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:

MADHU SARIKA

1BM22CS140

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 MADHU SARIKA (1BM22CS140) during the 5th Semester Sep 24- Jan2025.

Signature of the Faculty Incharge:

SANDHYA A KULKARNI

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

TABLE OF CONTENTS

HOTEL MANAGEMENT SYSTEM

Problem Statement:

This aims to develop a Hotel Management System to streamline operations and enhance guest experience. The system should encompass the following functionalities:

- Reservation Management: Enable online booking, real-time availability checks, secure payments, and automated confirmations.
- Guest Services: Facilitate smooth check-in/out, handle guest requests, and provide information on hotel amenities.
- Inventory Management: Track room availability, manage room rates, and handle room assignments and maintenance.
- Financial Management: Generate invoices, process payments, manage accounts, and generate financial reports.
- Employee Management: Manage employee schedules, track attendance, and control access permissions.

classmate
Date 23/1/24
Page

Software Requirement Specification (SRS)

1. Introduction

1.1 Purpose of this Document

The SRS document outlines the detailed requirements for Hotel Management System. It serves as blue print for the development team, ensuring final product meets the specified specification and fulfills the customer requirements.

1.2 Scope of this document

It aims to provide clear specifications for the system's development, including overview of costs and timelines involved. This HMS (Hotel Management System) is designed to streamline hotel operations and enhance customer experience.

1.3 Overview

The HMS will provide solution for managing hotel operations including reservations, customer management, billings and reportings. This will enable staff to handle daily tasks and offer user friendly interface for bookings.

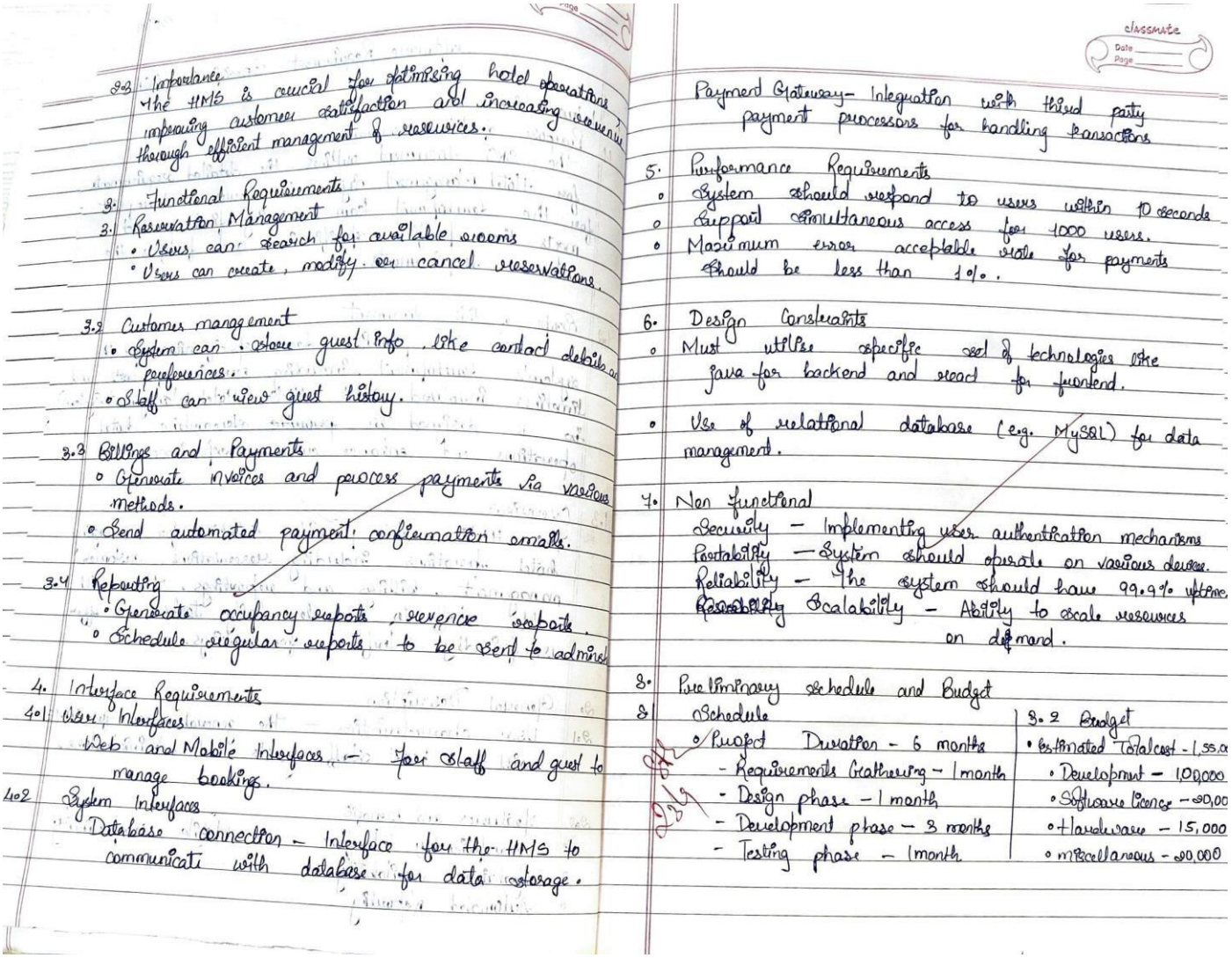
2. General Description

2.1 User characteristics — The general user for HMS will be hotel staff, guests and administrators.

2.2 Features and Benefits

- User-friendly Interface
- Mobile Compatibility
- Real time availability
- Automated Reporting

Fig 1.1



2.3 Importance
The HMS is crucial for optimising hotel operations, improving customer satisfaction and increasing revenue through efficient management of resources.

3. Functional Requirements
3.1 Reservation Management
- Users can search for available rooms
- Users can create, modify or cancel reservations.

3.2 Customer management
- System can store guest info like contact details and preferences.
- Staff can view guest history.

3.3 Billings and Payments
- Generate invoices and process payments via various methods.
- Send automated payment confirmation emails.

3.4 Reporting
- Generate occupancy reports, revenue reports.
- Schedule regular reports to be sent to admins.

4. Interface Requirements
4.1 User Interfaces
Web and Mobile Interfaces - for staff and guest to manage bookings.
4.2 System Interfaces
Database connection - Interface for the HMS to communicate with database for data storage.
Payment Gateways - Integration with third party payment processors for handling transactions

5. Performance Requirements
- System should respond to users within 10 seconds
- Support simultaneous access for 1000 users.
- Maximum error acceptable rate for payments should be less than 1%.

6. Design Constraints
- Must utilise specific set of technologies like java for backend and react for frontend.
- Use of relational database (eg. MySQL) for data management.

7. Non functional
Security - Implementing user authentication mechanisms
Portability - System should operate on various devices.
Reliability - The system should have 99.9% uptime.
Scalability - Ability to scale resources on demand.

8. Preliminary schedule and Budget
8.1 Schedule
- Project Duration - 6 months
  - Requirements Gathering - 1 month
  - Design phase - 1 month
  - Development phase - 3 months
  - Testing phase - 1 month

8.2 Budget
- Estimated Total cost - 1,55,0
- Development - 1,00,000
- Software Licence - 30,00
- Hardware - 15,000
- Miscellaneous - 10,000

Fig 1.2

UML DIAGRAMS

CLASS DIAGRAM

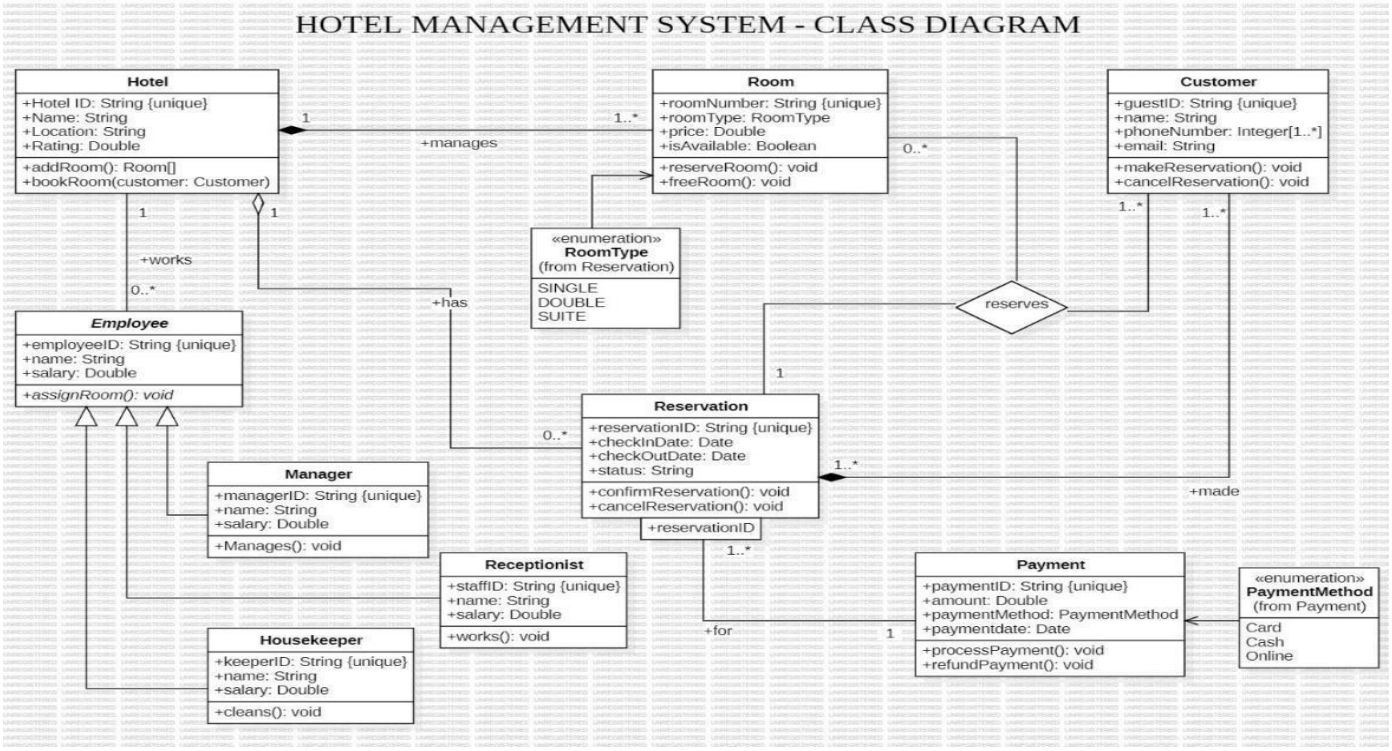


Fig 1.3

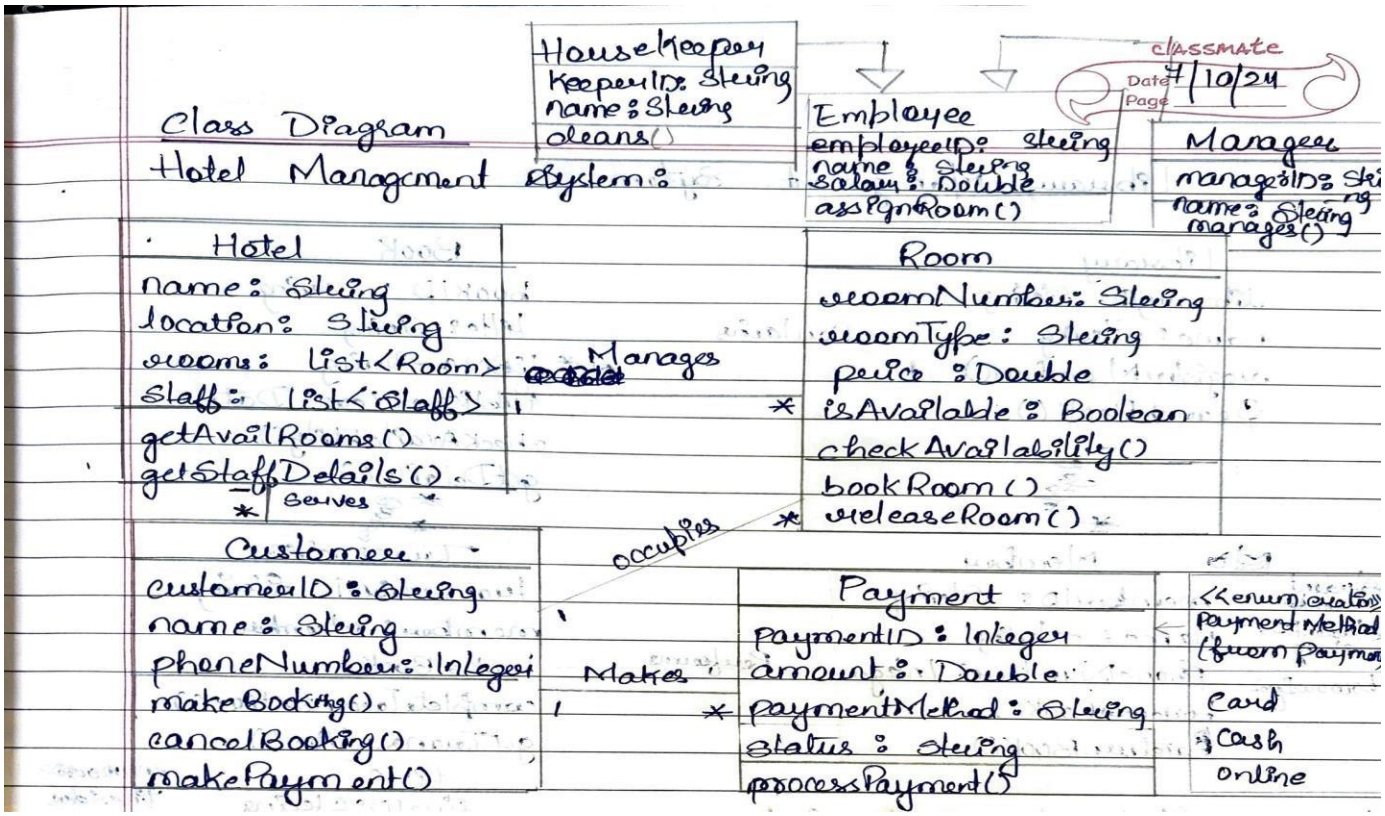


Fig 1.4

Brief Description :

1. **Hotel**: Represents a hotel, managing multiple rooms and employees. It allows room booking and addition.
2. **Room**: Represents rooms in a hotel with attributes like room type, price, and availability. Rooms can be reserved or freed.
3. **Reservation**: Manages bookings with details like check-in, check-out, and status. Customers can confirm or cancel reservations.
4. **Customer**: Represents guests, allowing them to make and cancel reservations. Includes personal details like name and contact info.
5. **Employee**: Generalised class for hotel staff with attributes like ID, name, and salary. Specialised into:
    - **Manager**: Oversees hotel operations.
    - **Receptionist**: Handles customer bookings and interactions.
    - **Housekeeper**: Maintains room cleanliness.
6. **Payment**: Manages payment transactions, including methods (Card, Cash, Online), processing, and refunds.

Key Features:

- **Associations** define relationships between classes, such as a hotel managing rooms, customers making reservations, and employees working in the hotel.
- **Enumerations**: Includes Room Type (Single, Double, Suite) and Payment Method (Card, Cash, Online).

STATE DIAGRAM

SIMPLE STATE DIAGRAM

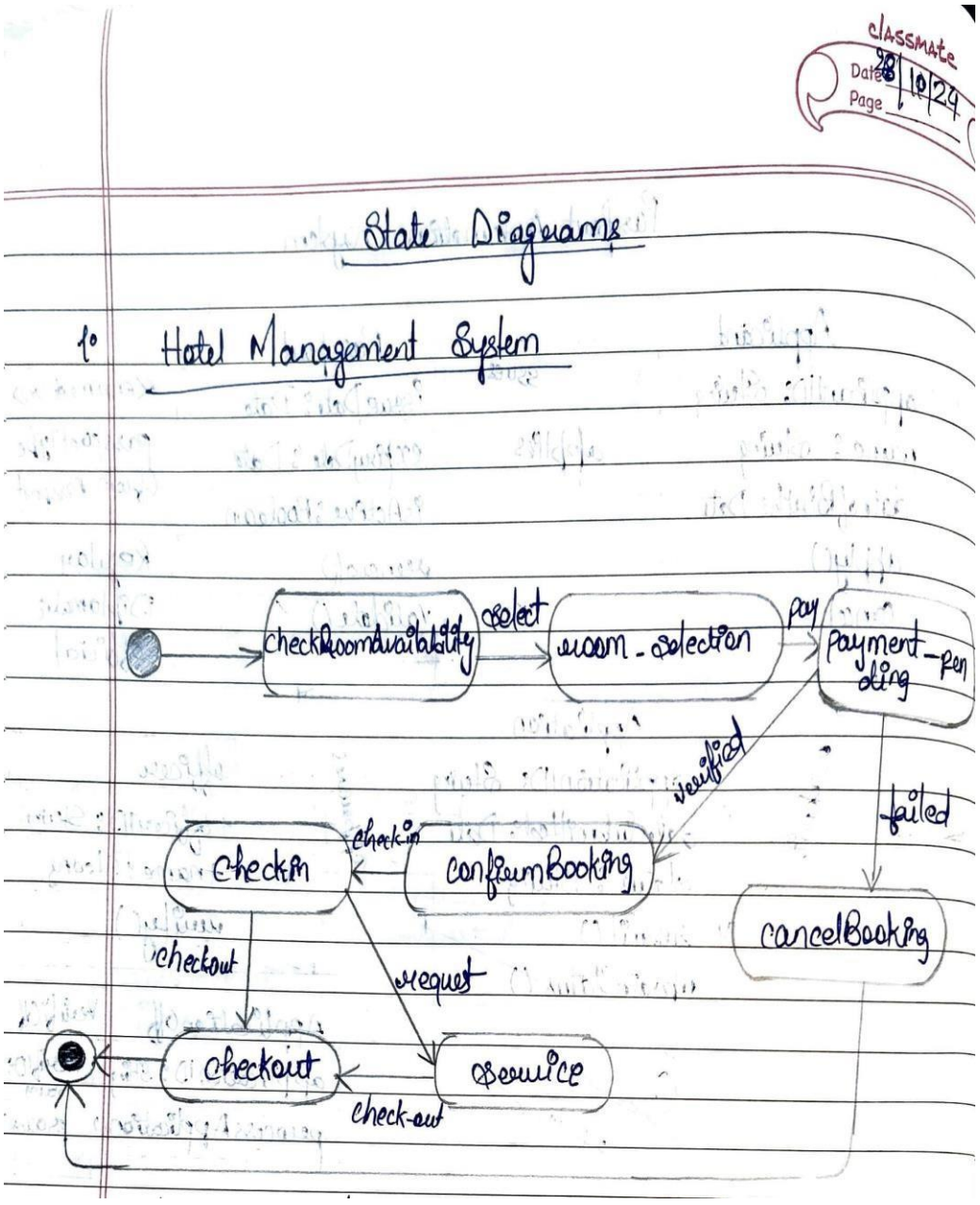


Fig 1.5

ADVANCED STATE DIAGRAM

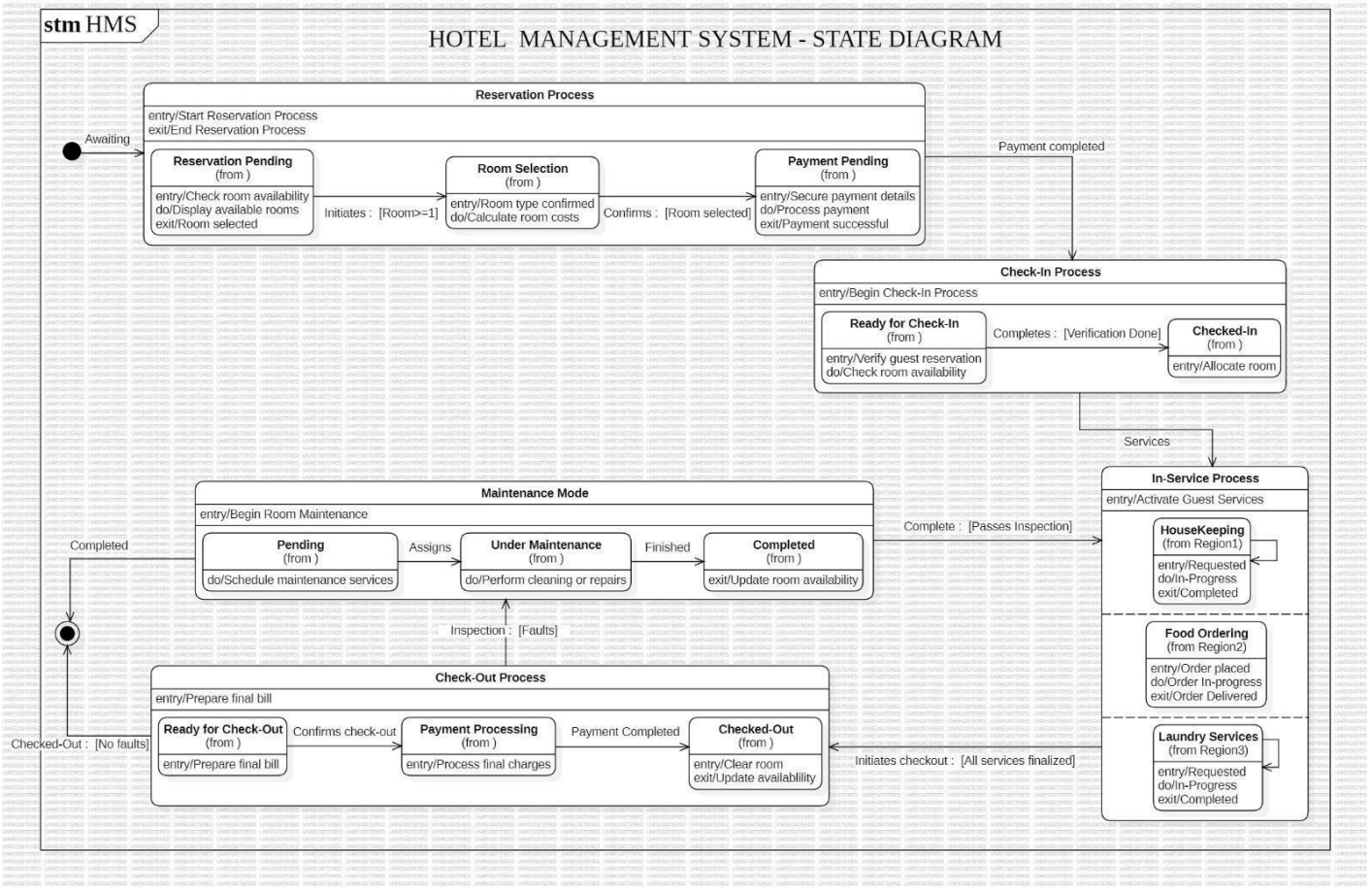


Fig 1.6

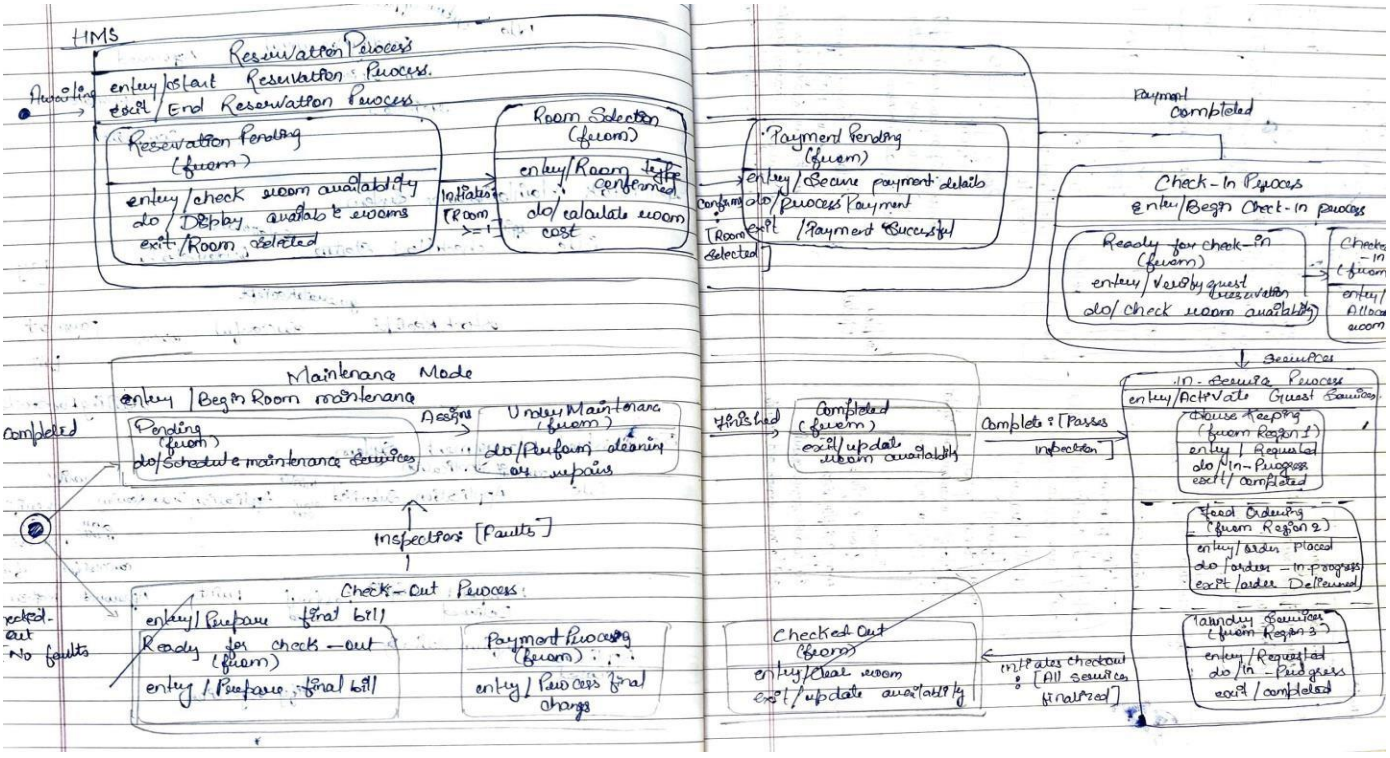


Fig 1.7

Brief Description:

1. **Reservation Process:**
    - Starts with "Awaiting Reservation."
    - Transitions to "Reservation Pending" after room availability is checked.
    - Moves to "Room Selection", where the guest chooses a room.
    - Proceeds to "Payment Pending" for payment processing.
    - Finally, reaches "Payment Completed" upon successful payment.
2. **Check-In Process:**
    - Begins with "Ready for Check-In."
    - Transitions to "Checked-In" after guest verification and room allocation.
3. **Maintenance Mode:**
    - Starts with "Pending" for scheduling maintenance services.
    - Moves to "Under Maintenance" for cleaning or repairs.
    - Transitions to "Finished" upon completion.
    - Finally, moves to "Completed" after inspection and room availability update.
4. **Check-Out Process:**
    - Starts with "Ready for Check-Out."
    - Transitions to "Checked-Out" after final bill preparation, payment processing, and room clearance.
5. **Services:**
    - Includes Housekeeping, Food Ordering, and Laundry Services.
    - Each service has its state transitions, such as "Requested," "In-Progress," and "Completed."

