passport.
8. **Send Passport to User:** The passport is sent to the user.
9. **Receive Passport:** The user receives the passport.

Figure 5.13: Activity Diagram O



bs bk