USE CASE DIAGRAM

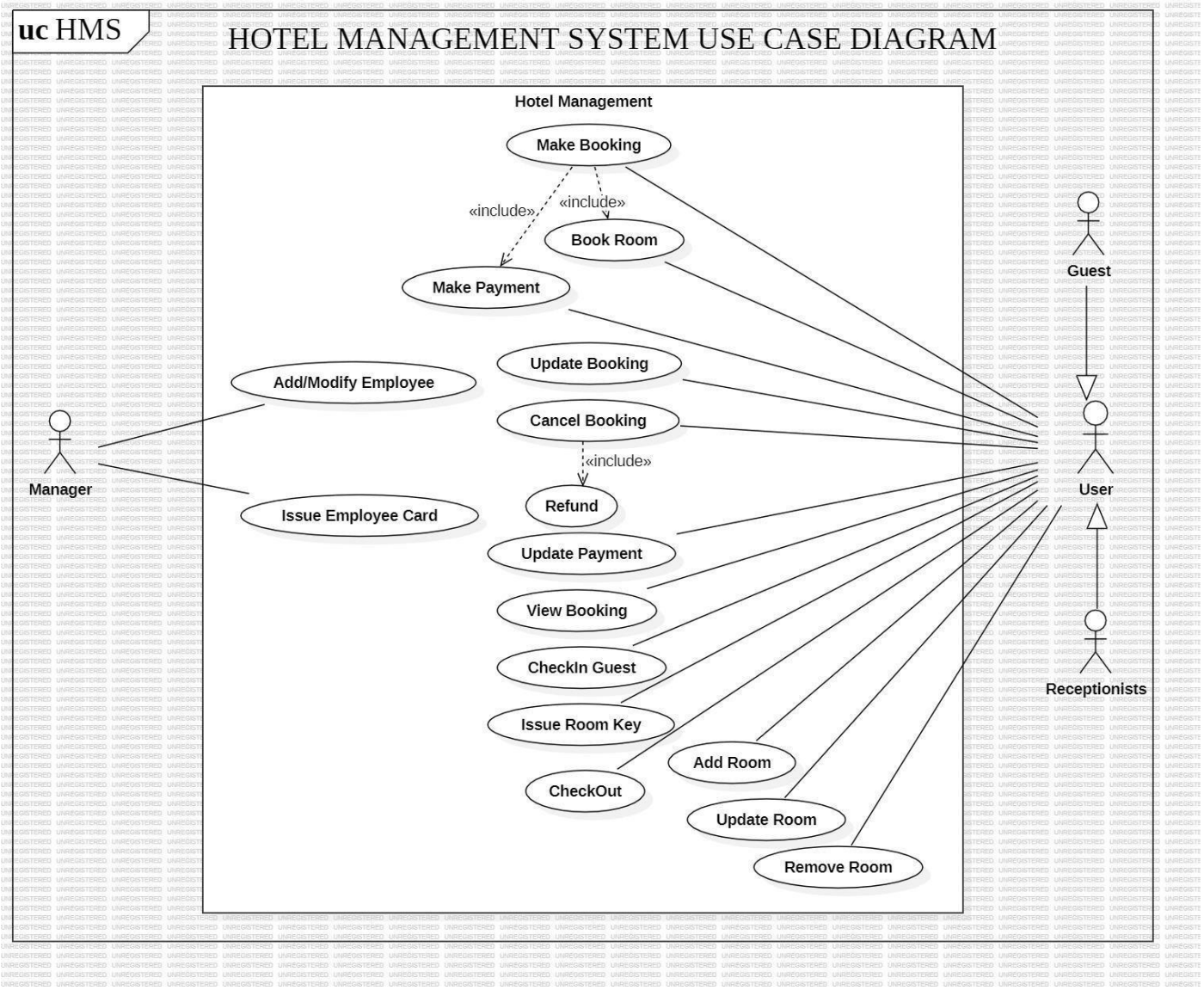


Fig 1.8

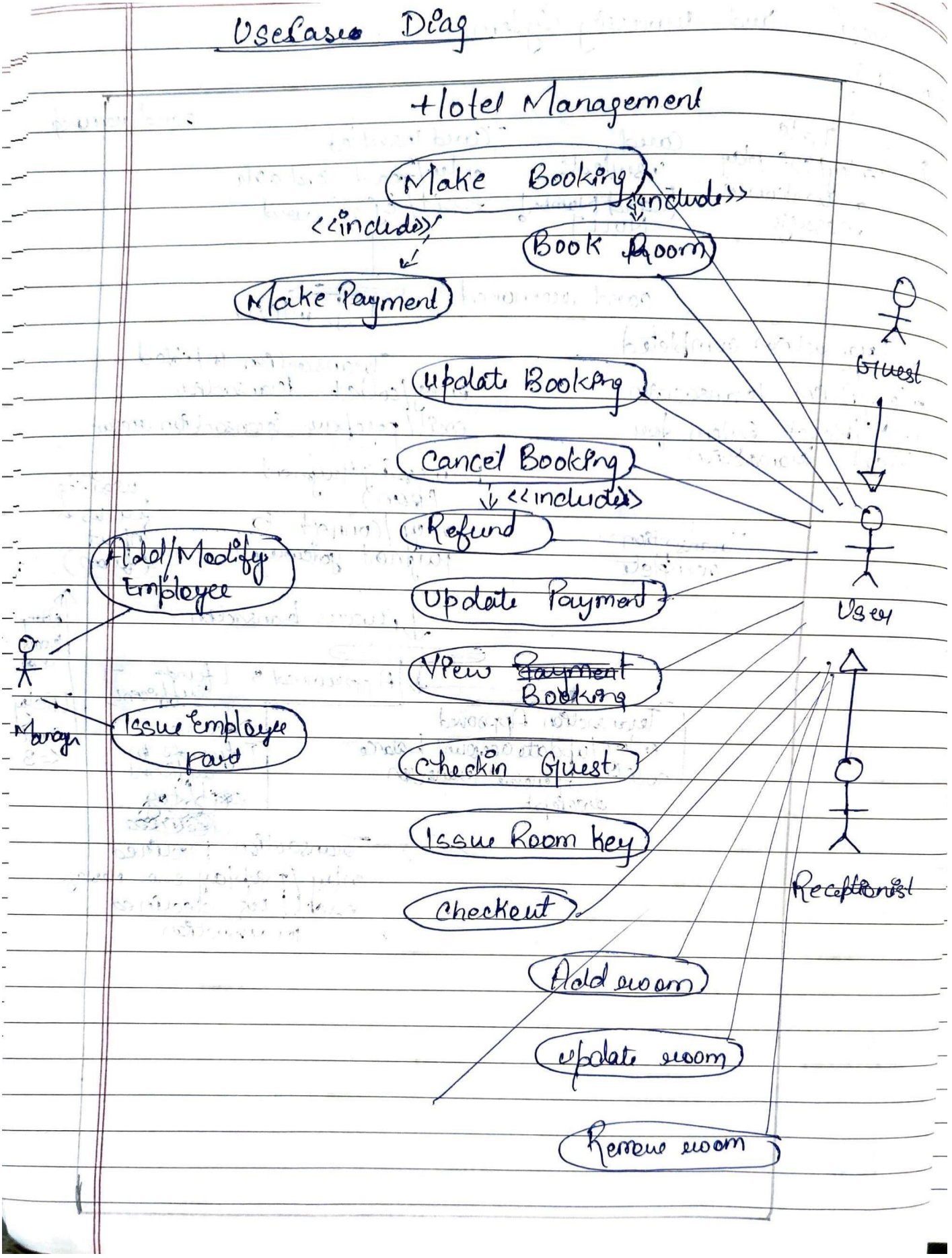

Usecase Diag
Hotel Management
Make Booking
<<include>>
Book Room
<<include>>
Make Payment
Update Booking
Cancel Booking
<<include>>
Refund
Add/Modify Employee
Update Payment
View Booking
Issue Employee Pay
Checkin Guest
Issue Room key
Checkout
Add room
update room
Remove room
Guest
User
Receptionist

Fig 1.9

Brief Description:

1. Hotel Management

This is the overarching use case, encompassing all other functionalities within the system.

2. Make Booking

Includes the sub-use cases of:

Book Room: Allows guests to select and reserve rooms.

Make Payment: Processes payment for the reservation.

3. Update Booking

Enables guests to modify or change their reservation details.

4. Cancel Booking

Includes the sub-use case of:

Refund: Processes refunds for cancelled bookings.

5. View Booking

Allows guests to view their reservation details and status.

6. Check-in Guest

Handles the process of checking guests into their rooms.

7. Issue Room Key

Provides room keys to checked-in guests.

8. Check-Out

Handles the process of checking guests out of their rooms.

9. Add Room

Allows the addition of new rooms to the hotel's inventory.

10. Update Room

Enables updates to room information, such as amenities or status.

11. Remove Room

Allows the removal of rooms from the hotel's inventory.

SEQUENCE DIAGRAM

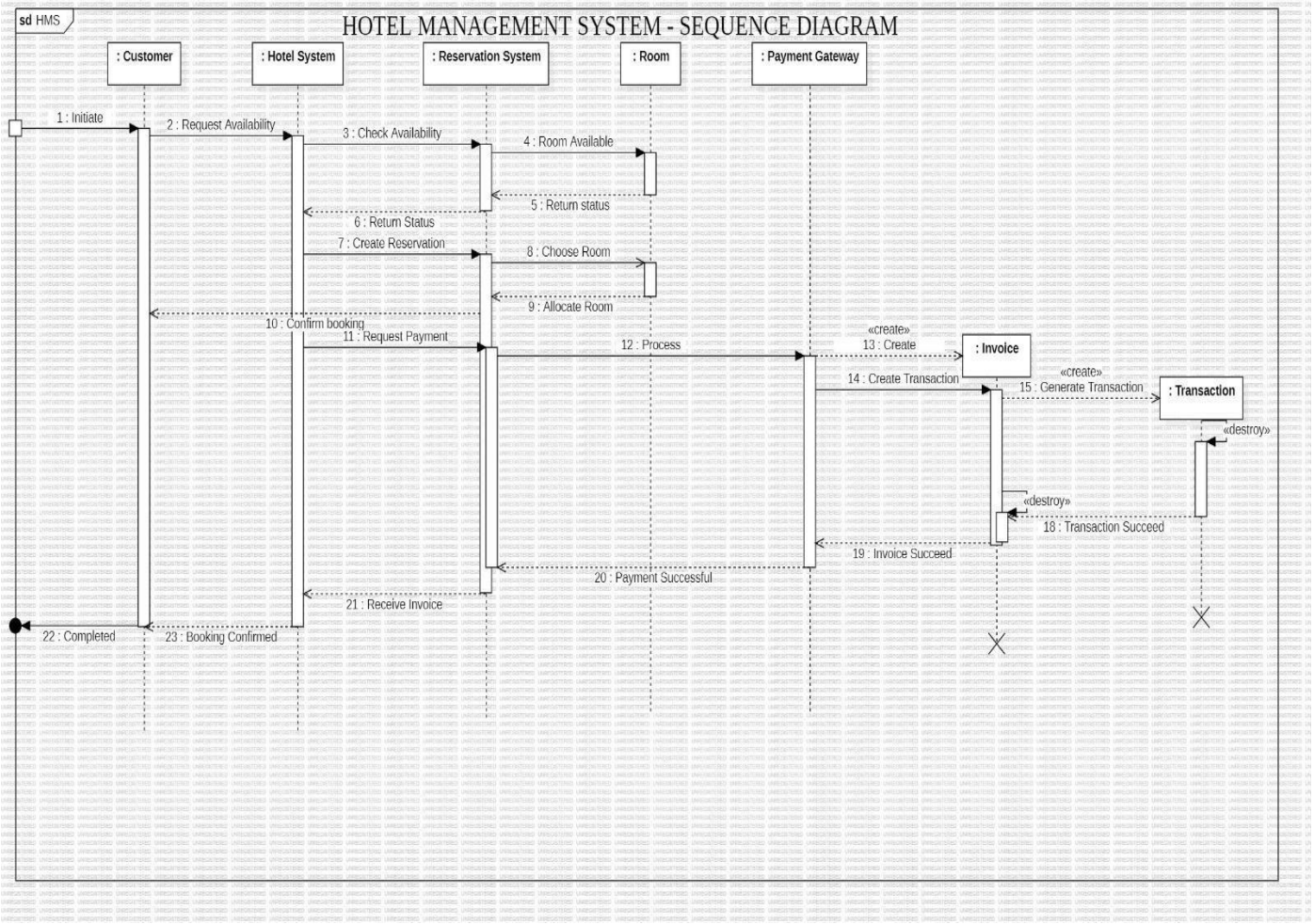


Fig 1.10

Sequence Diagram

Hotel Management System :

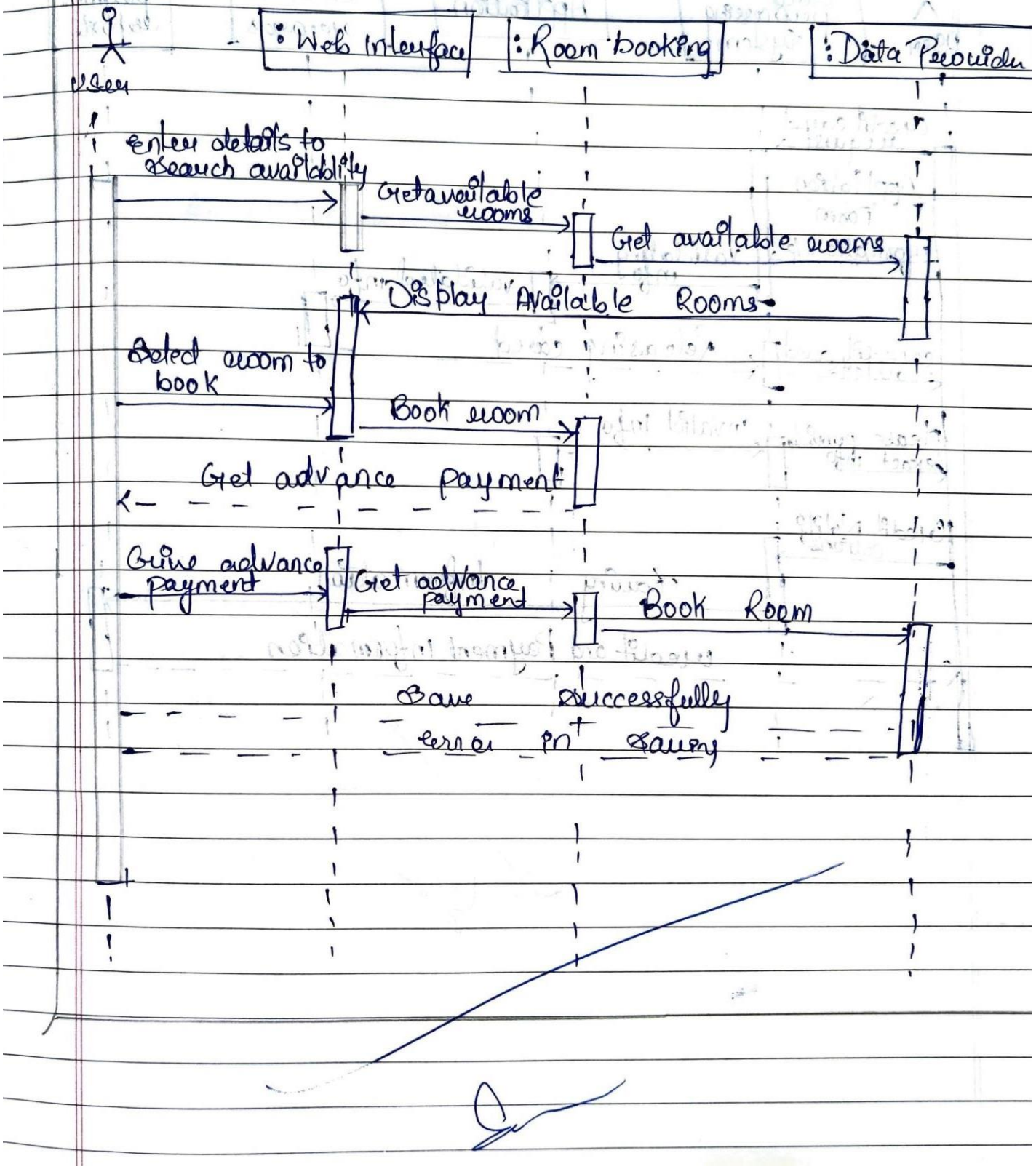


Fig 1.11

Brief Description:

1. **Customer Initiates Reservation:** The customer starts the process by initiating a reservation request.
2. **Check Availability:** The customer requests availability information, and the hotel system checks for available rooms.
3. **Room Available:** The hotel system returns the available rooms to the customer.
4. **Choose Room:** The customer selects a preferred room.
5. **Create Reservation:** The hotel system creates a reservation record.
6. **Request Payment:** The hotel system requests payment from the customer.
7. **Process Payment:** The payment gateway processes the payment.
8. **Transaction Succeed:** The payment gateway confirms a successful transaction.
9. **Booking Confirmed:** The hotel system confirms the booking to the customer.

ACTIVITY DIAGRAM

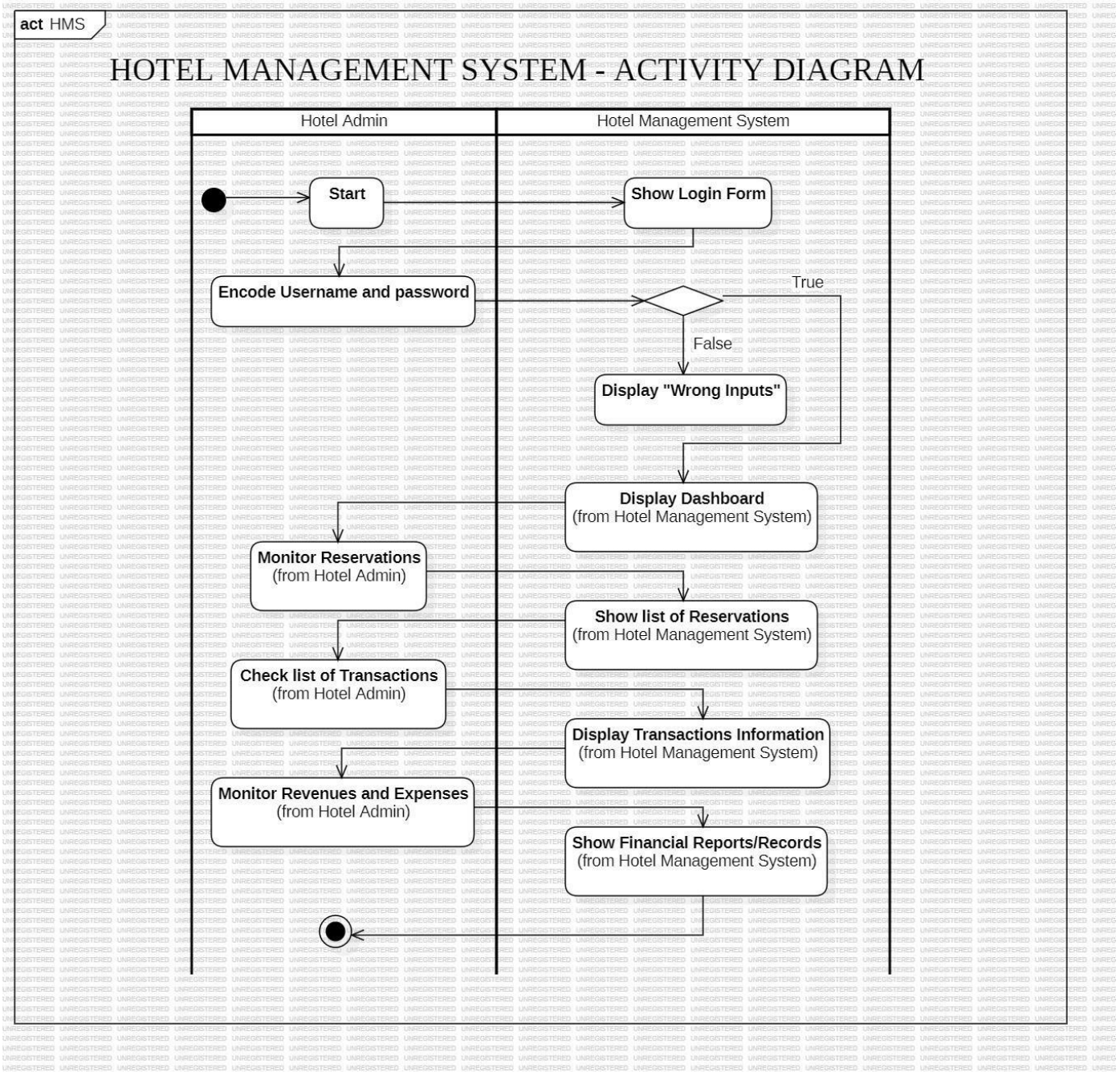


Fig 1.12

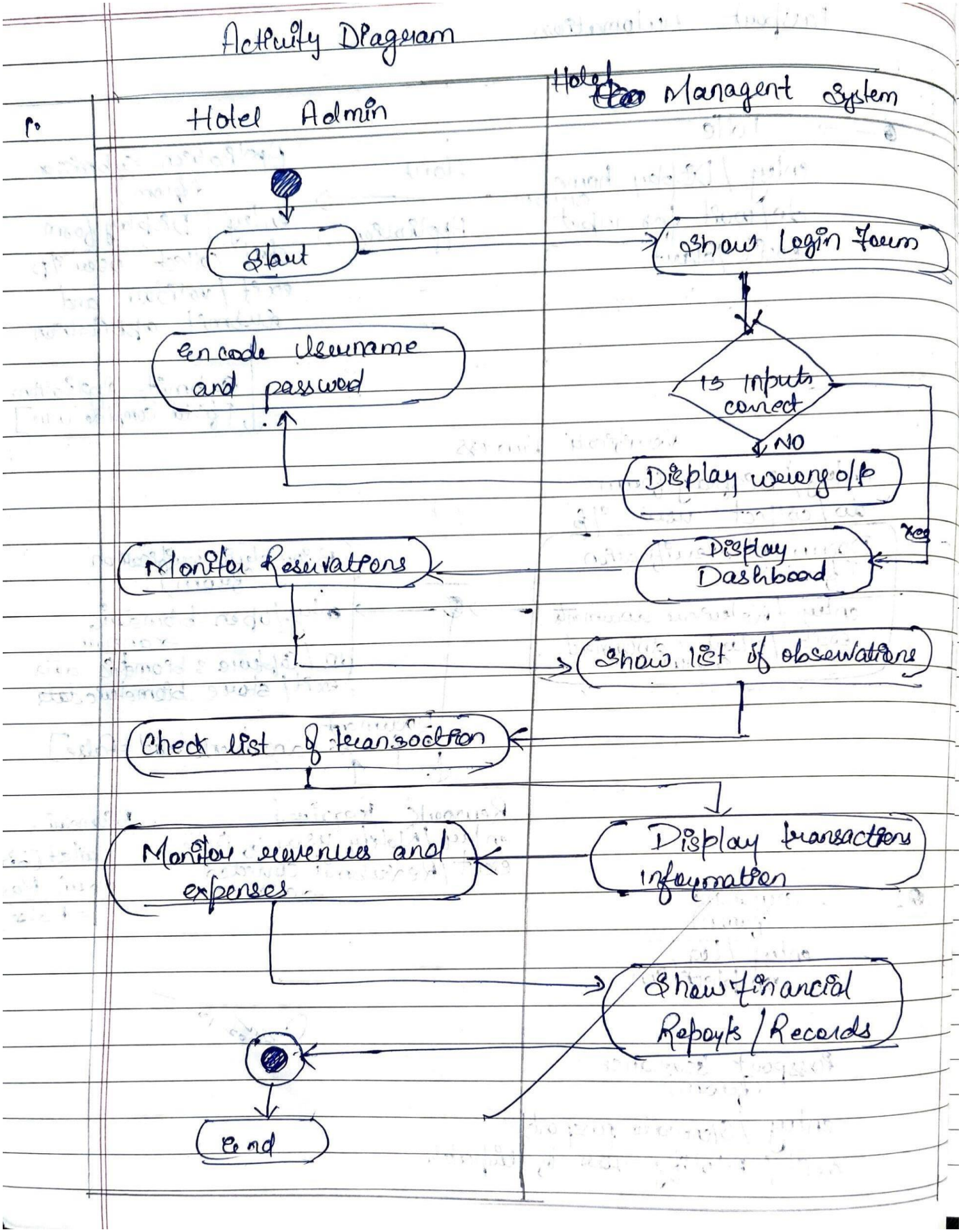

Activity Diagram
Hotel Admin
Hotel Managent System
Start
Encode Username and password
Show Login Form
Is Inputs corect
NO
Display wrong o/p
Display Dashboad
Monitor Reservations
Show list of observations
Check list of transaction
Monitor revenues and expenses
Display transactions information
Show financial Repoyts / Records
End

Fig 1.13

Brief Description:

1. **Start:** The process begins with the administrator starting the system.
2. **Show Login Form:** The system displays a login form to the administrator.
3. **Encode Username and Password:** The administrator enters their username and password.
4. **Login Validation:** The system validates the entered credentials. If they are correct:
    - **True:** The administrator is granted access to the dashboard.
    - **False:** An error message ("Wrong Inputs") is displayed, and the process returns to the login form.
5. **Display Dashboard:** Once logged in, the administrator is presented with the main dashboard.
6. **Monitor Reservations:** The administrator can view a list of current reservations.
7. **Check List of Transactions:** The administrator can access and review transaction records.
8. **Monitor Revenues and Expenses:** The administrator can view financial reports and records.

CREDIT CARD PROCESSING SYSTEM

Problem Statement:

Develop a robust and secure Credit Card Processing System to facilitate electronic business transactions. The system should:

- Process transactions securely: Ensure the confidentiality and integrity of cardholder data using industry-standard encryption and security protocols.
- Support various payment methods: Accept major credit cards (Visa, Mastercard, Amex, etc.), debit cards, and other electronic payment options.
- Integrate seamlessly with merchant systems: Allow easy integration with point-of-sale (POS) systems, e-commerce platforms, and other business software.
- Provide real-time transaction processing: Enable quick and efficient transaction authorisation and settlement.
- Generate comprehensive reports: Provide detailed business transaction reports, including sales analysis, fraud detection, and reconciliation.
- Ensure regulatory compliance: Adhere to all relevant payment card industry (PCI DSS) and other regulatory standards.

SOFTWARE REQUIREMENTS SPECIFICATION

30/9/24

Credit card Processing System

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to define the software requirements for the development of credit card processing system. This system will facilitate secure, efficient and reliable processing of credit card transactions.

1.2 Scope of the Document

This document covers the software specifications for a system that handles credit card authorization, transaction processing. It will be used by developers, testers and stakeholders to ensure the system meets both business and technical objectives.

1.3 Overview

This processing system will allow users to authorize, process and settle payments using credit cards. This system will be compatible with various payment gateways.

2. General Description

This system will provide users with secure and efficient way to complete transactions. Its key feature includes real-time authorization, encrypted communication, fraud detection mechanisms.

Fig 2.1

3. Functional Requirements

Authorization – The system will verify the availability of funds and validity of credit card details.

Transaction Processing – It will ensure purchases ensuring ~~and~~ accurate data processing.

Settlement – The system will transfer funds from customer's account to the receiver account.

4. Interface requirements

The system will offer an API for integration with e-commerce platforms.

5. Performance requirement

The system must process transaction within 2 sec under normal condition with failure rate of less than 0.1%.

6. Design Constraints

- Must use secure encryption methods
- should be compatible with major credit card networks.

7. Non-Functional

- Uptime should be at least 99.99%.
- All data must be encrypted during transmission

8. Preliminary Schedule & Budget

The project is estimated to be completed within 6 months and total development budget of 1 lac.

Development – 35,000, Software license – 10,000, Hardware – 25,000, Miscellaneous – 30,000

Fig 2.2

UML DIAGRAMS

CLASS DIAGRAM

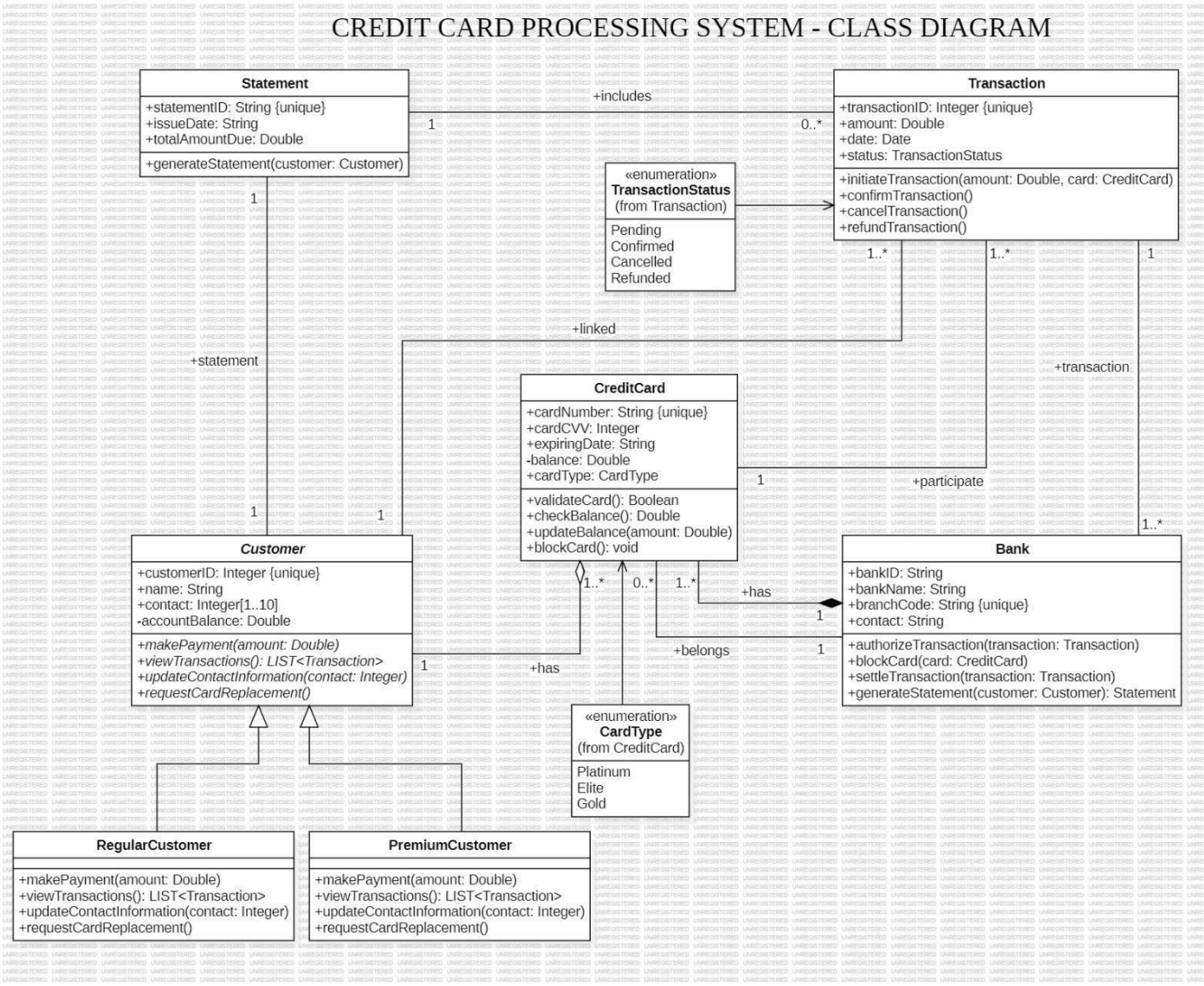


Fig 2.3

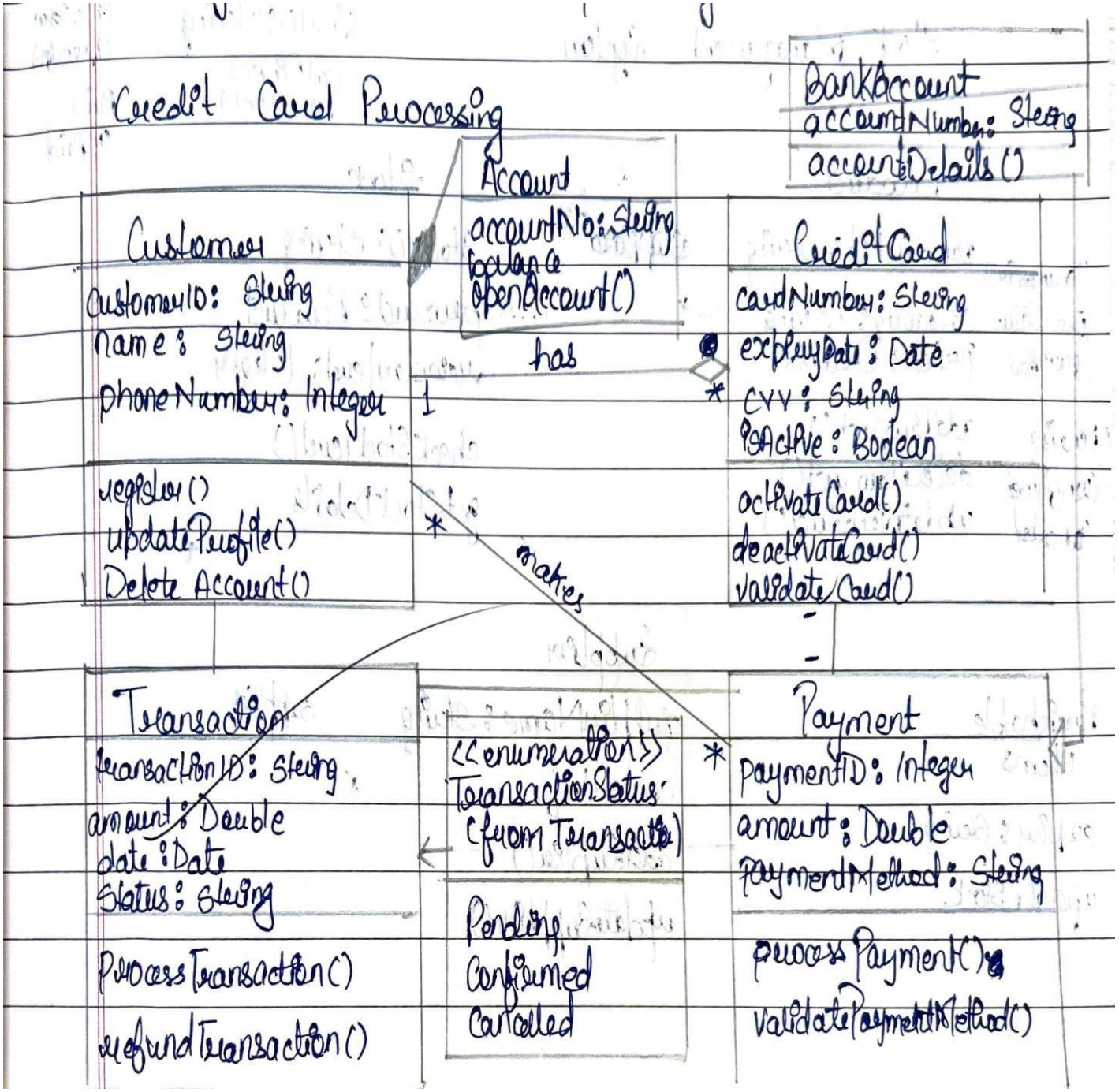

Credit Card Processing
BankAccount
accountNumber: String
accountDetails()
Account
accountNo: String
balance
openAccount()
Customer
customerID: String
name: String
phoneNumber: Integer
register()
updateProfile()
DeleteAccount()
has
1
*
CreditCard
cardNumber: String
expiryDate: Date
CVV: String
isActive: Boolean
activateCard()
deactivateCard()
validateCard()
makes
*
Transaction
transactionID: String
amount: Double
date: Date
status: String
processTransaction()
refundTransaction()
<<enumeration>>
TransactionStatus
(from Transaction)
Pending
Confirmed
Cancelled
Payment
paymentID: Integer
amount: Double
paymentMethod: String
processPayment()
validatePaymentMethod()

Fig 2.4

Brief Description:

1. Statement

- Represents a billing statement for a customer containing details of transactions and the total amount due.
- Includes a list of transactions associated with the statement.

2. Transaction

- Represents a single financial transaction, such as a purchase or payment.
- Contains information like transaction ID, amount, date, and status (Pending, Confirmed, Cancelled, Refunded).

3. CreditCard

- Represents a credit card issued to a customer.
- Contains attributes like card number, CVV, expiration date, card type, and current balance.
- Includes methods for validating card details and updating the balance.

4. Customer

- Represents a customer of the credit card processing system.
- Contains attributes like customer ID, name, contact information, and account balance.
- Includes methods for making payments, viewing transactions, updating contact information, and requesting card replacement.

5. Bank

- Represents a financial institution that issues credit cards and processes transactions.
- Contains attributes like bank ID, name, branch code, and contact information.
- Includes methods for authorising and settling transactions, generating statements, and blocking cards.

6. RegularCustomer

- Represents a customer with standard credit card features and functionalities.

7. PremiumCustomer

- Represents a customer with premium credit card features and benefits, such as higher credit limits or additional rewards.

8. CardType

- Represents the different types of credit cards offered by the bank (e.g., Platinum, Elite, Gold).

9. transaction status

- Represents the possible states of a transaction (Pending, Confirmed, Cancelled, Refunded).

STATE DIAGRAM

SIMPLE STATE DIAGRAM

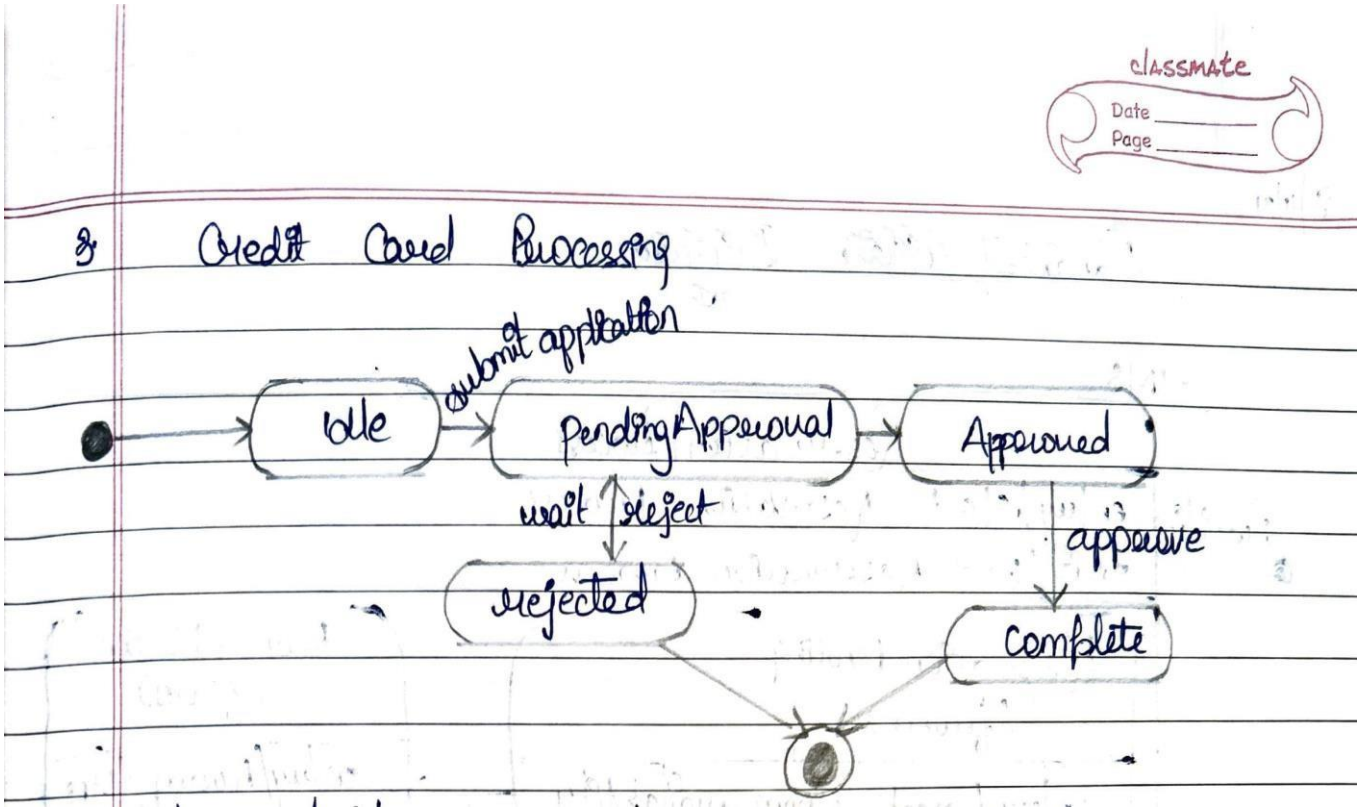


Fig 2.5

ADVANCED STATE DIAGRAM

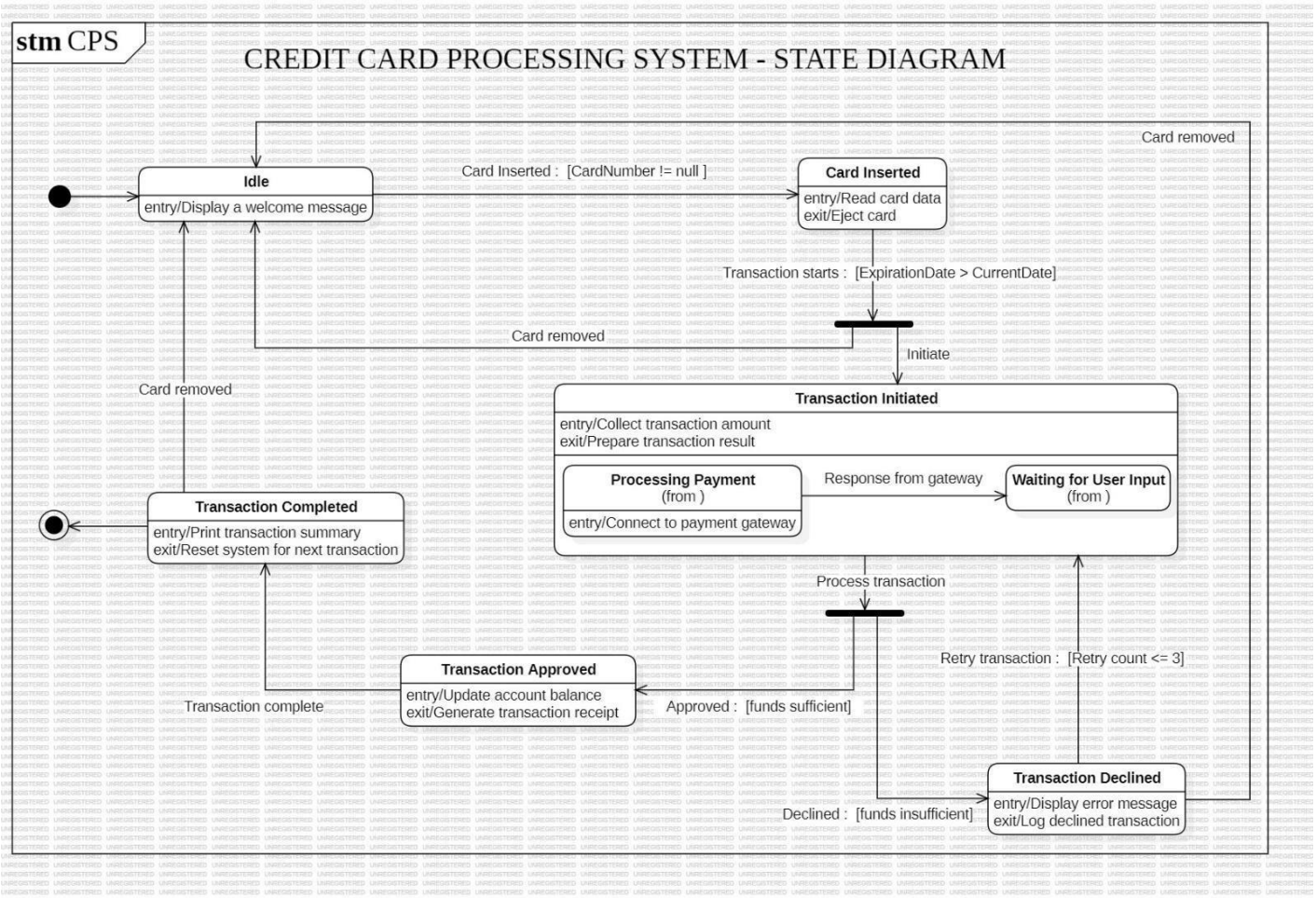


Fig 2.6

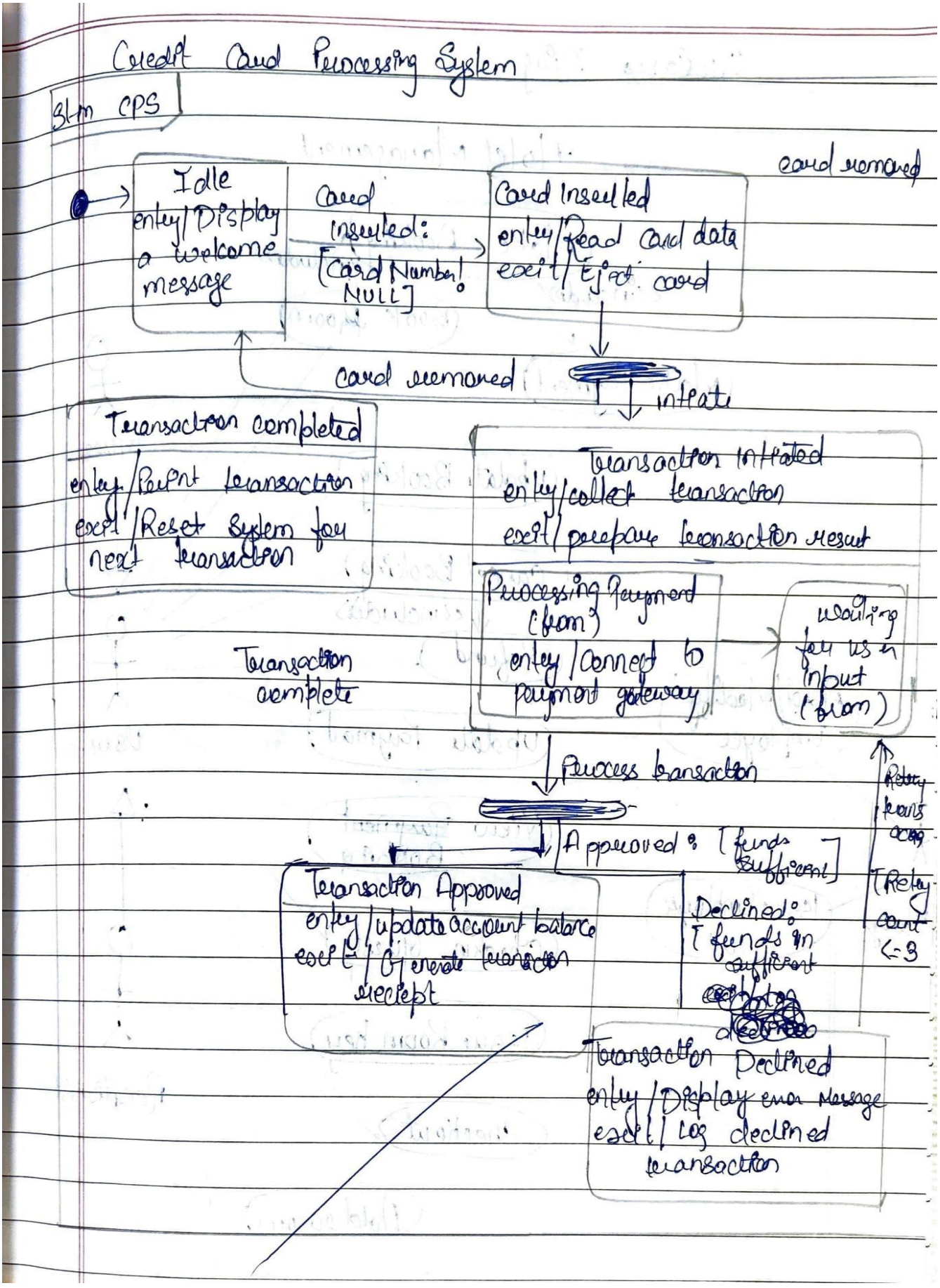

Credit Card Processing System
Sim CPS
Idle
entry/ Display a welcome message
Card inserted:
[Card Number != NULL]
Card Inserted
entry/ Read card data
exit/ Eject card
card removed
card removed
Transaction completed
entry/ Print transaction
exit/ Reset System for next transaction
Transaction initiated
entry/ collect transaction
exit/ prepare transaction result
Transaction complete
Processing Payment (from)
entry/ Connect to payment gateway
waiting for user input (from)
Process transaction
Approved: [funds sufficient]
Declined: [funds insufficient]
Retry trans acc
[Retry count <= 3
Transaction Approved
entry/ update account balance
exit/ Generate transaction receipt
Transaction Declined
entry/ Display error message
exit/ Log declined transaction

Fig 2.7

Brief Description:

1. Idle:
    - The initial state of the system.
    - Displays a welcome message to the user.
2. Card Inserted:
    - The state after a card is inserted.
    - Reads card data and checks for valid expiration date.
3. Transaction Initiated:
    - The state where the transaction process begins.
    - Collects transaction amount and initiates communication with the payment gateway.
4. Processing Payment:
    - The state where the system is interacting with the payment gateway.
    - Connects to the payment gateway and processes the transaction.
5. Waiting for User Input:
    - The state where the system awaits further input from the user.
    - It may be used for additional authorisation steps or error resolution.
6. Transaction Approved:
    - The state when the transaction is successfully approved.
    - Updates the account balance and generates a transaction receipt.
7. Transaction Declined:
    - The state when the transaction is declined.
    - Displays an error message to the user and logs the declined transaction.
8. Transaction Completed:
    - The final state after a successful or declined transaction.
    - Prints a transaction summary and resets the system for the next transaction.

USE CASE DIAGRAM

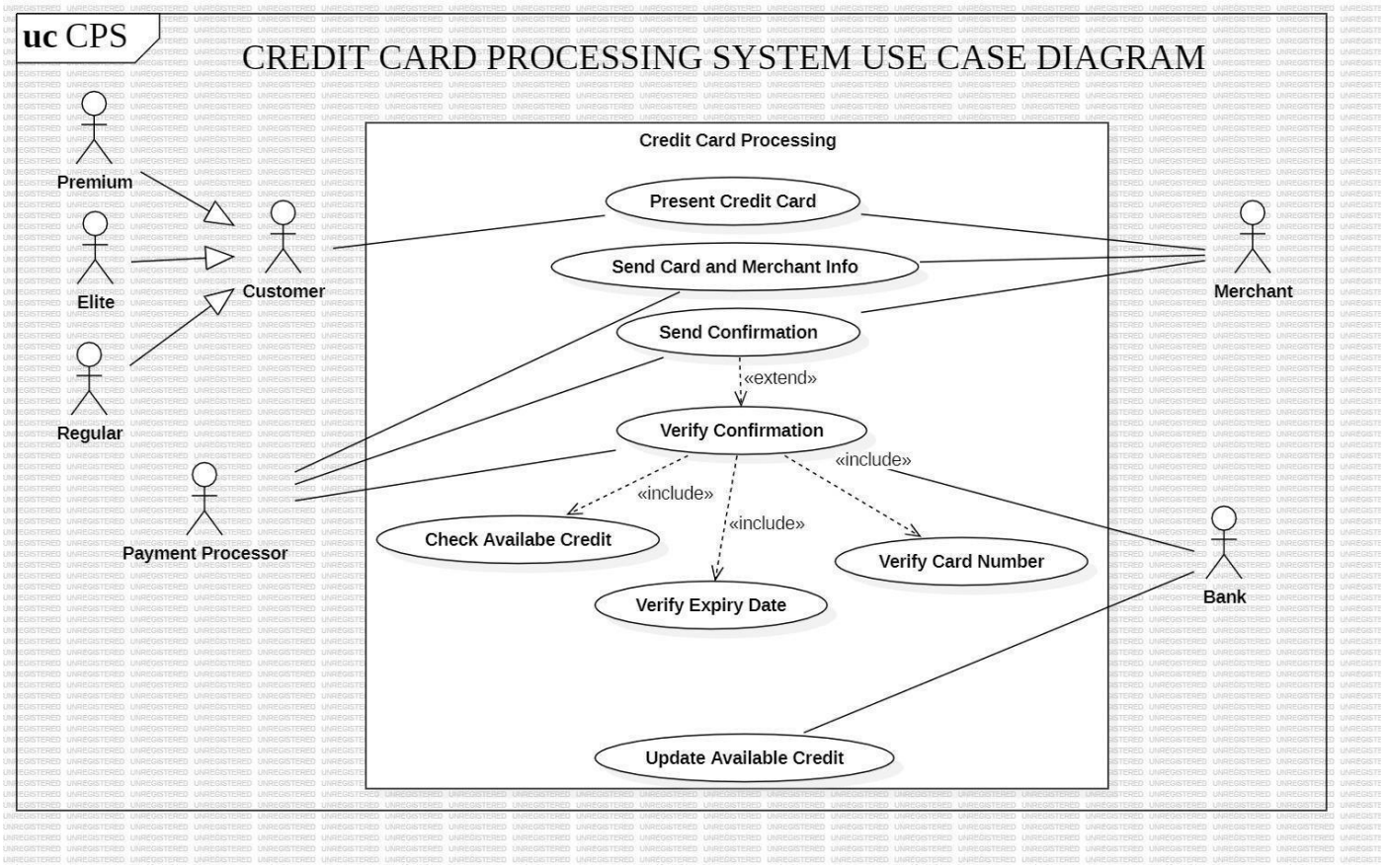


Fig 2.8

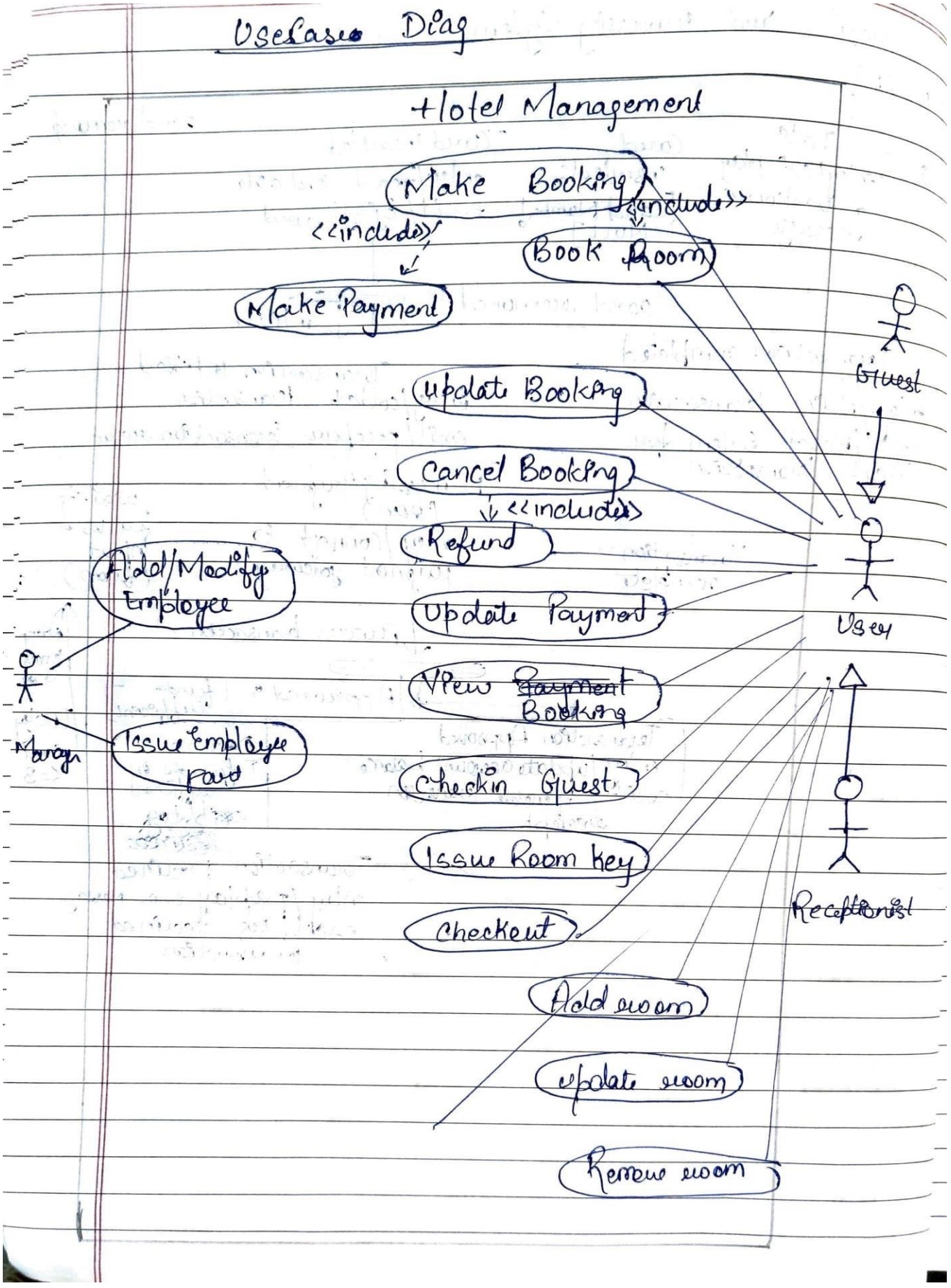

Usecase Diag
Hotel Management
Make Booking
<<include>>
<<include>>
Book Room
Make Payment
Update Booking
Cancel Booking
<<include>>
Refund
Update Payment
View Booking
Checkin Guest
Issue Room key
Checkout
Add room
update room
Remove room
Add/Modify Employee
Issue Employee Paid
Manager
Guest
User
Receptionist

Fig 2.9

Brief Description:

Use Cases

- **Present Credit Card:** The customer presents their credit card for a transaction.
- **Send Card and Merchant Info:** Information about the card and the merchant is sent for processing.
- **Send Confirmation**: A confirmation message is sent to the customer and/or merchant.
- **Verify Confirmation:** This use case includes several sub-use cases:
    - **Check Available Credit:** Verifies if the customer has sufficient credit available.
    - **Verify Card Number:** Validates the card number for accuracy.
    - **Verify Expiry Date:** Check if the card has not expired.
- **Update Available Credit:** Updates the customer's available credit balance after a successful transaction.

Actor Interactions

- **Customer:** Initiates the transaction by presenting the credit card and receiving confirmation.
- **Merchant:** Receives payment information and sends confirmation to the customer.
- **Payment Processor**: Handles the verification and processing of the transaction.
- **Bank:** Provides credit information and updates the customer's account balance.

Relationships

- **Include:** Indicates that one use case includes the functionality of another. For example, "Verify Confirmation" includes "Check Available Credit," "Verify Card Number," and "Verify Expiry Date."
- **Extend:** Represents optional behaviour that can be added to a use case under specific conditions.

SEQUENCE DIAGRAM

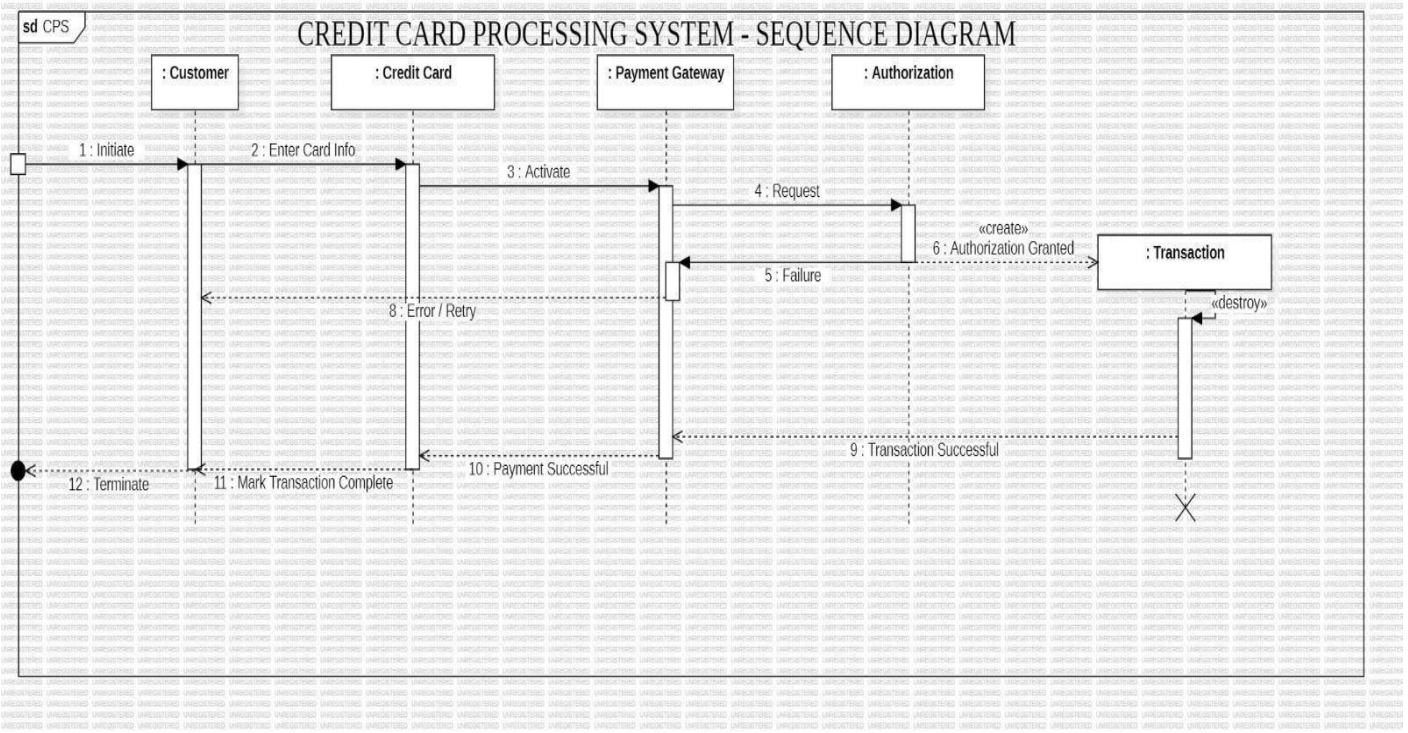


Fig 2.10

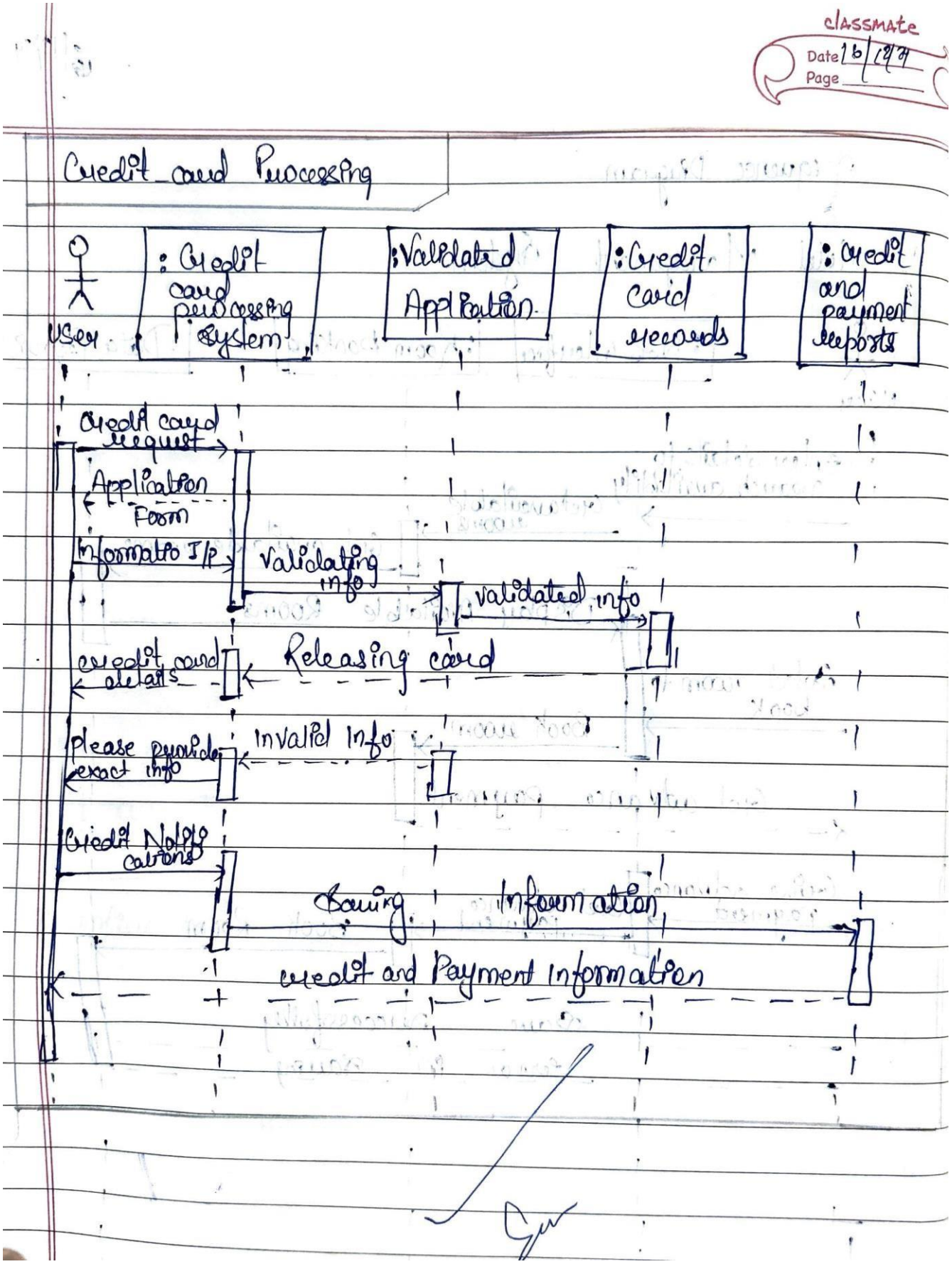

Credit card Processing
User
: Credit card processing System
:Validated Application
: Credit card records
: Credit and payment reports
Credit card request
Application Form
Information I/P
Validating info
validated info
credit card details
Releasing card
Invalid Info
please provide exact info
Credit Validations
Saving Information
credit and Payment information

Fig 2.11

Brief Description:

Interactions

1. **Initiate:** The customer initiates a transaction.
2. **Enter Card Info:** The customer enters their card information.
3. **Activate:** The credit card is activated for the transaction.
4. **Request:** The system sends a request to the payment gateway for authorisation.
5. **Failure:** If authorisation fails, an error message is displayed.
6. **Authorisation Granted:** If authorisation is successful, a transaction object is created.
7. **Payment Successful:** The payment is processed successfully.
8. **Error/Retry:** If an error occurs, the system may attempt to retry the transaction.
9. **Transaction Successful:** The transaction is completed successfully.
10. **Mark Transaction Complete:** The system marks the transaction as complete.
11. **Terminate:** The transaction process is terminated.

ACTIVITY DIAGRAM

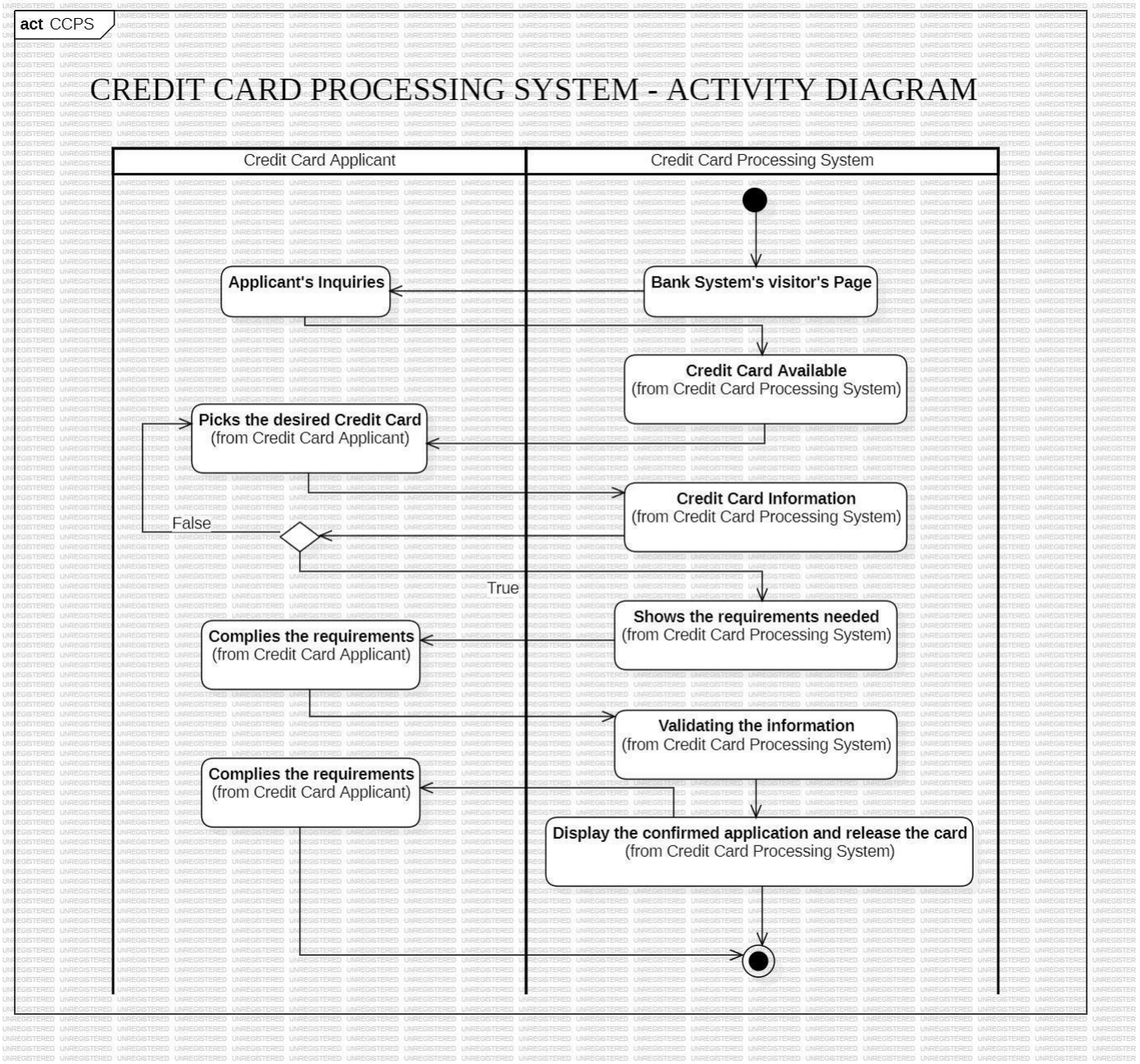


Fig 2.12

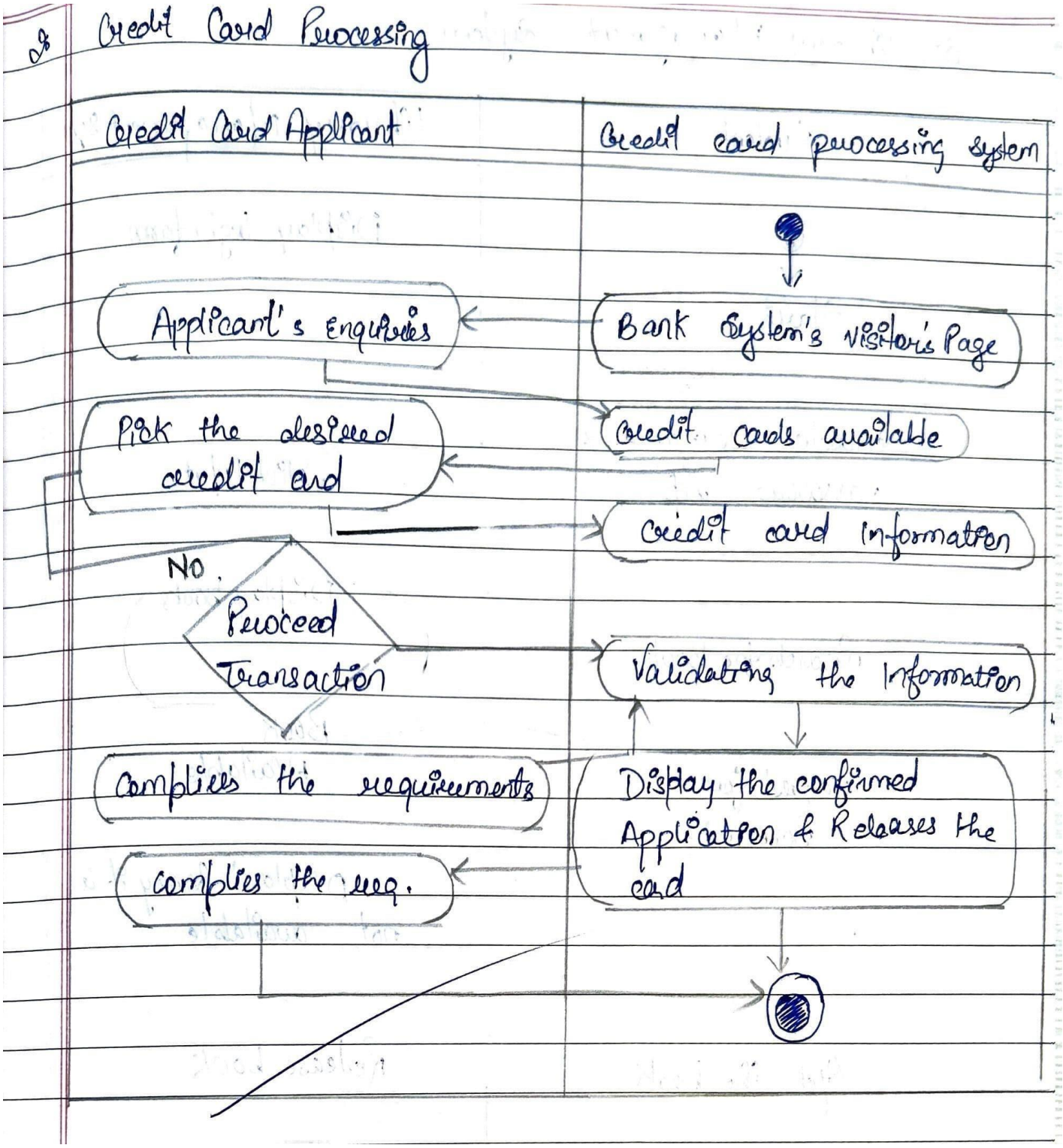

Credit Card Processing
Credit Card Applicant
Credit card processing system
Applicant's Enquiries
Bank System's visitor's Page
Pick the desired credit card
Credit cards available
Credit card information
No.
Proceed Transaction
Validating the Information
Complies the requirements
Display the confirmed Application & Releases the card
complies the req.

Fig 2.13

Brief description:

Activities:

1. **Applicant's Inquiries:** The process starts with the applicant making inquiries about available credit cards.
2. **Bank System's Visitor's Page:** The applicant is presented with information about available credit card options.
3. **Picks the Desired Credit Card:** The applicant selects a specific credit card based on their needs and preferences.
4. **Credit Card Information:** The system displays detailed information about the selected credit card, including terms and conditions.
5. **Complies with the Requirements:** The applicant reviews and confirms that they meet the eligibility criteria for the selected credit card.
6. **Shows the Requirements Needed:** If the applicant does not meet the requirements, the system displays the requirements to be eligible.
7. **Validating the Information:** Once the applicant meets the requirements, the system validates the information provided in the application.
8. **Display the Confirmed Application and Release the Card:** If the application is approved, the system displays a confirmation message and initiates the process of issuing the credit card to the applicant.

LIBRARY MANAGEMENT SYSTEM

Problem Statement:

Develop a comprehensive Library Management System to streamline operations and improve user experience. The system should encompass the following functionalities:

- **Book Management:** Maintain a comprehensive book catalogue, track loans and returns and manage book acquisitions and damage.
- **User Management:** Register and manage library members, issue cards, and control user access privileges.
- **Search and Cataloguing:** Provide a user-friendly search interface, generate a library catalogue, and support cataloguing standards.
- **Circulation Management:** Process loans and returns efficiently, manage reservations, and handle overdue notices and fines.
- **Reporting and Analytics:** Generate reports on book usage circulation statistics and track library inventory trends.

classmate
Date 30/9/24
Page

Library Management System

1. Introduction

1.1 Purpose of this document

The SRS document outlines the detailed requirement for Library Management system. It serves as a blue print for the development team, ensuring final product meets the specified specification.

1.2. Scope of the Document

This document aims to include book details, video lectures, magazine, previous year question papers. It will be used by students, staffs, faculty, indirect stakeholders like technician.

1.3 Overview

The Library Management System will provide solution for managing operations like issuing books, catalogues, fine management, member management, transaction tracking. This will enable staff and other users to efficiently use and offer userfriendly interface.

2. General Description

The LMS will provide a platform for managing library resources efficiently. This system will provide registering students, catalogging books, issuing/returning books.

3. Functional Requirements

Registeration Management – Users should be able to register to the system.

Fig 3.1

List of

- Users should be able to search for available overdues.
- System should be able to store registeration details like contact information, etc.

Book Management:
The system will allow for addition, modification of book details.

Search facility - users should be able to search for books based on title, ISBN.

Interface Requirements:
The system should be able to graphical user interface accessible through web browsers.
The system will integrate existing database to store book and user information.

Performance Requirements:
The system must be able to handle 1,00,000 books and 10,000 users.
Book search result should be available within 3 sec.

Design Constraints
- The database should support MySQL for large database.
- Must be compatible with modern web browsers (Google chrome).

Non-functional.
- Uptime should be atleast 99.99%.
- The system must protect user data from unauthorised access.

Preliminary Schedule & Budget:
The project is expected to take four months with development - 35,000, software license - 10,000; budget of 1.0 lacs.

Fig 3.2

UML DIAGRAMS

CLASS DIAGRAM

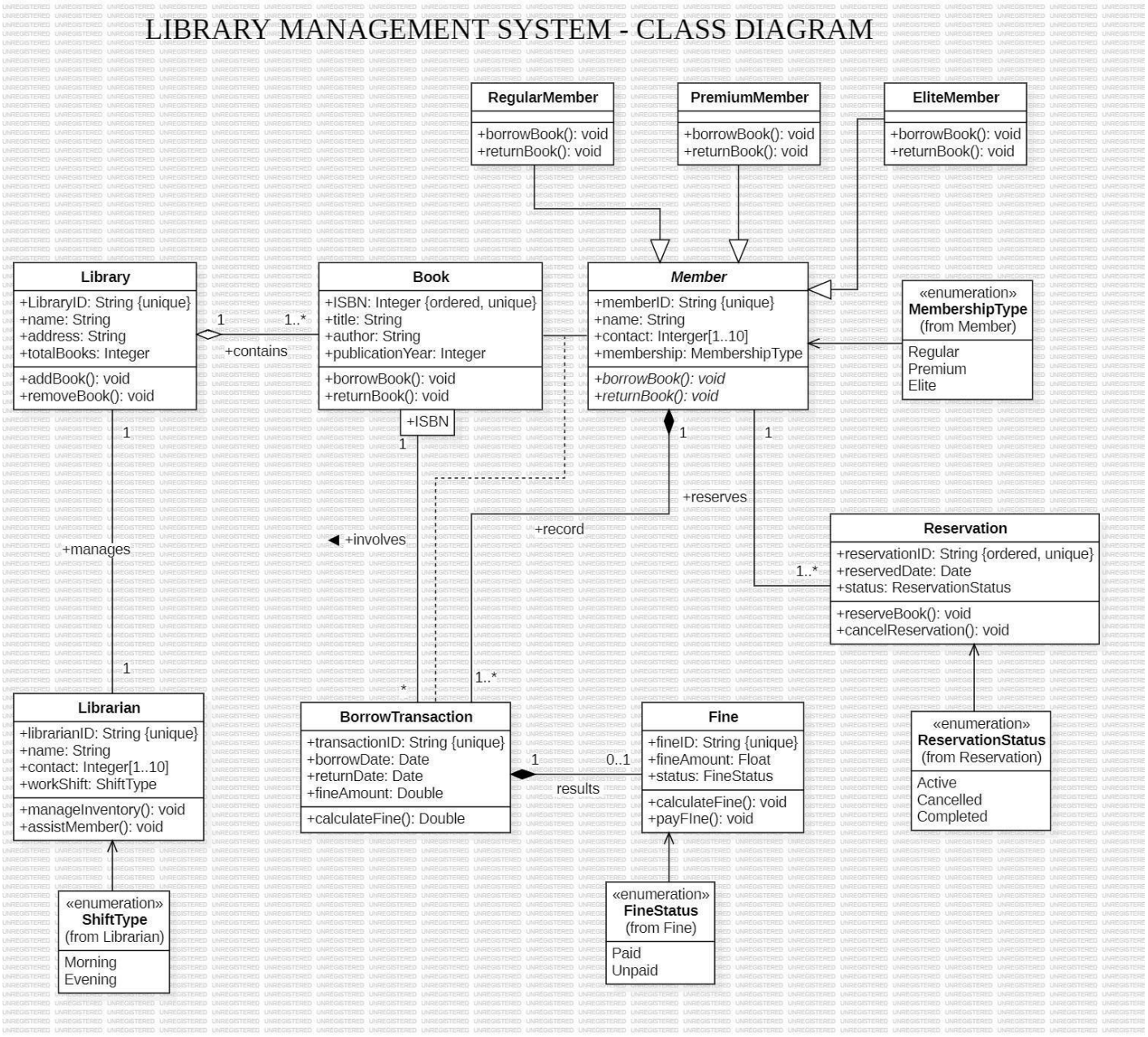


Fig 3.3

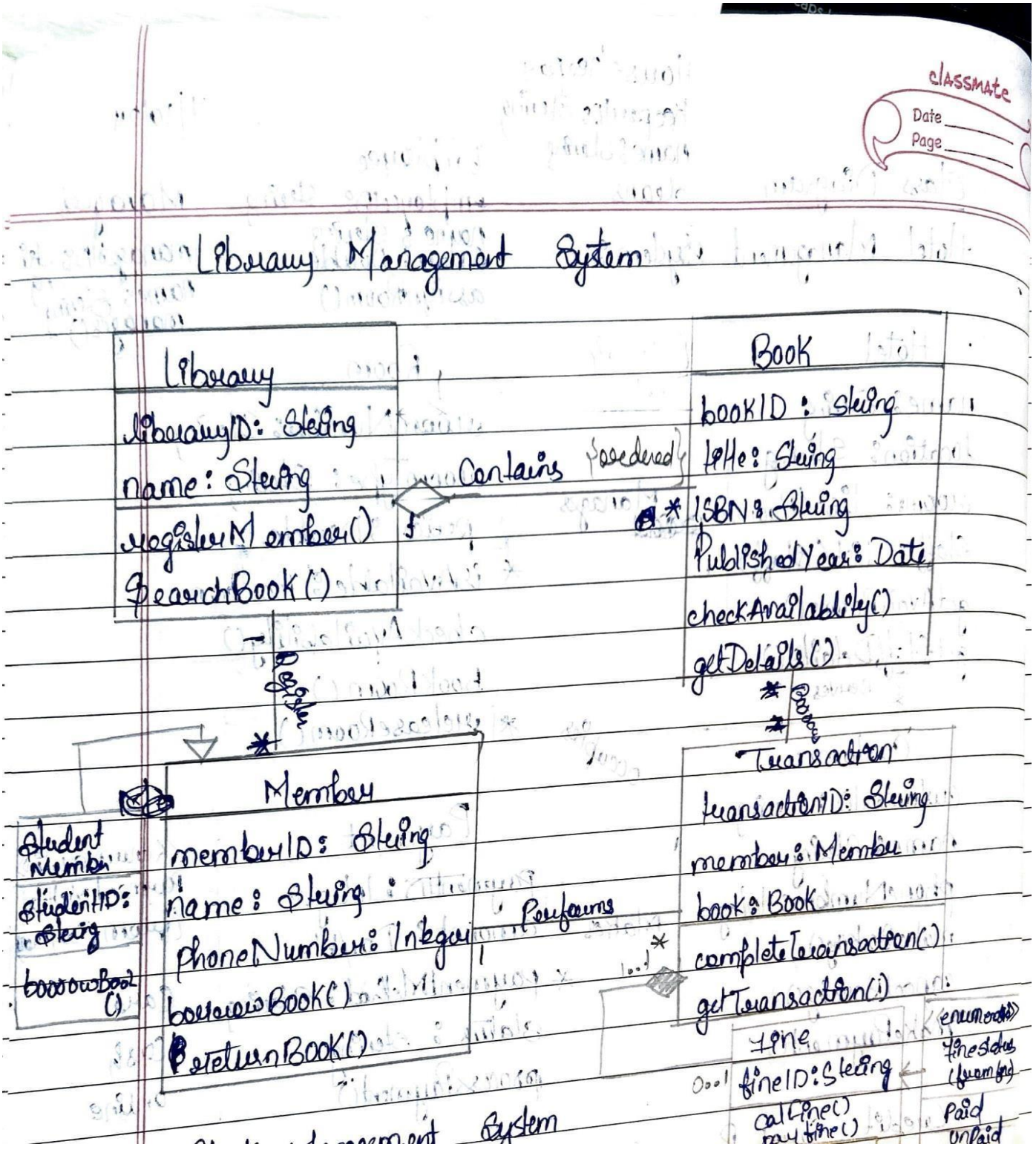

Library Management System
Library
libraryID: String
name: String
registerMember()
searchBook()
Contains
Recorded
Book
bookID: String
title: String
ISBN: String
PublishedYear: Date
checkAvailability()
getDetails()
Member
memberID: String
name: String
phoneNumber: Integer
borrowBook()
returnBook()
Performs
Transaction
transactionID: String
member: Member
book: Book
completeTransaction()
getTransaction()
Fine
fineID: String
calFine()
payFine()
Student Member
studentID: String
borrowBook()
Paid
Unpaid

Fig 3.4

Brief description:

1. Library

- Represents the library entity.
- Manages book inventory, adds and removes books, and tracks total books.

2. Book

- Represents a book in the library.
- Has attributes like ISBN, title, author, publication year, and is involved in borrow transactions.

3. Member

- Represents a library member (Regular, Premium, or Elite).
- Has attributes like member ID, name, contact information, and membership type.

4. Librarian

- Represents a library staff member.
- Manages inventory, assists members, and works in shifts (Morning/Evening).

5. BorrowTransaction

- Represents a record of a book being borrowed by a member.
- Includes transaction ID, borrow date, return date, and calculates fines.

6. Fine

- Represents a fine imposed on a member.
- Has attributes like fine ID, amount, and status (Paid/Unpaid).

7. Reservation

- Represents a reservation made by a member for a book.
- Has attributes like reservation ID, reserved date, and status (Active/Cancelled/Completed).

8. MembershipType

- Enumeration representing different membership types (Regular, Premium, Elite).

9. ReservationStatus

- Enumeration representing the status of a reservation (Active, Cancelled, Completed).

10. ShiftType

- Enumeration representing librarian shift types (Morning, Evening).

SIMPLE STATE DIAGRAM

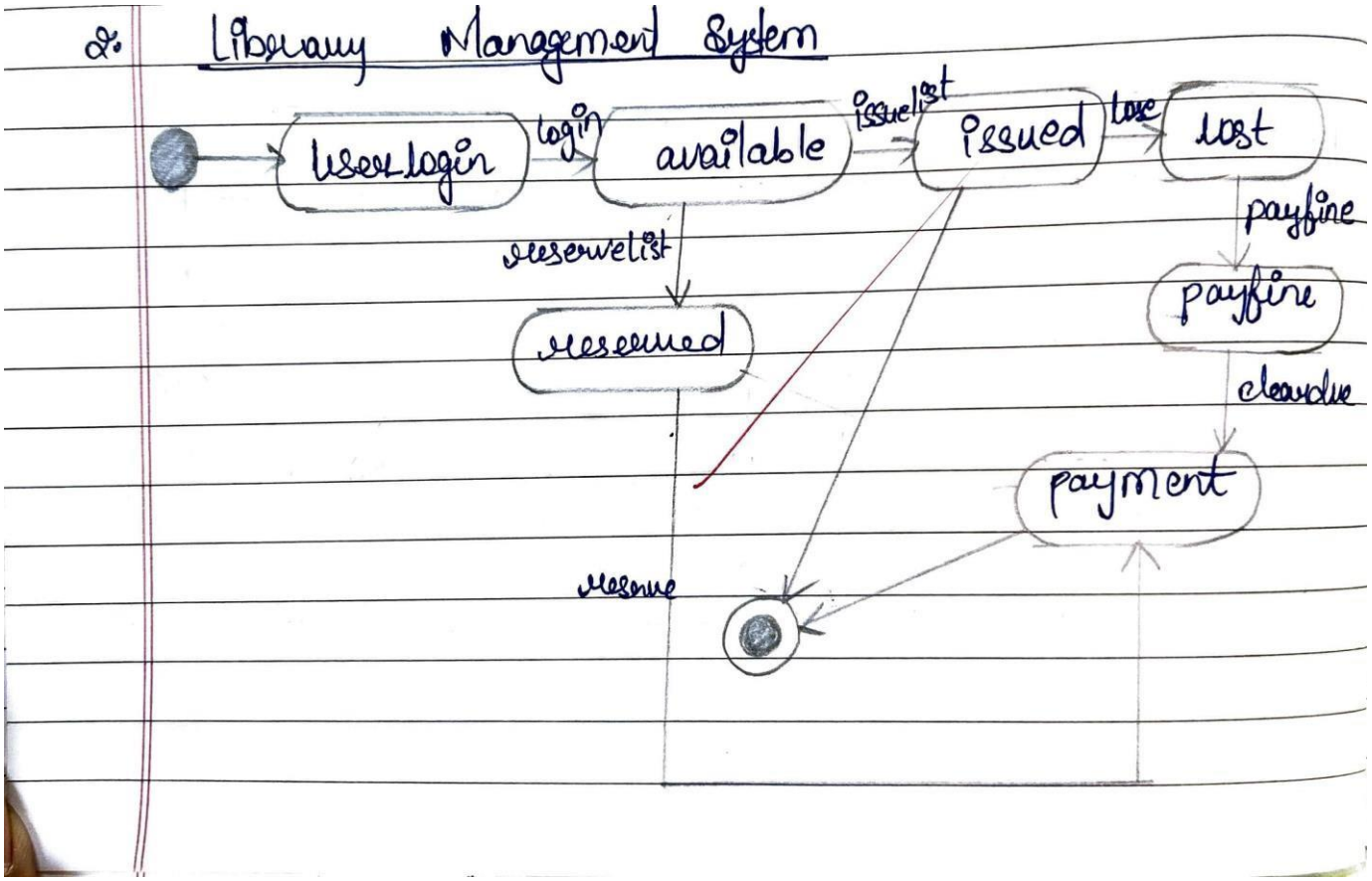


Fig 3.5

ADVANCE STATE DIAGRAM

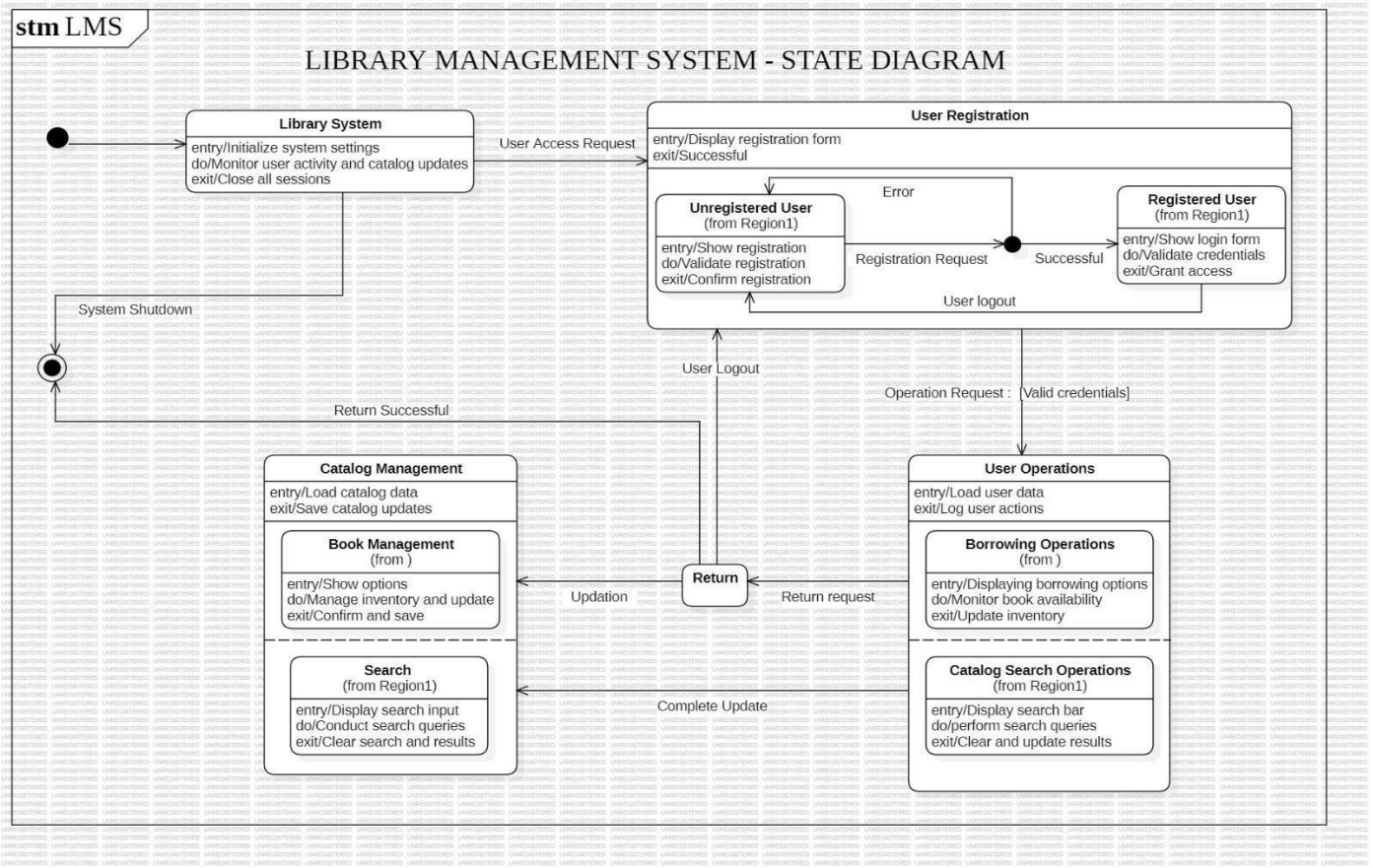


Fig 3.6

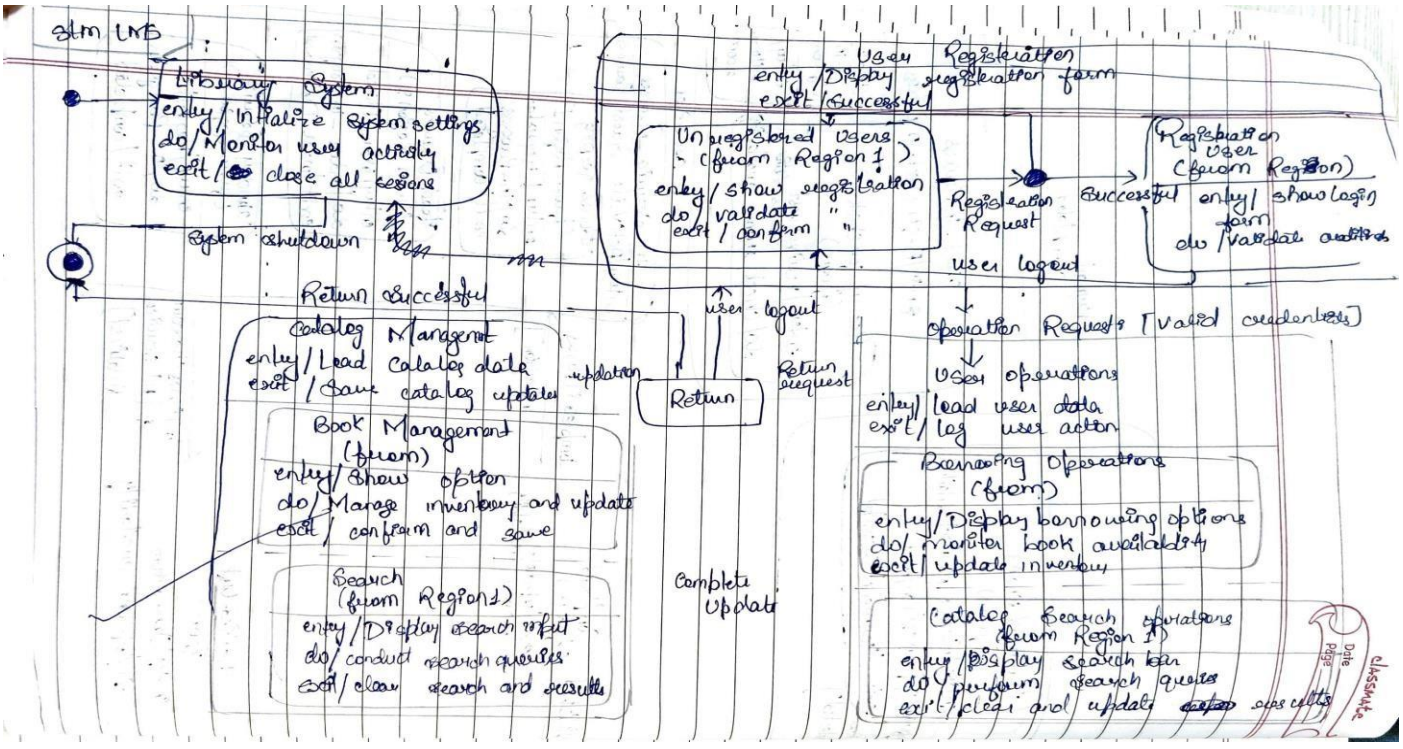


Fig 3.7

Brief description:

1. Library System

- The initial state of the system, where system settings are initialised, and user activity/catalogue updates are monitored.

2. User Registration and Unregistered User

- The state where new users can register for library membership.
- The state of a user before they have registered for library membership.

3. Registered User

- The state of a user after they have successfully registered and logged in.

4. Catalog Management

- The state where the library's catalogue data is managed and updated.

5. Book Management

- The state where book inventory is managed, including adding, removing, and updating book information.

6. Search

- The state where users can search the library catalogue for books.

7. Borrowing Operations

- The state where users can borrow books, renew loans, and return books.

8. Catalog Search Operations

- The state where users can search the library catalogue for specific books or information.

9. Return

- The state where a user returns a borrowed book.

10. Operation Request

- The state where a user requests to operate, such as borrowing a book or searching the catalogue.

11. Return Successful

- The state indicates that a book has been successfully returned.

12. User Access Request and Logout

- The state where a user requests access to the library system.

13. System Shutdown

- The final state of the system is where all sessions are closed, and the system is shut down.

USE CASE DIAGRAM

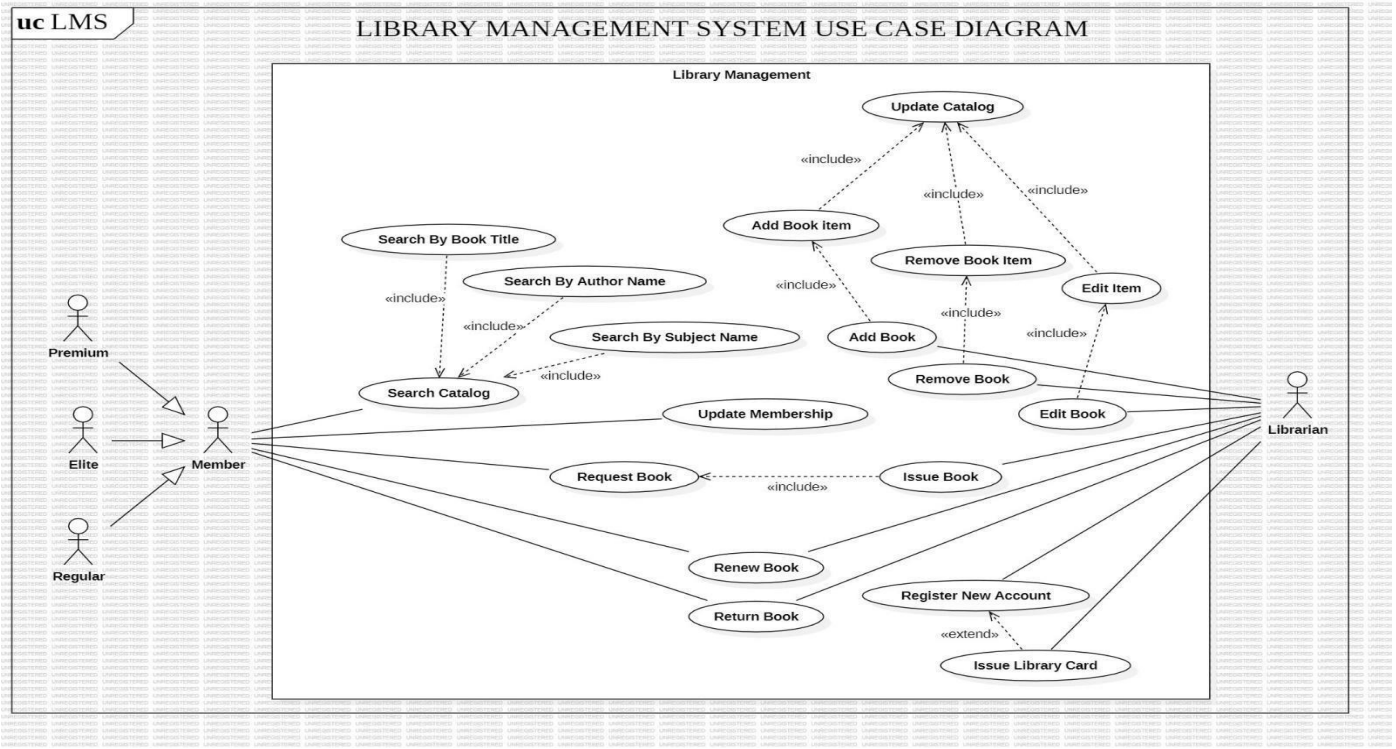


Fig 3.8

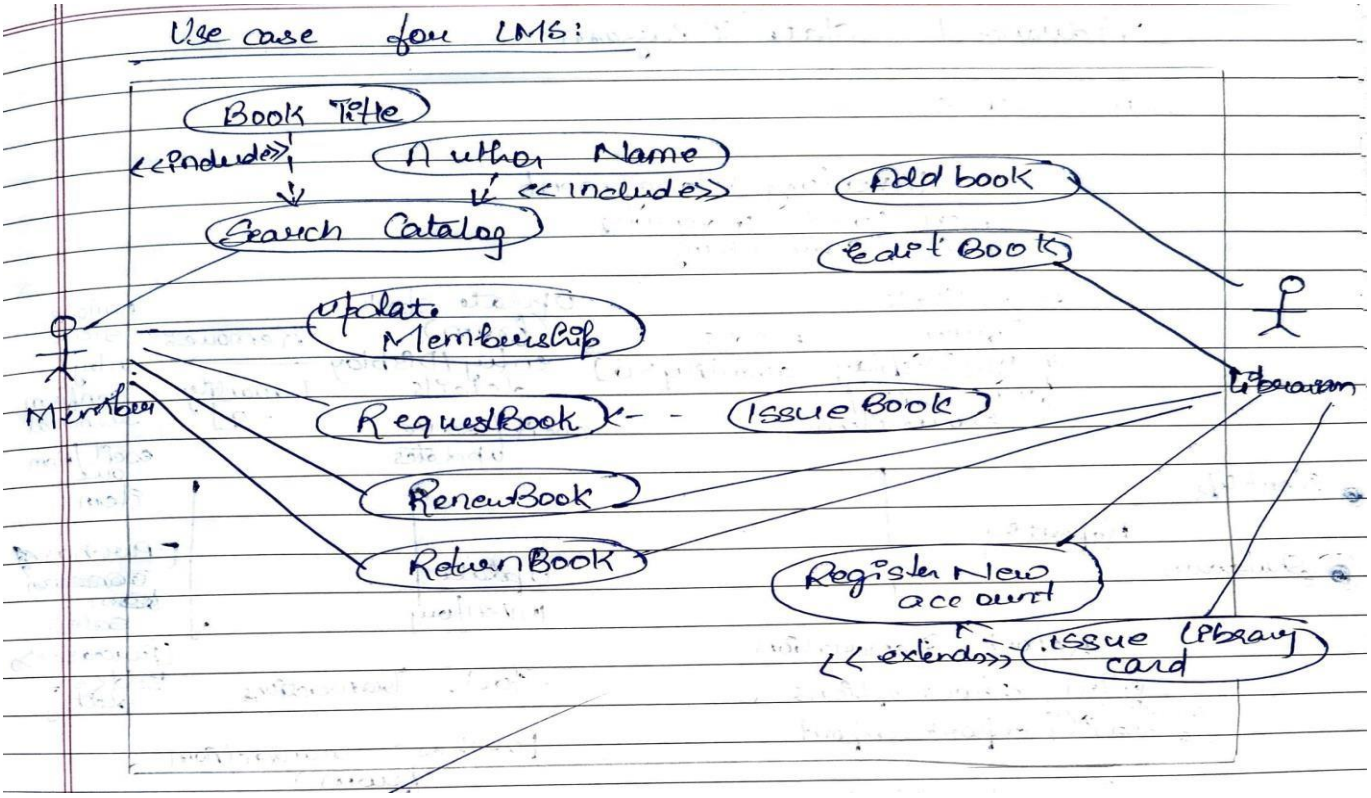


Fig 3.9

Brief description:

1. Library Management

- This is the overarching use case, encompassing all other functionalities within the library system.

2. Update Catalog

- This use case includes several sub-use cases related to managing the library catalogue:
  - **Add Book Item:** Adding a new book item to the catalogue.
  - **Remove Book Item:** Remove a book item from the catalogue.
  - **Edit Item:** Modifying the details of a book item in the catalogue.

3. Search Catalog

- This use case also includes several sub-use cases for searching the catalogue:
  - **Search By Book Title:** Searching the catalogue by the title of the book.
  - **Search By Author Name:** Searching the catalogue by the author of the book.

4. Update Membership

- This use case allows for updating the membership information of library members.

5. Request Book

- This use case allows members to request a book that is currently unavailable or checked out.

6. Issue Book

- This use case handles the process of issuing a book to a member.

7. Renew Book

- This use case allows members to renew their book loans.

8. Return Book

- This use case handles the process of returning a borrowed book.

9. Register a New Account

- This use case handles the process of registering new members.

10. Issue Library Card

- This use case is a sub-use case of "Register New Account" and specifically handles the issuance of library cards to new members.

SEQUENCE DIAGRAM

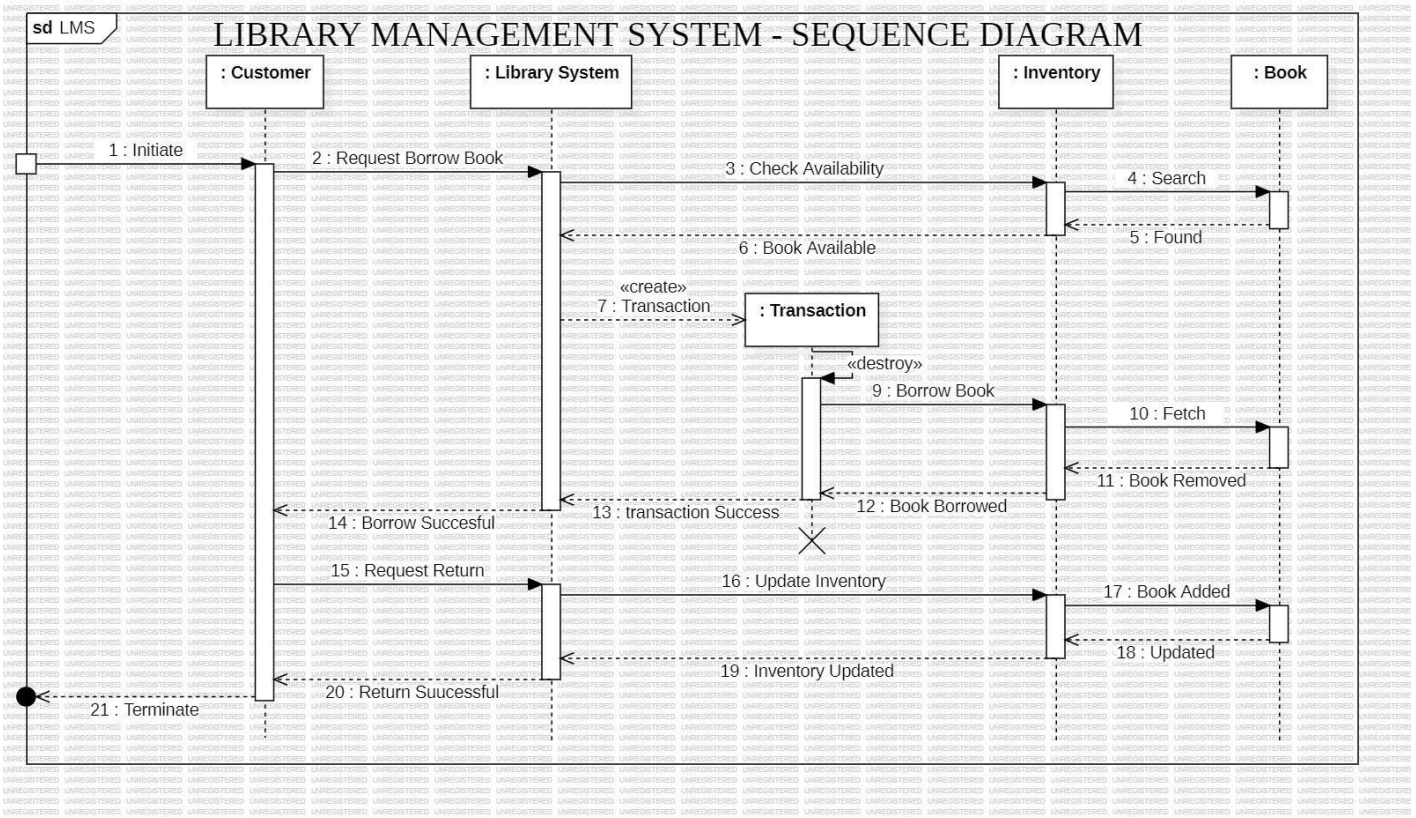


Fig 3.10

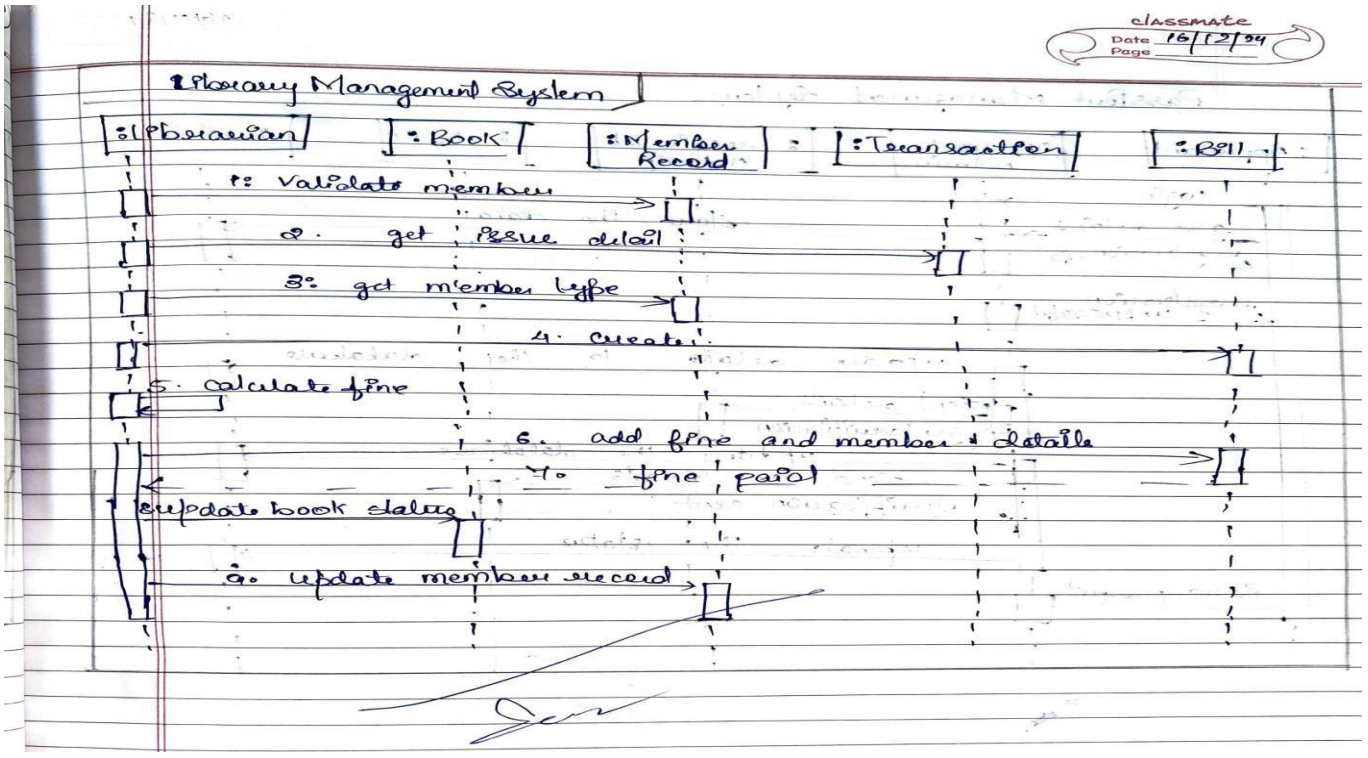


Fig 3.11

Brief description:

Interactions

1. **Initiate:** The customer initiates a request to borrow a book.
2. **Request Borrow Book:** The customer sends a request to the library system to borrow a specific book.
3. **Check Availability:** The library system checks the availability of the requested book in the inventory.
4. **Search:** The library system searches the inventory for the requested book.
5. **Found:** The inventory confirms that the book has been found.
6. **Book Available:** The inventory indicates that the book is available for borrowing.
7. **Transaction:** A new transaction object is created to record the borrowing process.
8. **Borrow Book:** The library system records the borrowing of the book.
9. **Fetch:** The book object is fetched from the inventory.
10. **Book Borrowed:** The book is marked as borrowed in the inventory.
11. **Book Removed:** The book is removed from the available inventory.
12. **Borrow Successful:** The library system confirms that the book has been successfully borrowed.
13. **Transaction Success:** The transaction is marked as successful.
14. **Request Return:** The customer requests to return the borrowed book.
15. **Update Inventory:** The library system updates the inventory to reflect the return of the book.
16. **Book Added:** The book is added back to the available inventory.
17. **Updated:** The inventory is updated to reflect the returned book.
18. **Inventory Updated:** The inventory is confirmed as updated.
19. **Return Successful:** The library system confirms that the book has been successfully returned.
20. **Terminate:** The transaction process is terminated.

ACTIVITY DIAGRAM

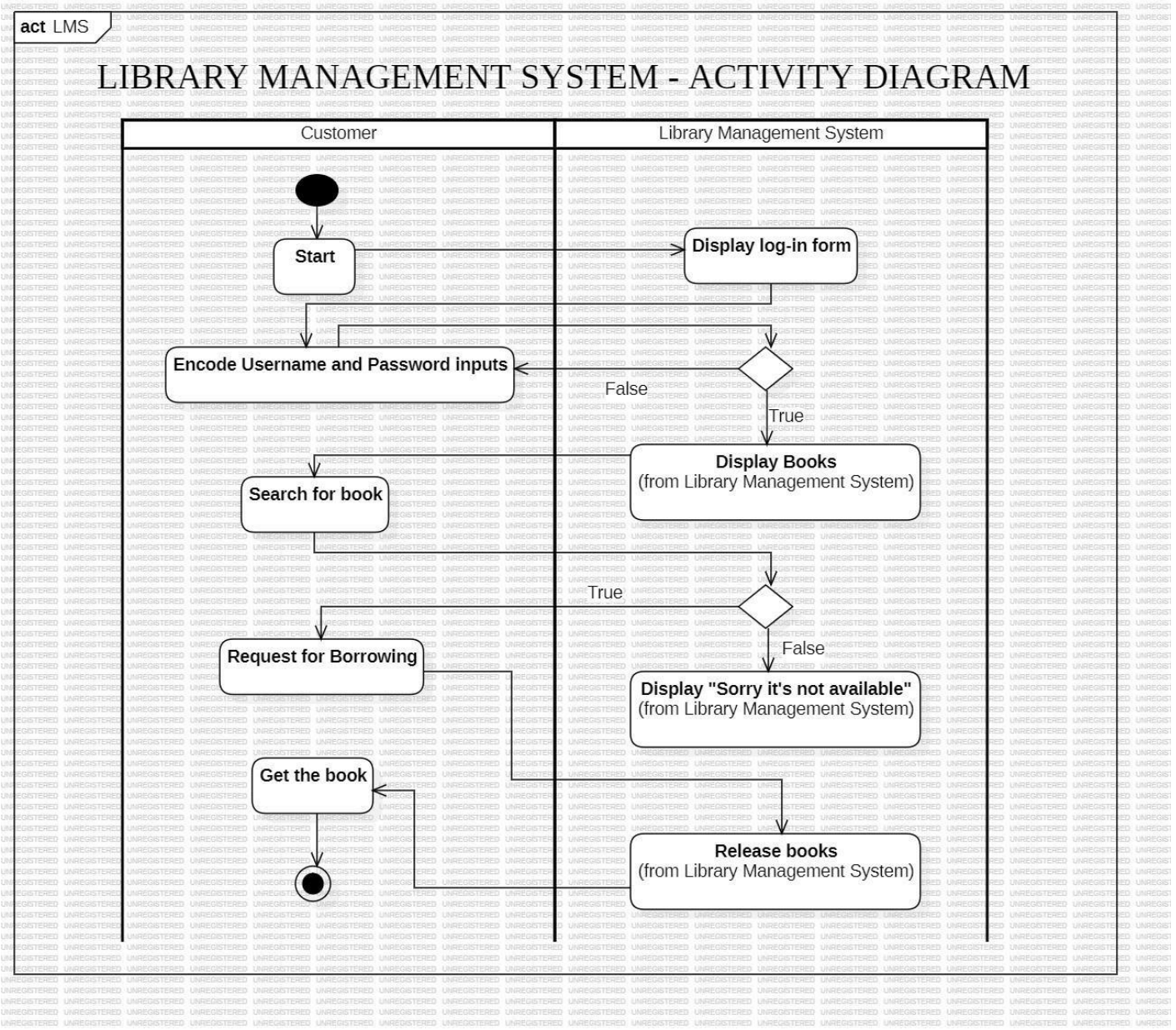


Fig 3.11

3. Library Management System

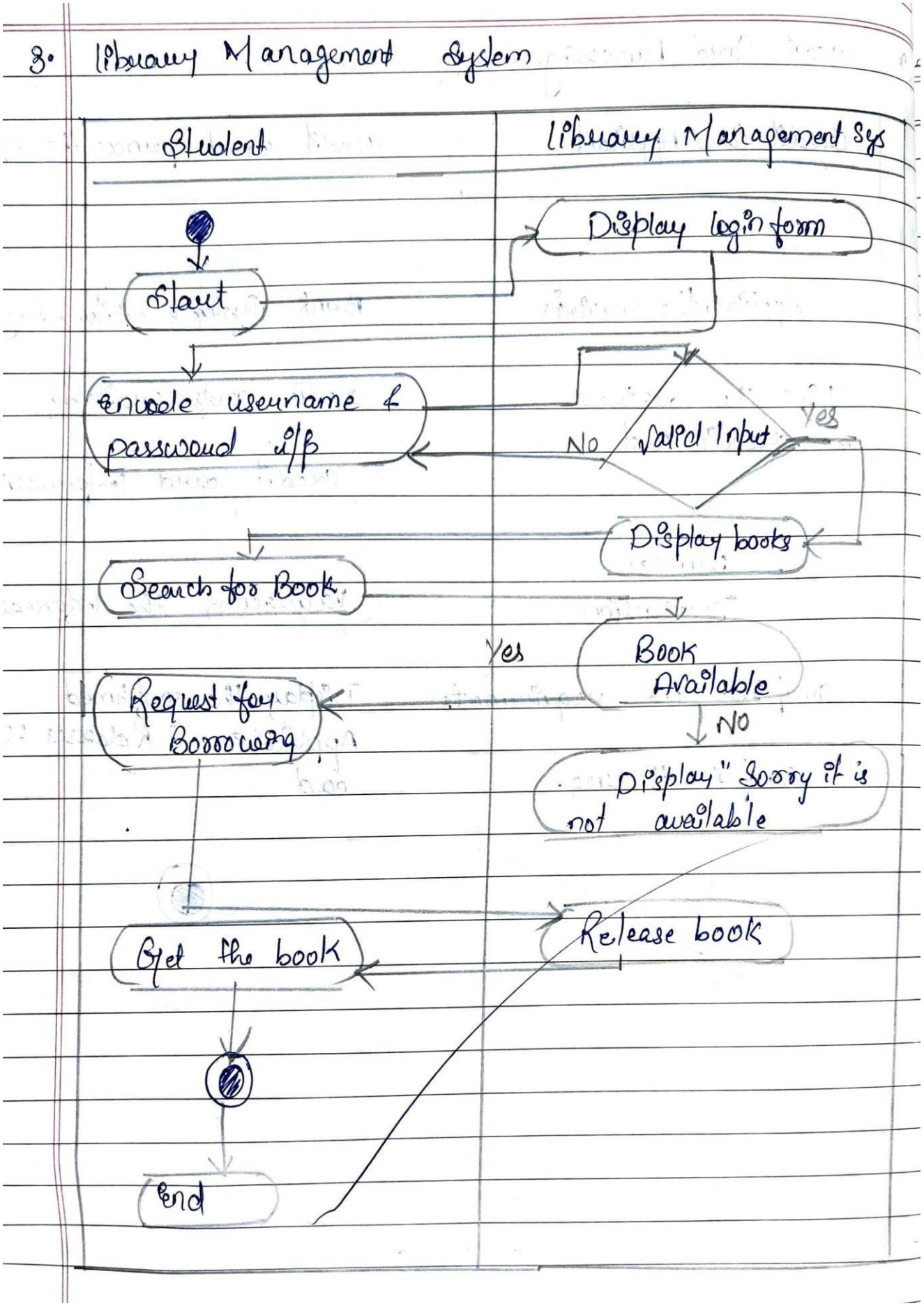


Fig 3.12

Brief description:

Activities

1. **Start:** The process begins with the customer starting the interaction with the system.
2. **Display Log-in Form:** The library management system displays a login form to the customer.
3. **Encode Username and Password Inputs:** The customer enters their username and password.
4. **Display Books:** If the login credentials are correct, the system displays a list of available books to the customer.
5. **Search for Book:** The customer searches for the book they want to borrow.
6. **Request for Borrowing:** The customer requests to borrow the selected book.
7. **Get the Book:** If the book is available, the system allows the customer to borrow the book.
8. **Release Books:** The customer returns the borrowed book to the library.

STOCK MAINTENANCE SYSTEM

Problem Statement:

Develop a robust and secure Stock Management System to streamline inventory control and enhance operational efficiency for businesses. The system should:

- **Accurate Inventory Tracking:** Maintain real-time and accurate records of stock levels across all locations and warehouses.
- **Efficient Order Management:** Facilitate efficient order processing, including order entry, fulfilment, and shipment tracking.
- **Demand Forecasting:** Predict future stock requirements based on historical data, sales trends, and seasonality.
- **Seamless Integrations:** Integrate with existing ERP, POS, and e-commerce platforms.
- **Robust Reporting:** Generate comprehensive reports on stock levels, sales trends, order history, and other key metrics.
- **User-Friendly Interface:** Provide an intuitive interface for easy data entry, stock adjustments, and order management.
- **Compliance Focus:** Adhere to industry standards and regulations related to inventory management and data security.

Hardware – 25,000
Miscellaneous – 30,000

classmate
Date 30/9/24
Page

Stock Maintenance System

1. Introduction

1.1 Purpose of this Document
The SRS document outlines the detailed requirement for stock Maintenance system. This system will help business to manage and track stock levels accurately.

1.2 Scope of this Document
It aims to manage stocks of various products. It helps to maintain and record purchases of the various stocks. The system is intended for business and retail stores.

1.3 Overview
This SMS will help in tracking realtime stock levels and generate reports based on these stock levels, and purchases.

2. General Description
This stock Maintenance system will help in the process of managing stocks. The primary users will be: business mangers, store employees and owners of various business.

3. Functional Requirements

- The system should be able to track the quantity of stock available for each system.
- Automatically generate purchase reorder of product quantity goes below the minimum level.

Fig 4.1

- The system should be able to generate reports based on current stocks, sales trends.

- The system should display alerts if it goes below low.

4. Interface Requirements
The system will feature user friendly web-based interface for stock management. It also will provide APIs to integrate with other systems like sales platform.

5. Performance Requirements
- The system should handle real-time stocks updates for 1000 products and 10,000 individuals
- Reports should be generated within 5 sec.

6. Design Constraints
- The system should be compatible with modern web browsers like Brave.
- The system must support barcode scanning

7. Non-functional requirements
Security - The system must have implemented access control to protect sensitive information.
Uptime should be atleast 99.99%.

8. Preliminary Schedule
The project is estimated to be complete within 4 months and total development budget of 1 lac.

Requirements - 1 month
Design Phase - 0.5 month
Development Phase - 2 month
Testing Phase - 0.5 month

Development - 35,000; Miscellaneous - 30,000
Software license - 10,000
+ hardware - 25,000

Fig 4.2

UML DIAGRAMS

CLASS DIAGRAM

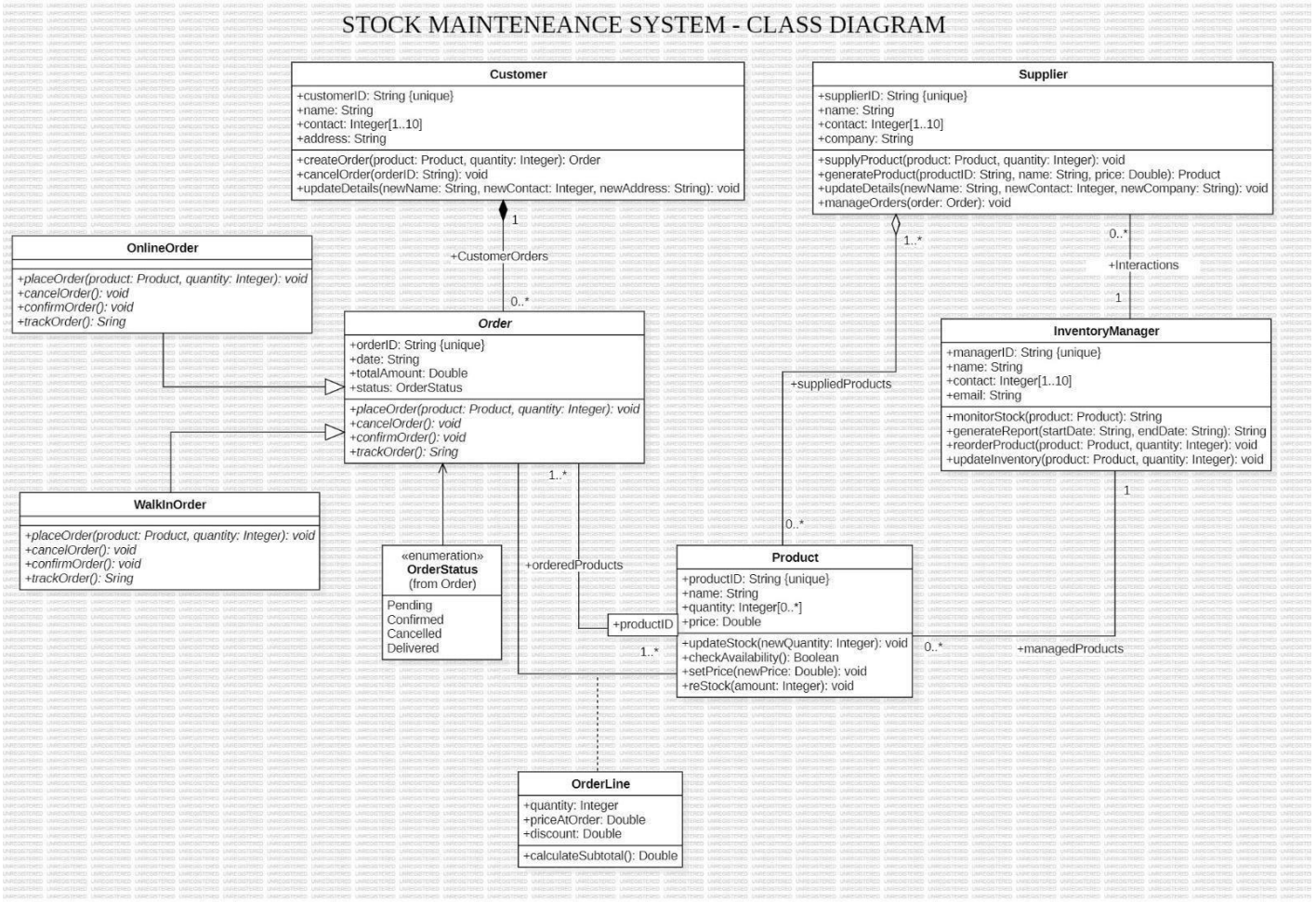


Fig 4.3

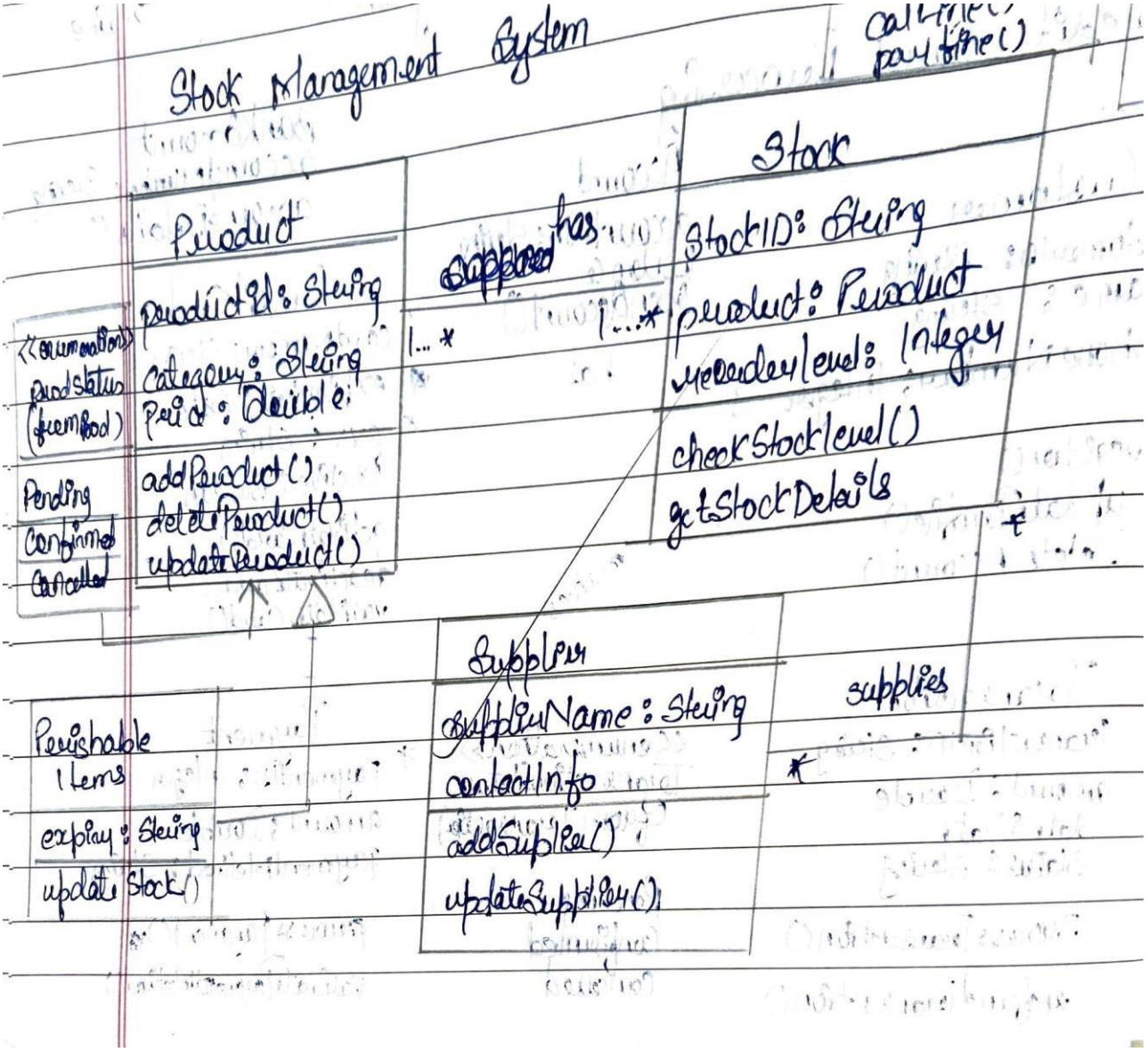

Stock Management System
Product
productId: String
category: String
Price: Double
addProduct()
deleteProduct()
updateProduct()
<<enumeration>>
ProdStatus
(from Prod)
Pending
Confirmed
Cancelled
has
1..*
1..*
Stock
StockID: String
product: Product
reorderLevel: Integer
checkStockLevel()
getStockDetails
Supplier
SupplierName: String
contactInfo
addSupplier()
updateSupplier()
supplies
*
PerishableItems
expiry: String
updateStock()

Fig 4.4

Brief descriptions:

1. **Customer:** Represents a customer with attributes like customer ID, name, contact, and address.
2. **Supplier:** Represents a supplier with attributes like supplier ID, name, contact, and company.
3. **Product:** Represents a product with attributes like product ID, name, price, and quantity.
4. **Order:** Represents an order with attributes like order ID, date, total amount, and status.
5. **Online Order:** Represents an online order, inheriting from the Order class.
6. **WalkinOrder:** Represents a walk-in order, inheriting from the Order class.
7. **Inventory Manager:** Represents the manager responsible for managing inventory, with methods for monitoring stock, generating reports, and reordering products.
8. **OrderLine:** Represents a line item within an order with details like product, quantity, price, and discount.
9. **Interactions:** Represents interactions between customers, suppliers, and the inventory manager.

SIMPLE STATE DIAGRAM

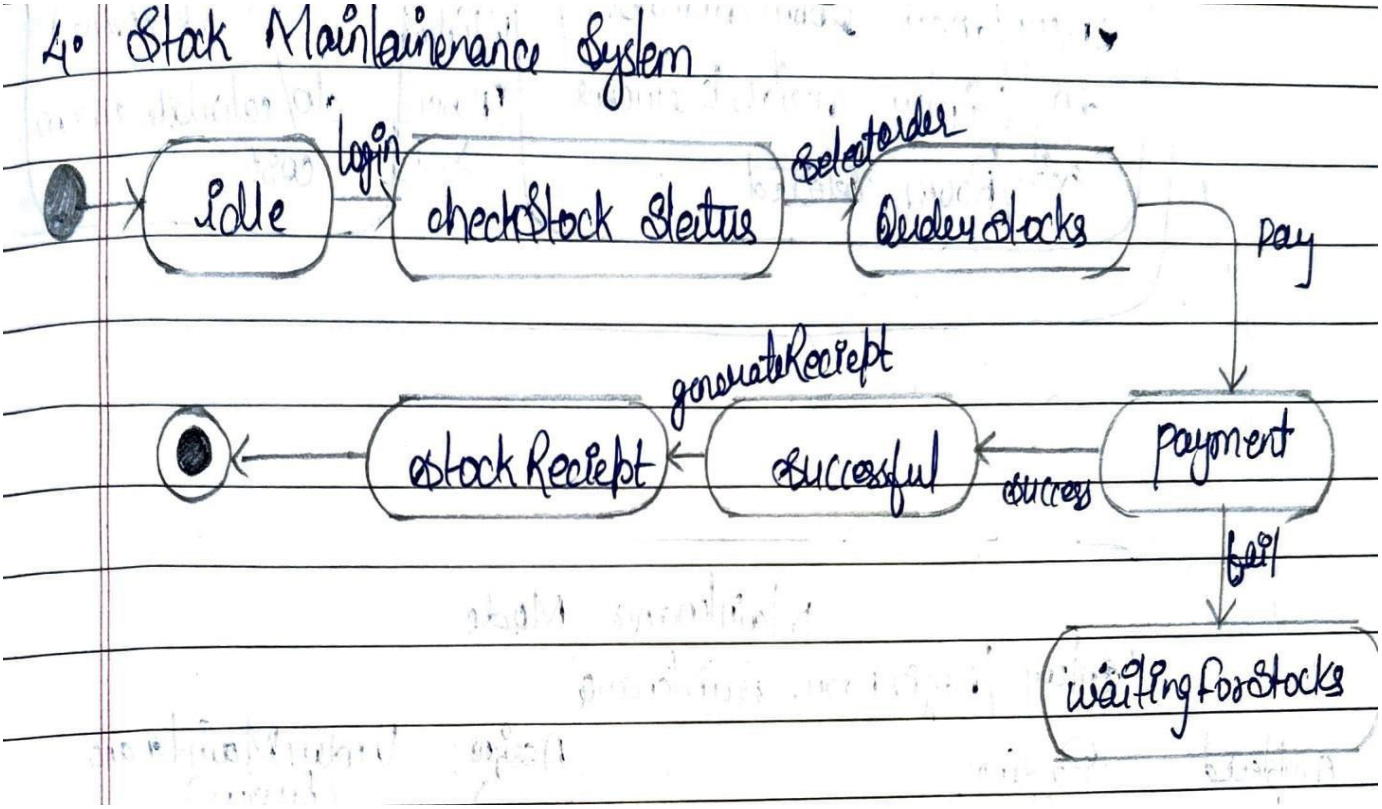


Fig 4.5

ADVANCE STATE DIAGRAM

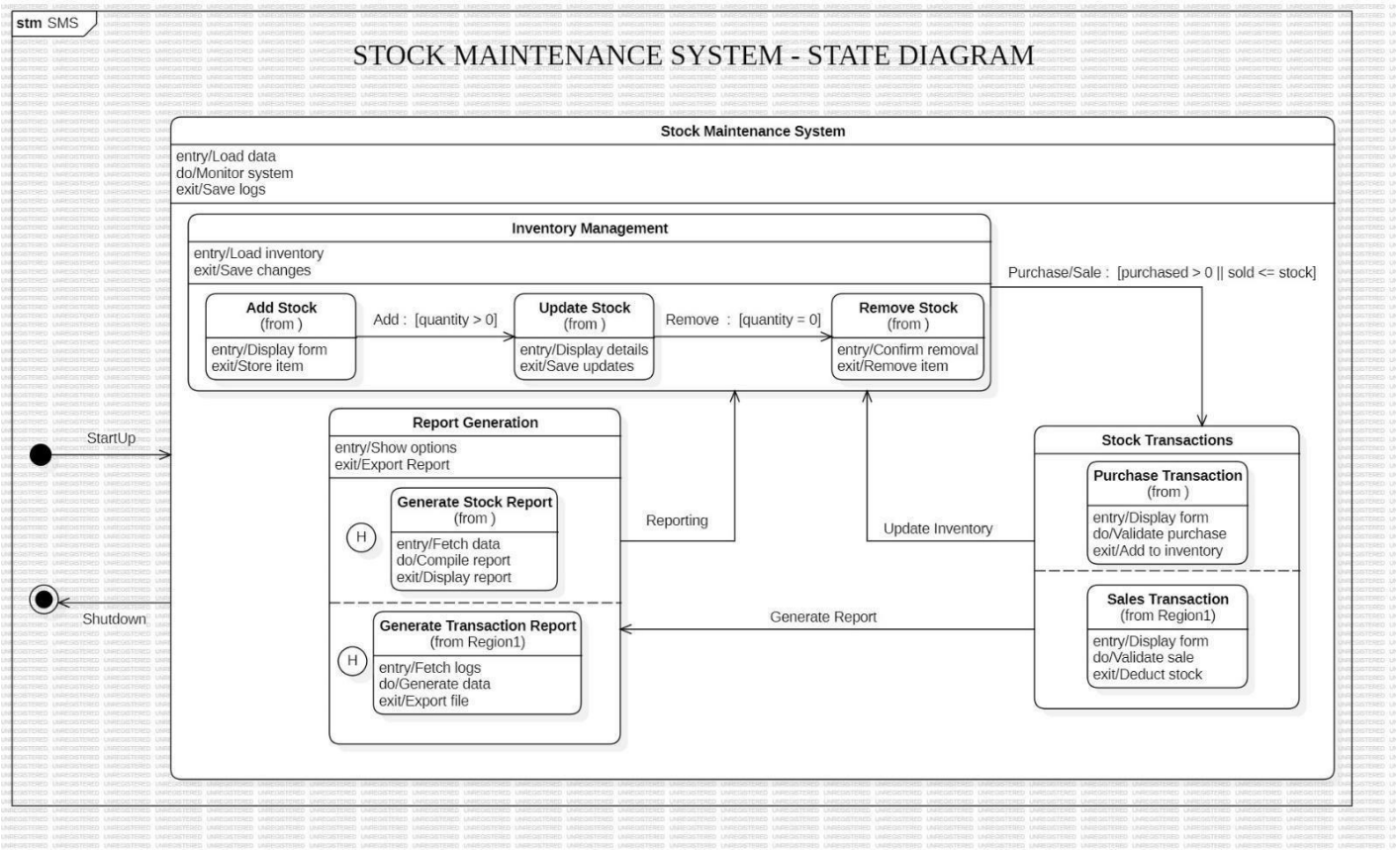


Fig 4.6

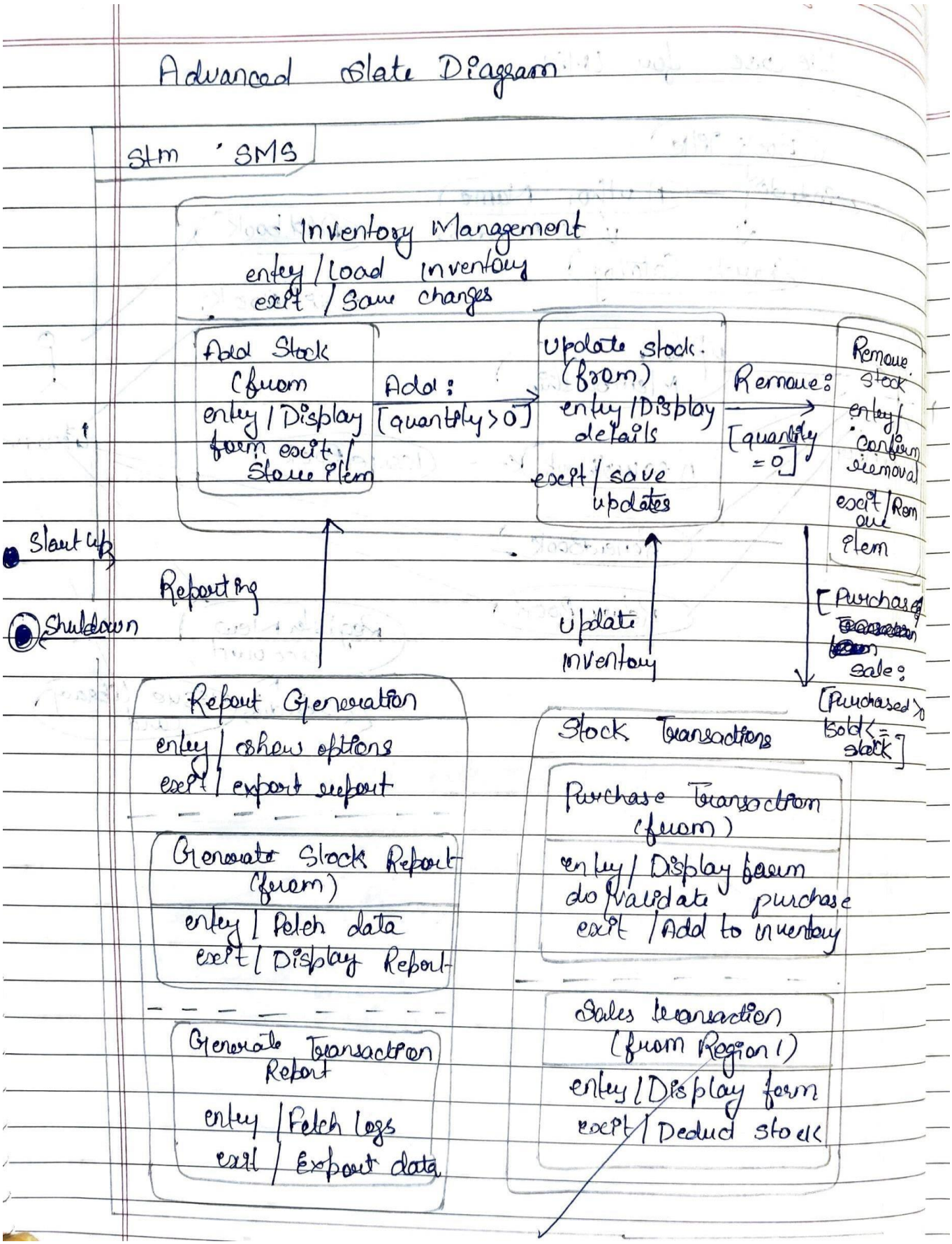

Advanced State Diagram
Stm 'SMS
Inventory Management
entry / Load inventory
exit / Save changes
Add Stock (from
entry / Display form
exit / Store Item
Add: [quantity > 0]
Update stock. (from)
entry / Display details
exit / save updates
Remove: [quantity = 0]
Remove Stock
entry / confirm removal
exit / Remove Item
[Purchase
Sale:
[Purchased
Start up
Shutdown
Reporting
Update inventory
Report Generation
entry / show options
exit / export report
Generate Stock Report (from)
entry / Fetch data
exit / Display Report
Generate Transaction Report
entry / Fetch logs
exit / Export data
Stock Transactions
Purchase Transaction (from)
entry / Display form
do / validate purchase
exit / Add to inventory
Sales transaction (from Region 1)
entry / Display form
exit / Deduct stock

Fig 4.7

Brief Description:

- **StartUp:** System starts and loads data.
- **Shutdown:** The system shuts down and saves logs.
- **Inventory Management:** The system transitions to an inventory management state.
- **Add Stock:** Transitions to add stock state when new stock is added.
- **Update Stock:** Transitions to update stock state when existing stock levels are modified.
- **Remove Stock:** Transitions to remove stock state when stock is removed.
- **Report Generation:** Transitions to report generation state.
- **Generate Stock Report:** Transitions to generate stock report state.
- **Generate Transaction Report:** Transitions to generate transaction report state.
- **Stock Transactions:** Transitions to stock transactions state.
- **Purchase Transaction:** Transitions to purchase transaction state.
- **Sales Transaction:** Transitions to sales transaction state.

USE CASE DIAGRAM

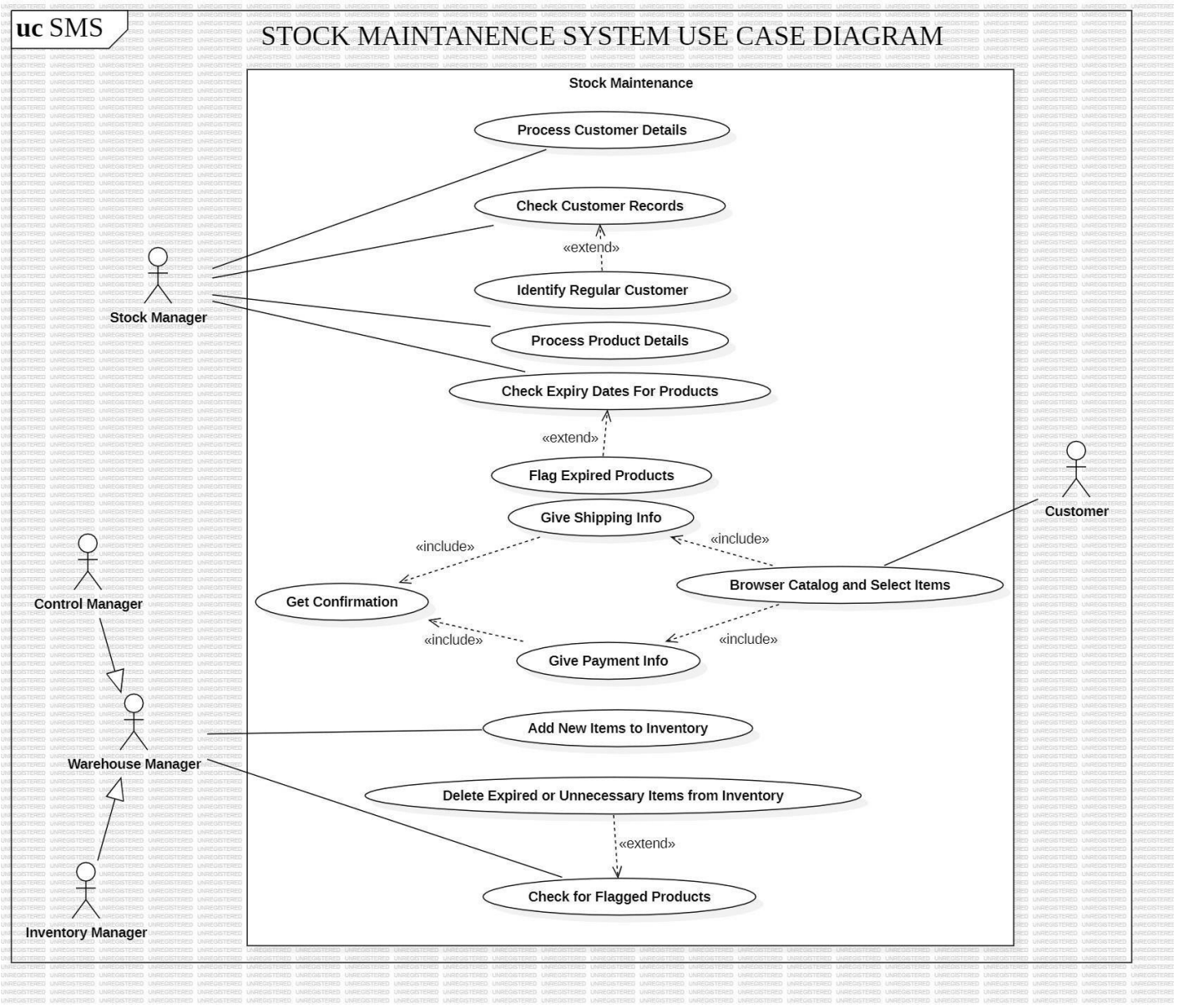


Fig 4.8

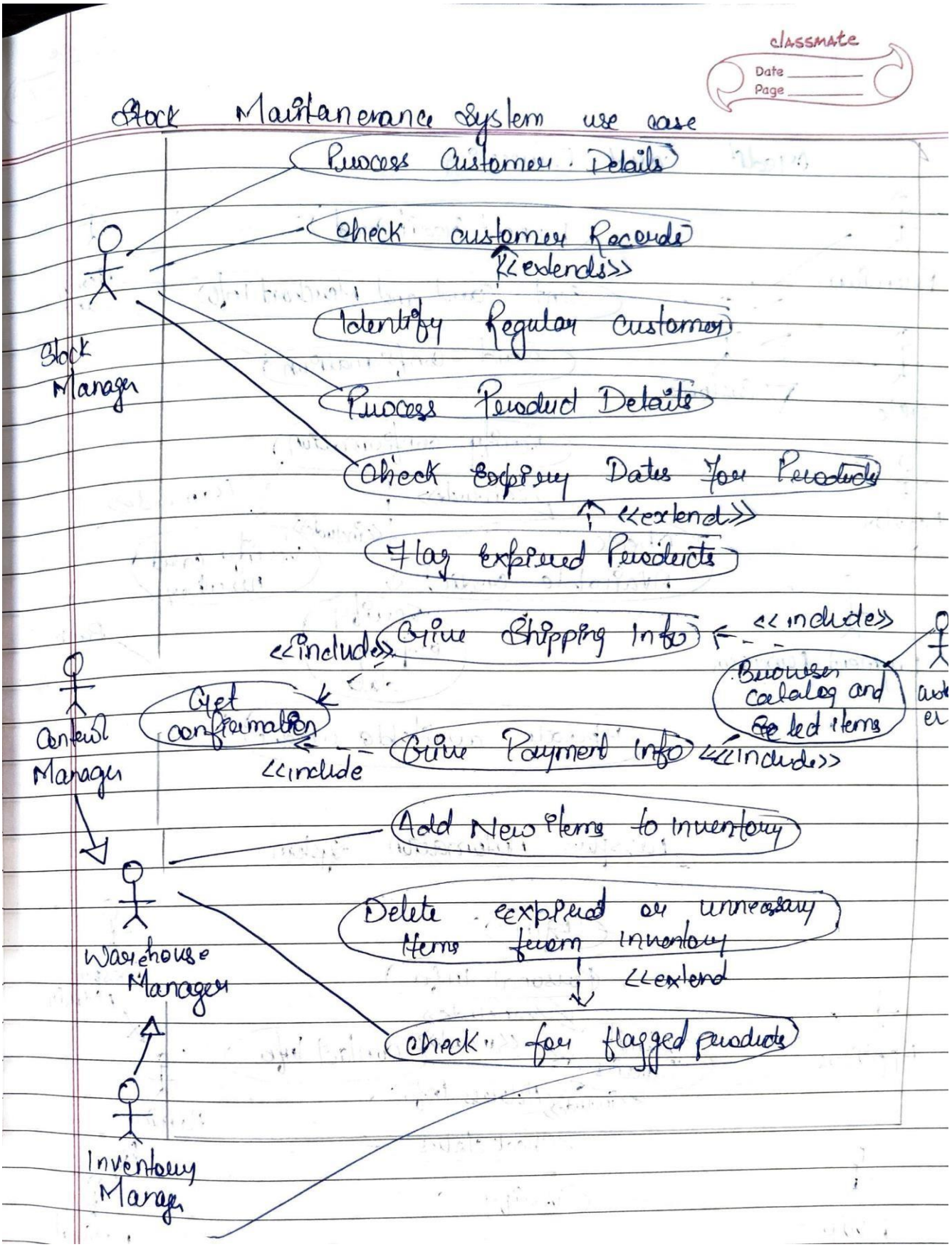

Stock Maintanenance System use case
Process Customer Details
Check customer Records
<<extends>>
Identify Regular Customer
Process Product Details
Check Expiry Dates for Products
<<extend>>
Flag expired Products
Give Shipping Info
<<include>>
<<include>>
Browses catalog and selected items
Get conformation
<<include
Give Payment Info
<<include>>
Add New Items to Inventory
Delete expired or unnecessary Items from inventory
<<extend
Check for flagged products
Stock Manager
Account Manager
Warehouse Manager
Inventory Manager

Fig 4.9

Brief descriptions:

- **Stock Maintenance:** The overarching use case encompassing all stock-related activities.
- **Process Customer Details:** Use case for managing customer information.
- **Check Customer Records:** Use case for verifying customer details.
- **Identify Regular Customer:** An extension of Check Customer Records, used to identify frequent customers.
- **Process Product Details:** Use case for managing product information.
- **Check Expiry Dates For Products:** Use case for checking and tracking product expiration dates.
- **Flag Expired Products:** An extension of Check Expiry Dates used to flag expired products.
- **Get Confirmation:** Use case for obtaining confirmation from relevant parties (e.g., Warehouse Manager) on various actions.
- **Give Shipping Info:** Use case related to providing shipping information.
- **Give Payment Info:** Use case related to managing payment information.
- **Browser Catalog and Select Items:** Use cases for customers to browse the product catalogue and select items.
- **Add New Items to Inventory:** Use case for adding new products to inventory.
- **Delete Expired or Unnecessary Items from Inventory:** Use case for removing expired or obsolete products from inventory.
- **Check for Flagged Products:** An extension of Delete Expired or Unnecessary Items, used to specifically check for and remove flagged products.

SEQUENCE DIAGRAM

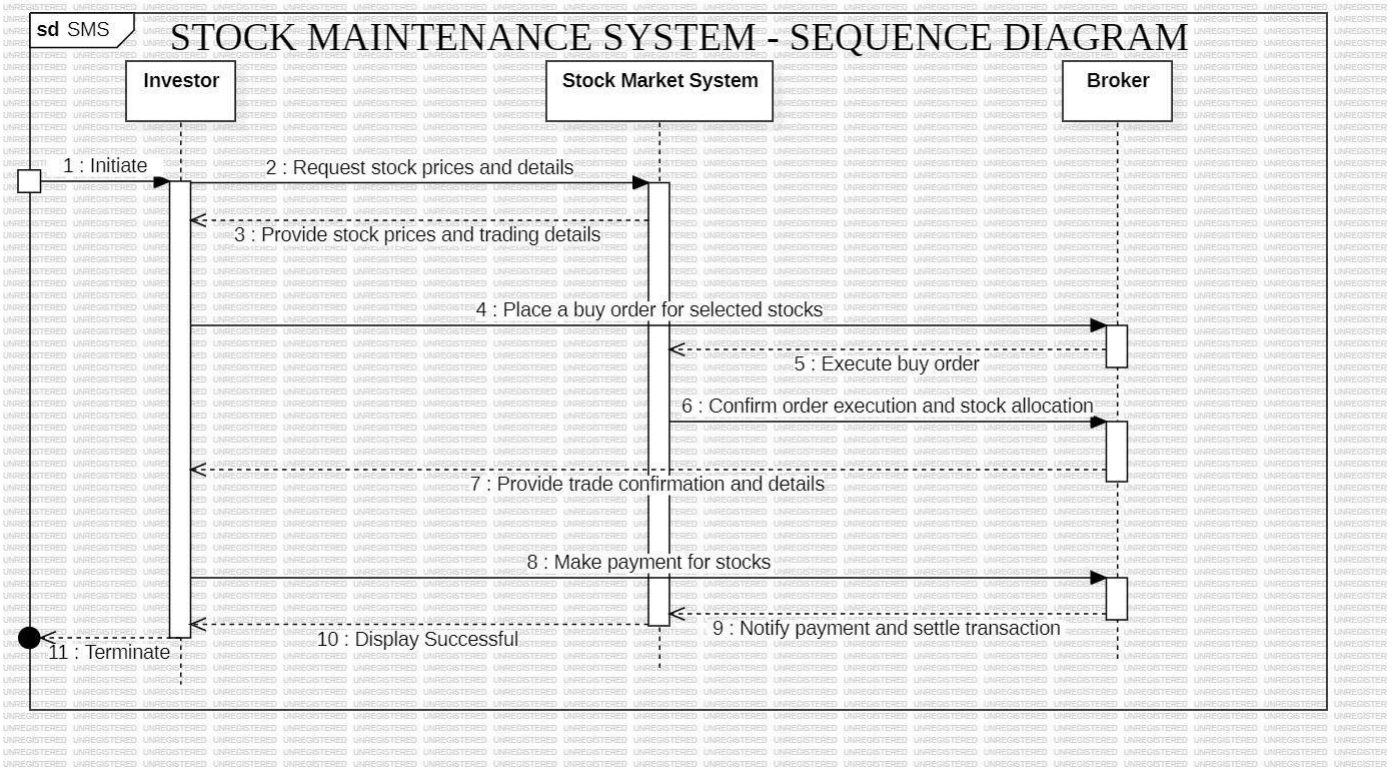


Fig 4.10

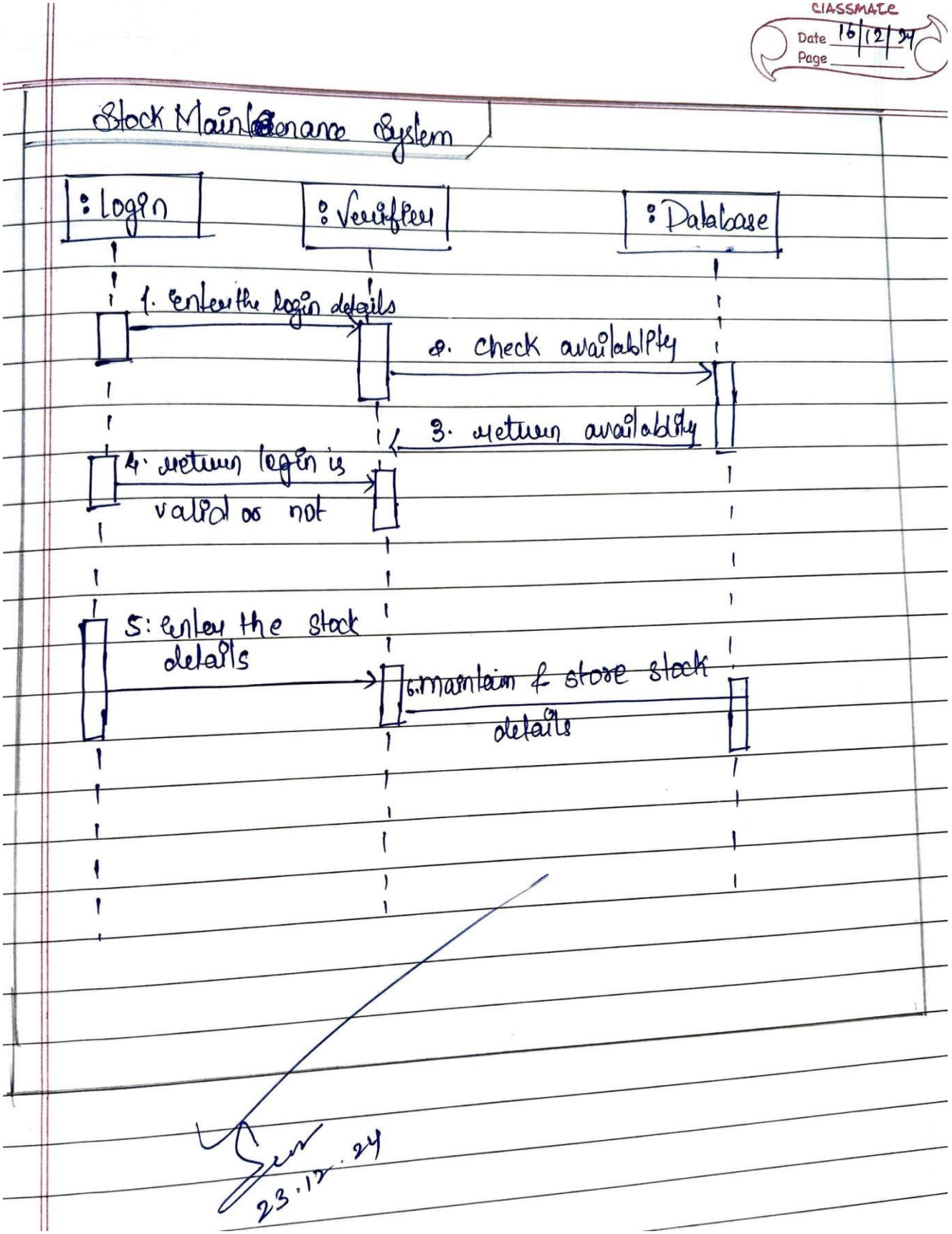

Stock Maintenance System
: Login
: Verifier
: Database
1. Enter the login details
2. check availability
3. return availability
4. return login is valid or not
5: Enter the stock details
6. maintain & store stock details
23.12.24

Fig 4.11

Brief descriptions:

1. **Initiate:** The investor initiates the stock trading process.
2. **Request stock prices and details:** The investor requests stock prices and trading details from the Stock Market System.
3. **Provide stock prices and trading details:** The Stock Market System provides the requested information to the investor.
4. **Place a buy order for selected stocks:** The investor places a buy order for selected stocks with the Stock Market System.
5. **Execute buy order:** The Stock Market System executes the buy order and allocates the stocks.
6. **Confirm order execution and stock allocation:** The Stock Market System confirms the order execution and stock allocation to the investor.
7. **Provide trade confirmation and details:** The Stock Market System provides trade confirmation details to the investor.
8. **Make payment for stocks:** The investor makes payment for the stocks.
9. **Notify payment and settle transaction:** The Stock Market System notifies the broker about the payment and settles the transaction.
10. **Display Successful:** The Stock Market System displays a successful transaction message to the investor.
11. **Terminate:** The trading process is completed.

ACTIVITY DIAGRAM

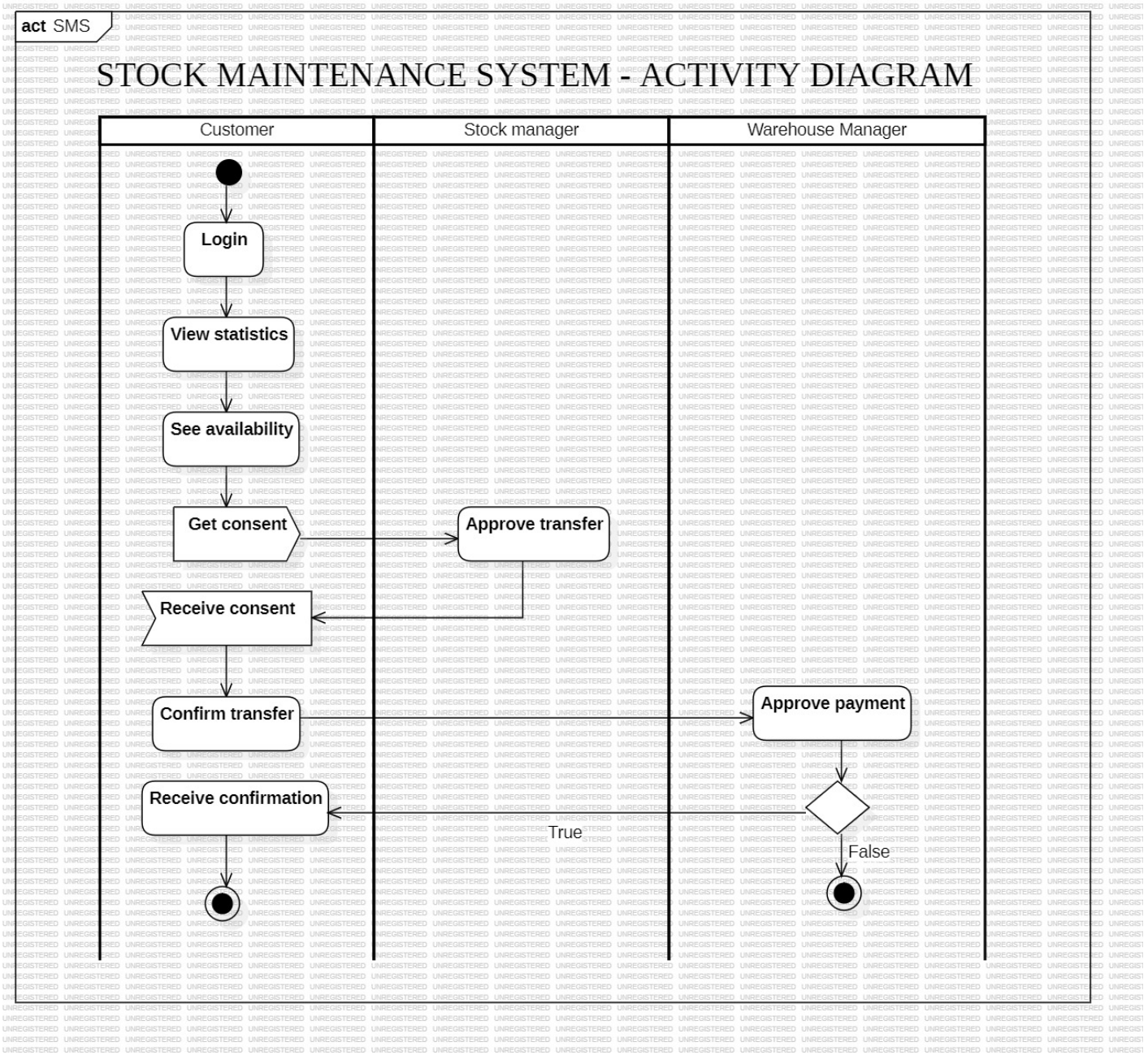


Fig 4.12

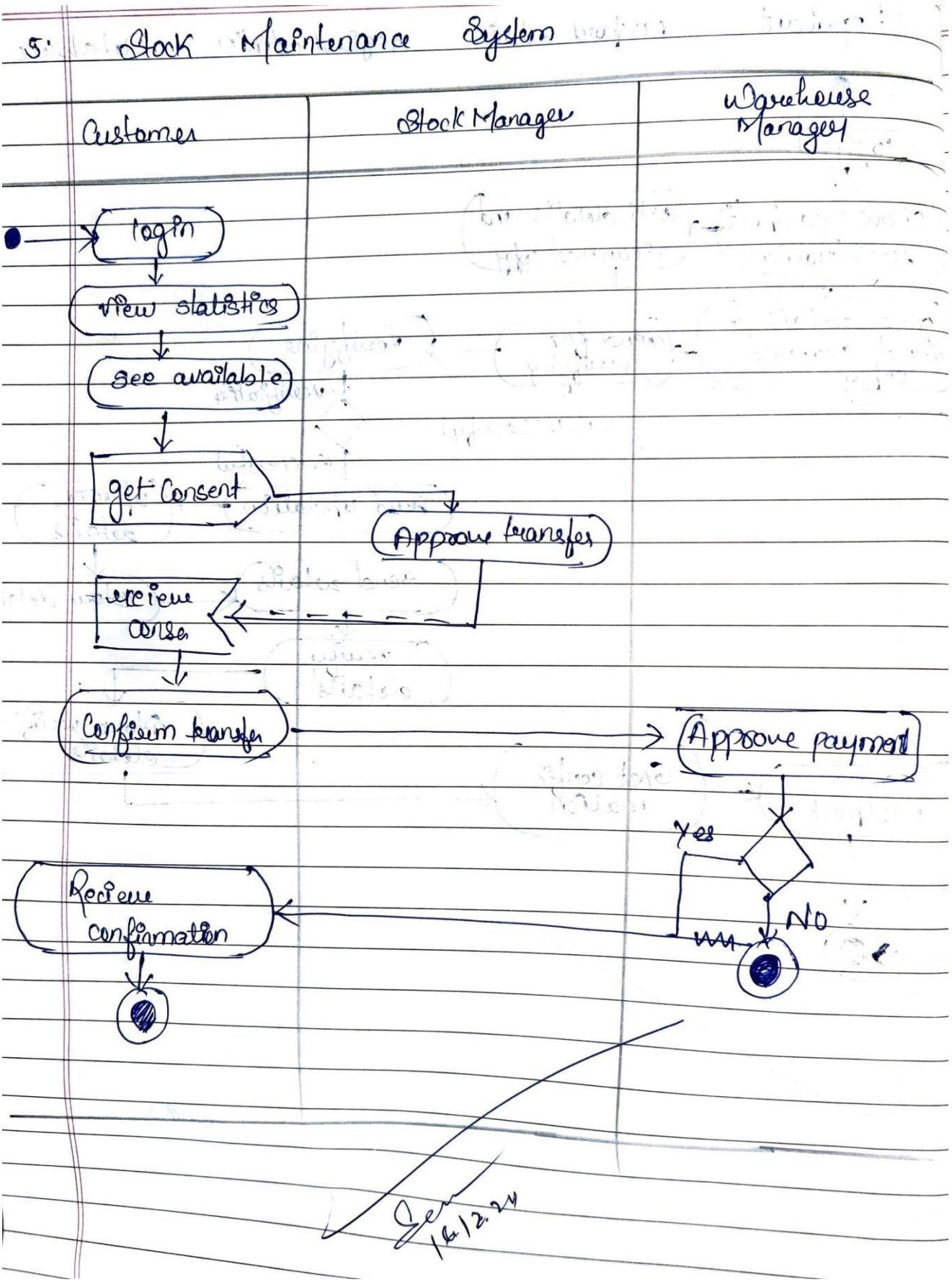

5. Stock Maintenance System
Customer
Stock Manager
Warehouse Manager
login
view statistics
see available
get Consent
Approve transfer
recieve confirm
Confirm transfer
Approve payment
Yes
No
Recieve confirmation
16/12/24

Fig 4.13

Brief descriptions:

Customer:

- **Login:** The customer logs into the system.
- **View statistics:** The customer views stock availability statistics.
- **See availability:** The customer checks the availability of a particular product.
- **Get consent:** The customer requests consent from the Warehouse Manager for a transfer.
- **Receive consent:** The customer receives confirmation from the Warehouse Manager.
- **Confirm transfer:** The customer confirms the transfer.
- **Receive confirmation:** The customer receives confirmation of the completed transfer.

Stock Manager:

- **Approve transfer:** The Stock Manager approves the transfer request from the customer.

Warehouse Manager:

- **Approve payment:** The Warehouse Manager approves the payment for the transfer.

PASSPORT AUTOMATION SYSTEM

Problem Statement:

Develop a robust and secure Password Automation System to streamline user account management and enhance security for organisations. The system should:

- **Generate Strong Passwords:** Create complex passwords with a mix of characters meeting security best practices.
- **Unique Password Generation:** Ensure each user and application has a unique password.
- **Secure Password Storage:** Store passwords securely using robust encryption methods.
- **Seamless Integrations:** Integrate with existing IAM and other relevant systems.
- **Robust Auditing:** Track all password activities for security and compliance.
- **User-Friendly Interface:** Provide an intuitive interface for easy password management.
- **Compliance Focus:** Adhere to industry standards and relevant regulations.

classmate
Date 23/1/24
Page

Software Requirement Specification (SRS)

1. Introduction

1.1 Purpose of this Document

The SRS document outlines the detailed requirements for Hotel Management System. It serves as blue print for the development team, ensuring final product meets the specified specification and fulfills the customer requirements.

1.2 Scope of this document

It aims to provide clear specifications for the system's development, including overview of costs and timelines involved. This HMS (Hotel Management System) is designed to ~~provide~~ streamline hotel operations and enhance customer experience.

1.3 Overview

The HMS will provide solution for managing hotel operations including reservations, customer management, billings and reportings. This will enable staff to handle daily tasks and offer user friendly interface for bookings.

2. General Description

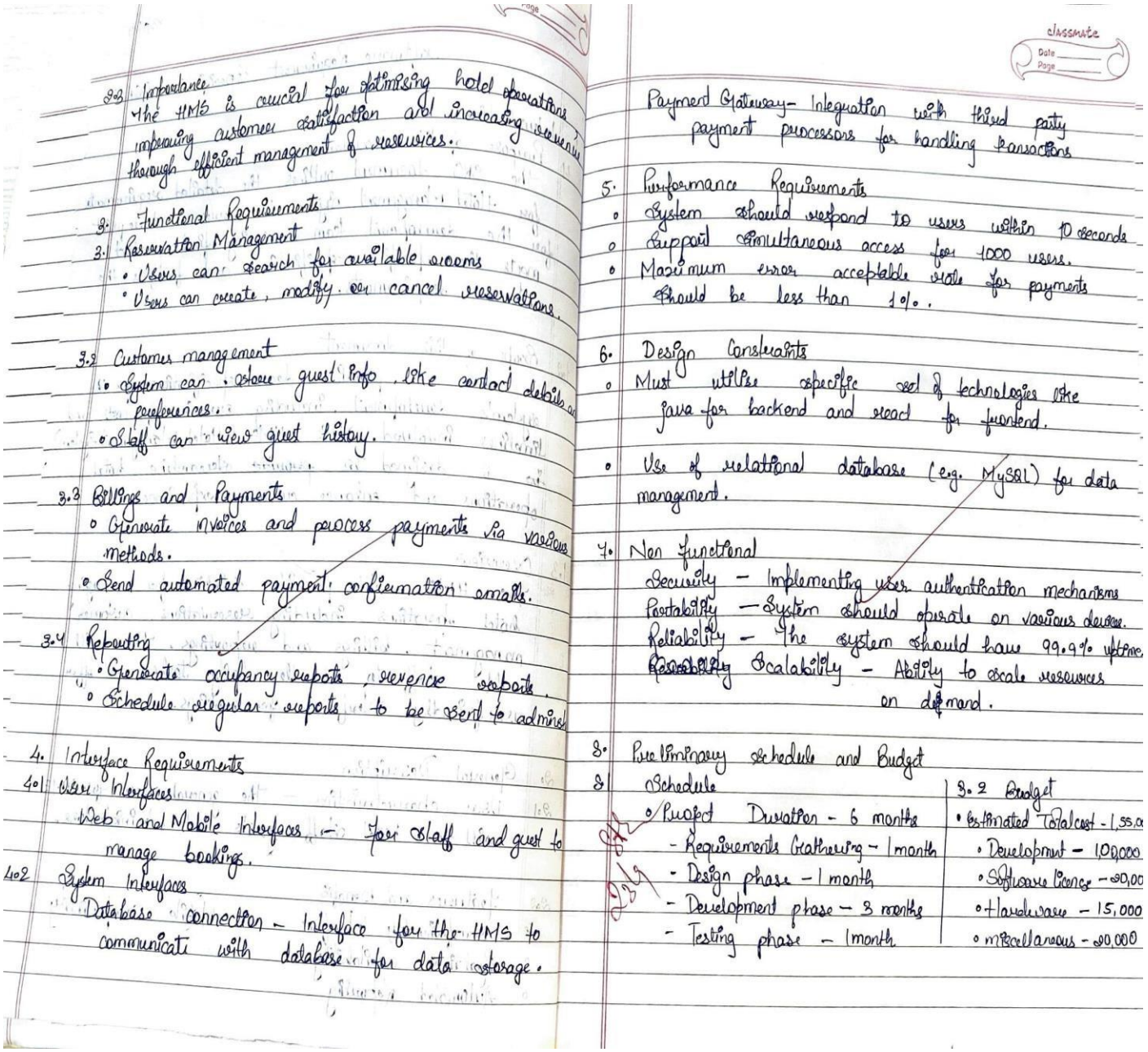
2.1 User characteristics — The general user for HMS will be hotel staff, guests and administrators.

2.2 Features and Benefits

- User-friendly Interface
- Mobile Compatibility
- Real time availability
- Automated Reporting

Fig 5.1

MENTS SPECIFICATION



2.3 Importance
The HMS is crucial for optimising hotel operations, improving customer satisfaction and increasing revenue through efficient management of resources.

3. Functional Requirements
3.1 Reservation Management
- Users can search for available rooms
- Users can create, modify or cancel reservations.

3.2 Customer management
- System can store guest info like contact details, preferences.
- Staff can view guest history.

3.3 Billings and Payments
- Generate invoices and process payments via various methods.
- Send automated payment confirmation emails.

3.4 Reporting
- Generate occupancy reports, revenue reports.
- Schedule regular reports to be sent to admins.

4. Interface Requirements
4.1 User Interfaces
Web and Mobile Interfaces - for staff and guest to manage bookings.
4.2 System Interfaces
Database connection - Interface for the HMS to communicate with database for data storage.
Payment Gateways - Integration with third party payment processors for handling transactions

5. Performance Requirements
- System should respond to users within 10 seconds
- Support simultaneous access for 1000 users.
- Maximum error acceptable rate for payments should be less than 1%.

6. Design Constraints
- Must utilise specific set of technologies like java for backend and react for frontend.
- Use of relational database (e.g. MySQL) for data management.

7. Non functional
Security - Implementing user authentication mechanisms
Portability - System should operate on various devices.
Reliability - The system should have 99.9% uptime.
Scalability - Ability to scale resources on demand.

8. Preliminary Schedule and Budget
8.1 Schedule
- Project Duration - 6 months
  - Requirements Gathering - 1 month
  - Design phase - 1 month
  - Development phase - 3 months
  - Testing phase - 1 month

8.2 Budget
- Estimated Total cost - 1,55,0
- Development - 1,00,000
- Software licence - 20,00
- Hardware - 15,000
- Miscellaneous - 20,000

Fig 5.2

UML DIAGRAMS

CLASS DIAGRAM

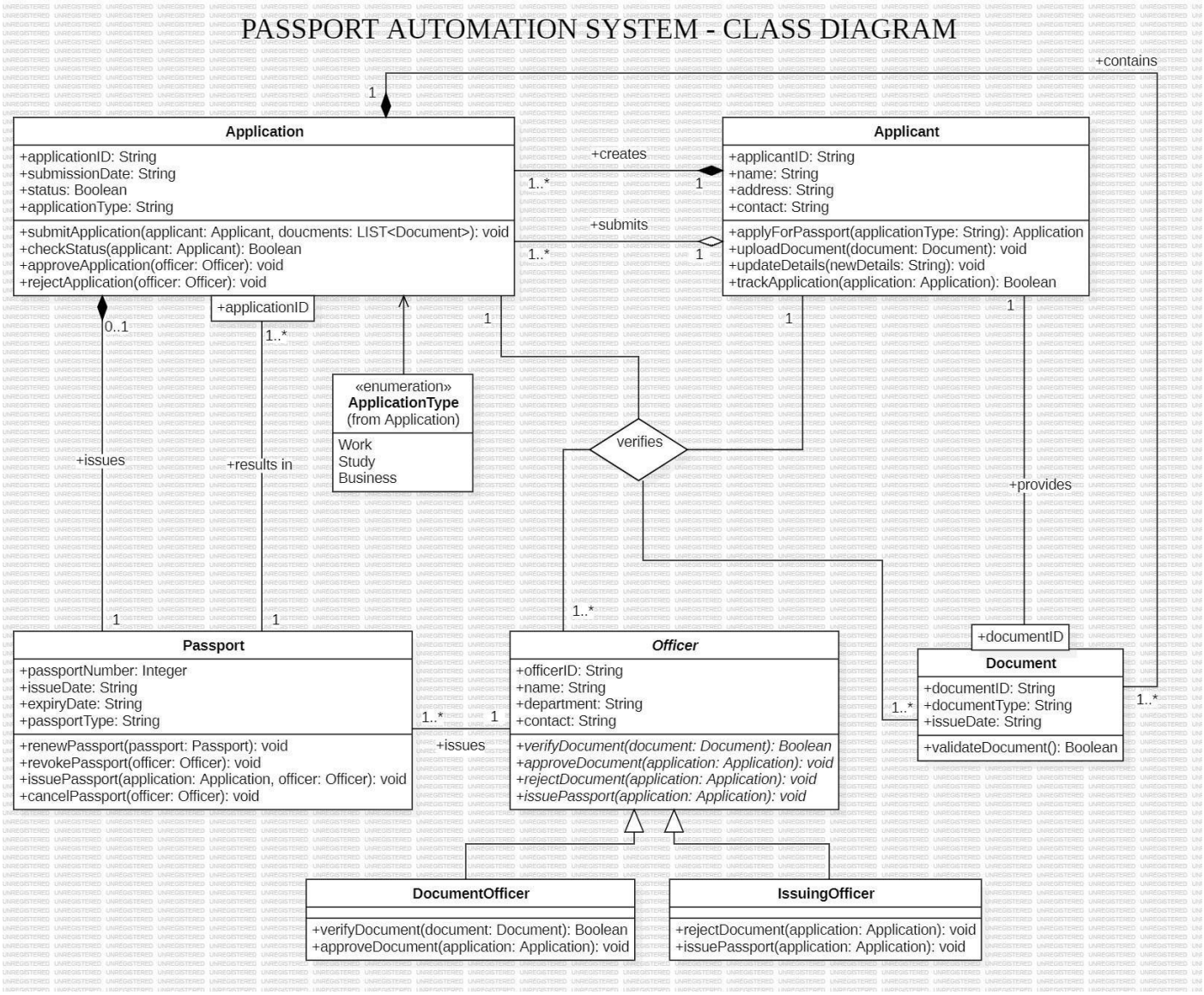


Fig 5.3

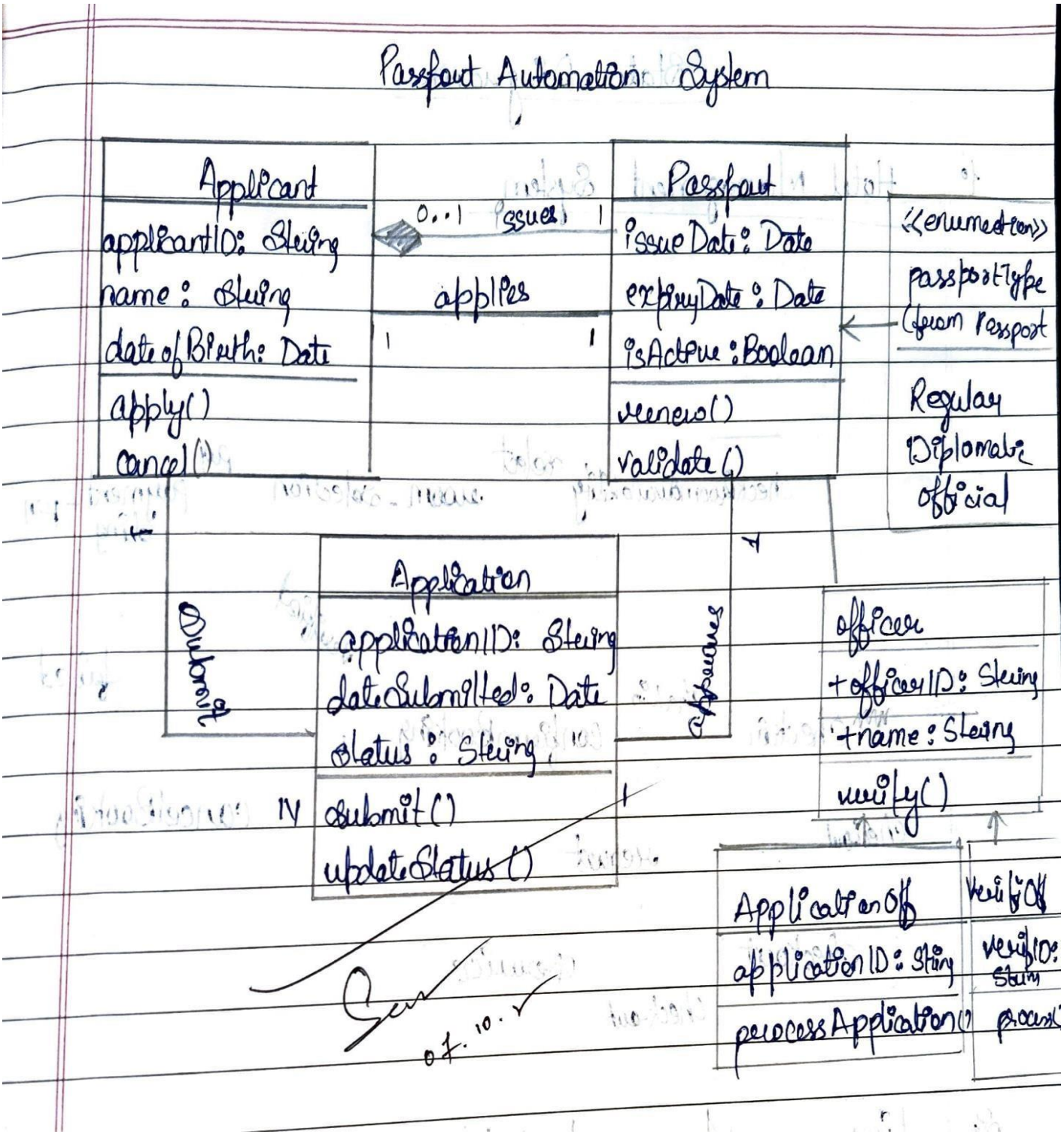

Passport Automation System
Applicant
applicantID: String
name: String
dateOfBirth: Date
apply()
cancel()
0..1
issues
1
applies
1
1
Passport
issueDate: Date
expiryDate: Date
isActive: Boolean
renew()
validate()
<<enumeration>>
passportType
(from Passport)
Regular
Diplomatic
official
Application
applicationID: String
dateSubmitted: Date
status: String
submit()
updateStatus()
Submits
Approves
officer
+officerID: String
+name: String
verify()
ApplicationOff
applicationID: String
processApplication()
VerifiOff
verifID:
String
proces

Fig 5.4

Brief Description:

1. **Applicant:** Represents the individual applying for a passport, holding their personal information and application-related methods.
2. **Application:** Represents a passport application with details like status, type, and submission date, including methods for submission, status checks, and approvals.
3. **Passport:** Represents the issued passport document with information like number, issue/expiry dates, and methods for renewal, revocation, and cancellation.
4. **Officer:** Represents a passport officer with information like ID, name, and department, responsible for verifying documents, approving/rejecting applications, and issuing passports.
5. **Document:** Represents a document submitted with the application, storing details like ID, type, and issue date and a validation method.
6. **DocumentOfficer:** Specializes in verifying documents, inheriting from the Officer class with additional verification methods.
7. **Issuing Officer:** Specializes in issuing passports, inheriting from the Officer class with additional issuance methods.
8. **ApplicationType:** An enumeration defining the different types of passport applications (e.g., work, study, business).

SIMPLE STATE DIAGRAM

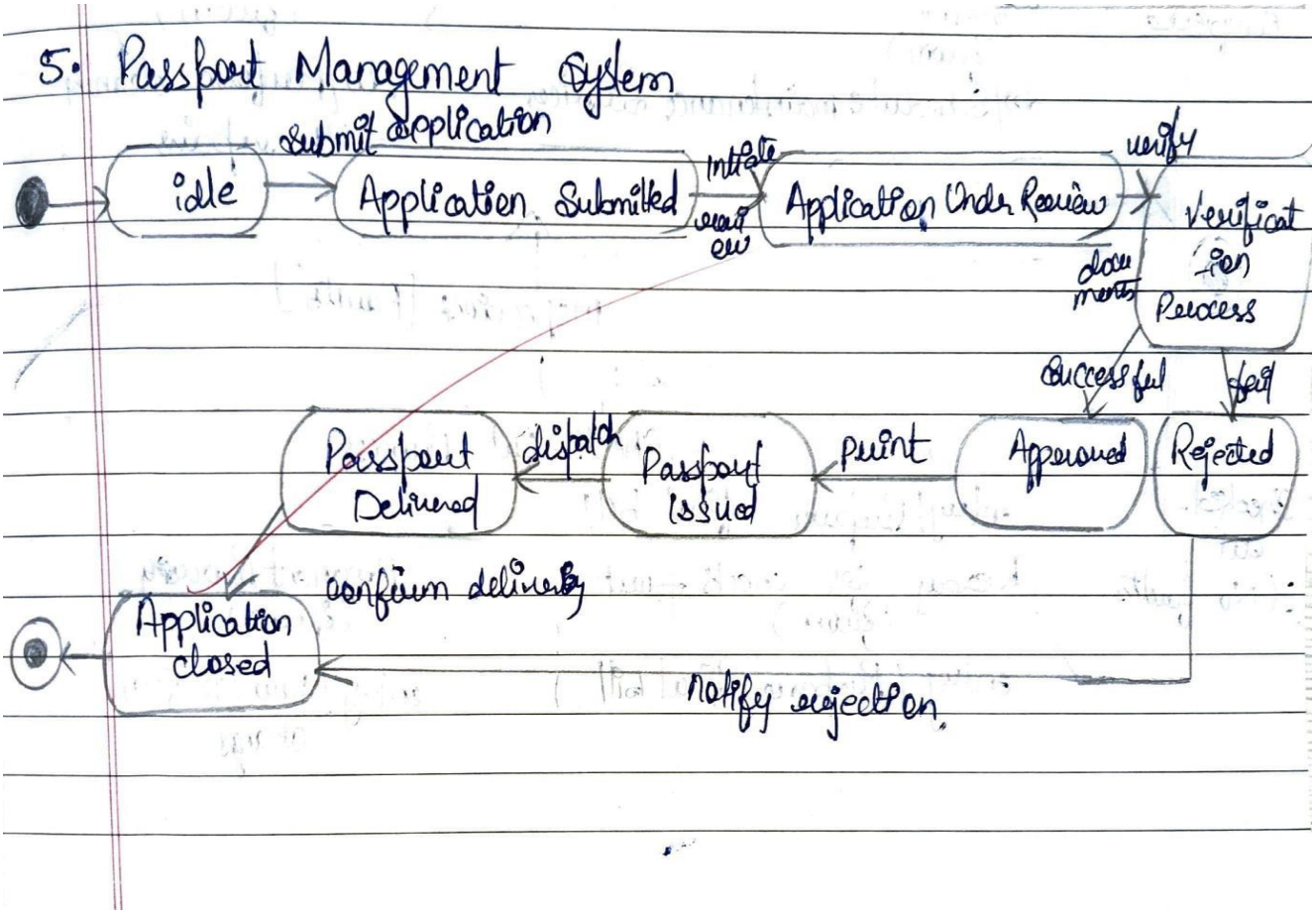


Fig 5.5

ADVANCE STATE DIAGRAM

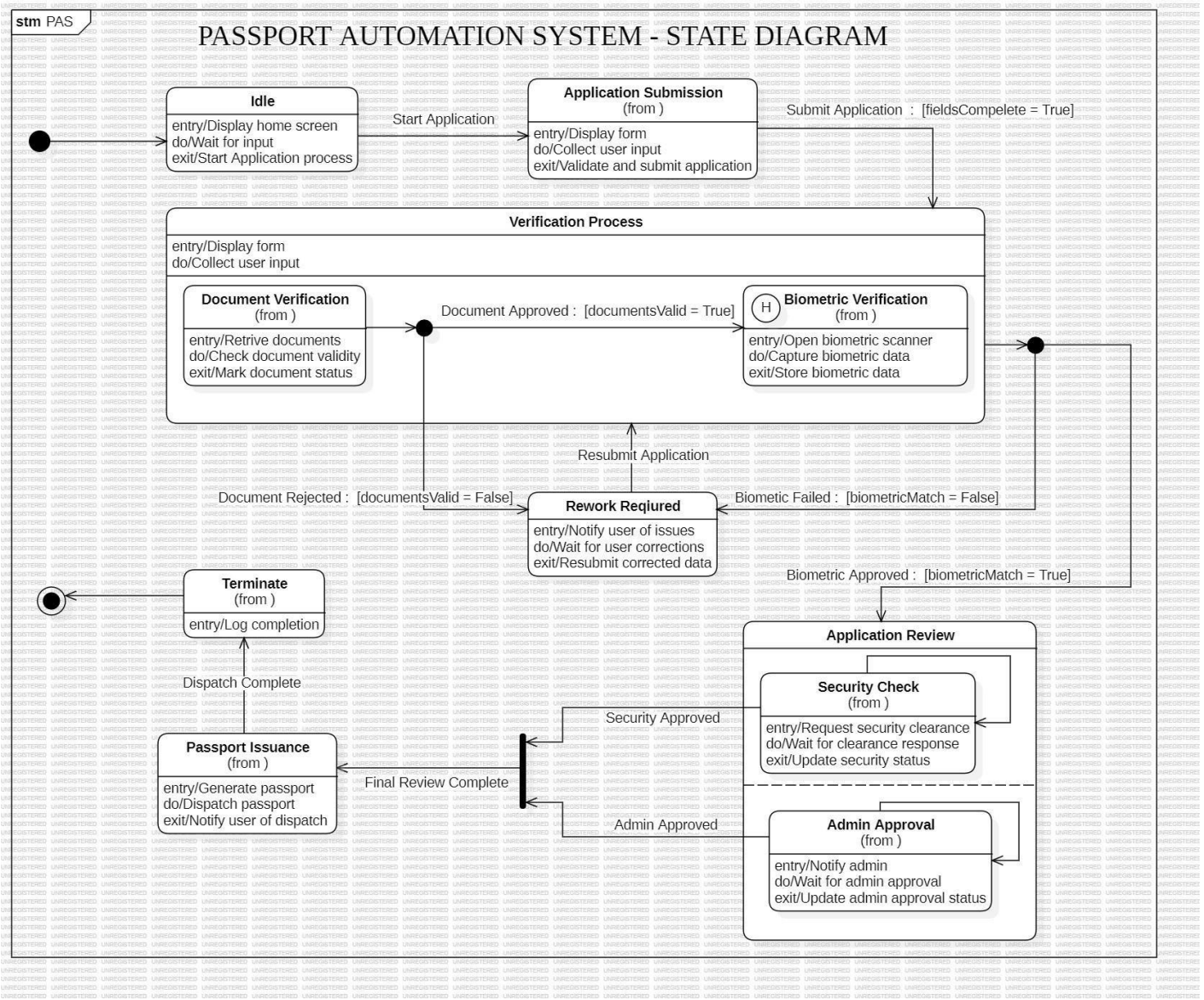


Fig 5.6

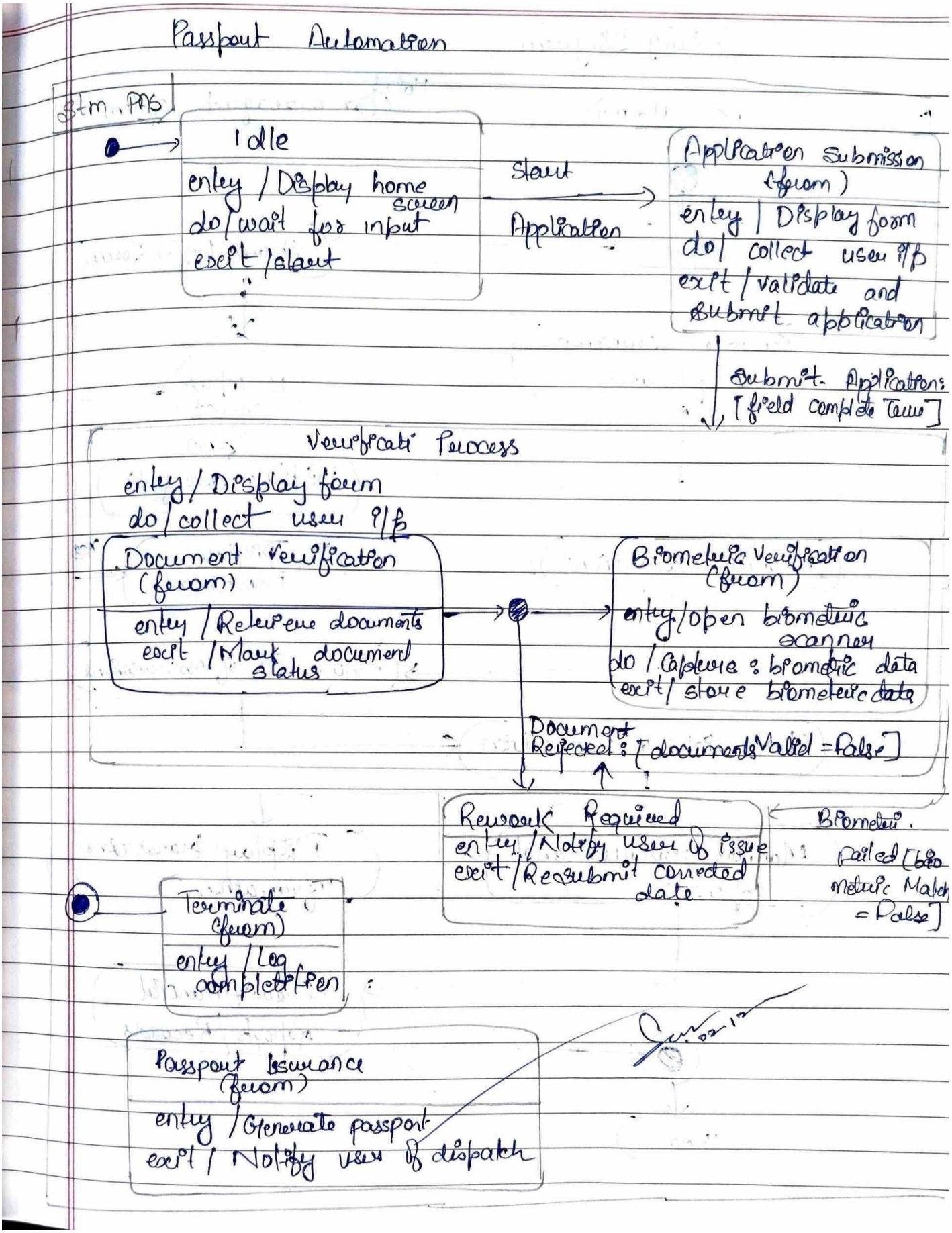

Passport Automation
Stm. PAS
Idle
entry / Display home screen
do / wait for input
exit / start
start Application
Application Submission (form)
entry / Display form
do / collect user i/p
exit / validate and submit application
Submit. Application: [field complete Tcue]
Verificati Process
entry / Display form
do / collect user i/p
Document Verification (form)
entry / Retrieve documents
exit / Mark document status
Biometric Verification (form)
entry / open biometric scanner
do / Capture : biometric data
exit / store biometric data
Document Rejected : [documentsValid = False]
Rework Required
entry / Notify user of issue
exit / Resubmit corrected date
Biometri. Failed [bio metric Match = False]
Terminate (form)
entry / Log completion
Passport Issuance (form)
entry / Generate passport
exit / Notify user of dispatch

Fig 5.7

Brief descriptions:

- **Start Application:** The system transitions from the Idle state to the Start Application state when the user initiates the application process.
- **Application Submission:** The system transitions from the Start Application state to the Application Submission state when the user submits the completed application form.
- **Document Verification:** The system transitions from the Application Submission state to the Document Verification state to begin checking the validity of the submitted documents.
- **Biometric Verification:** The system transitions from the Document Verification state to the Biometric Verification state if the documents are valid.
- **Rework Required:** The system transitions from the Document Verification or Biometric Verification state to the Rework Required state if the documents are invalid or the biometric data does not match.
- **Terminate:** The system transitions from the Rework Required state to the Terminate state if the user fails to correct the issues or resubmit the application.
- **Dispatch Complete:** The system transitions from the Admin Approval state to the Dispatch Complete state when the passport has been generated and dispatched to the user.
- **Application Review:** The system transitions from the Biometric Verification state to the Application Review state if the biometric data is valid.
- **Security Check:** The system transitions from the Application Review state to the Security Check state to perform security clearance checks.
- **Admin Approval:** The system transitions from the Security Check state to the Admin Approval state for final administrative review and approval.

USE CASE DIAGRAM

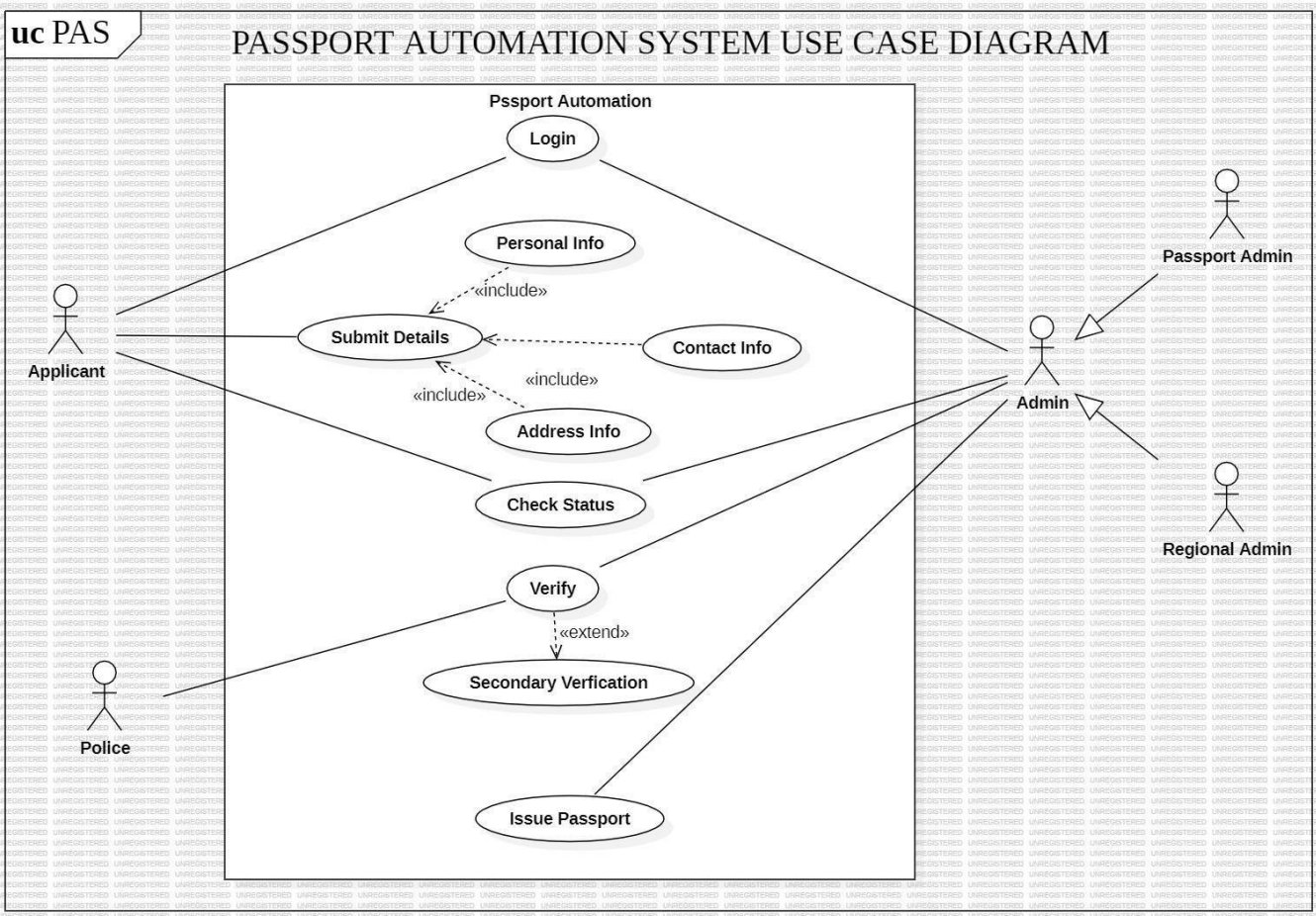


Fig 5.8

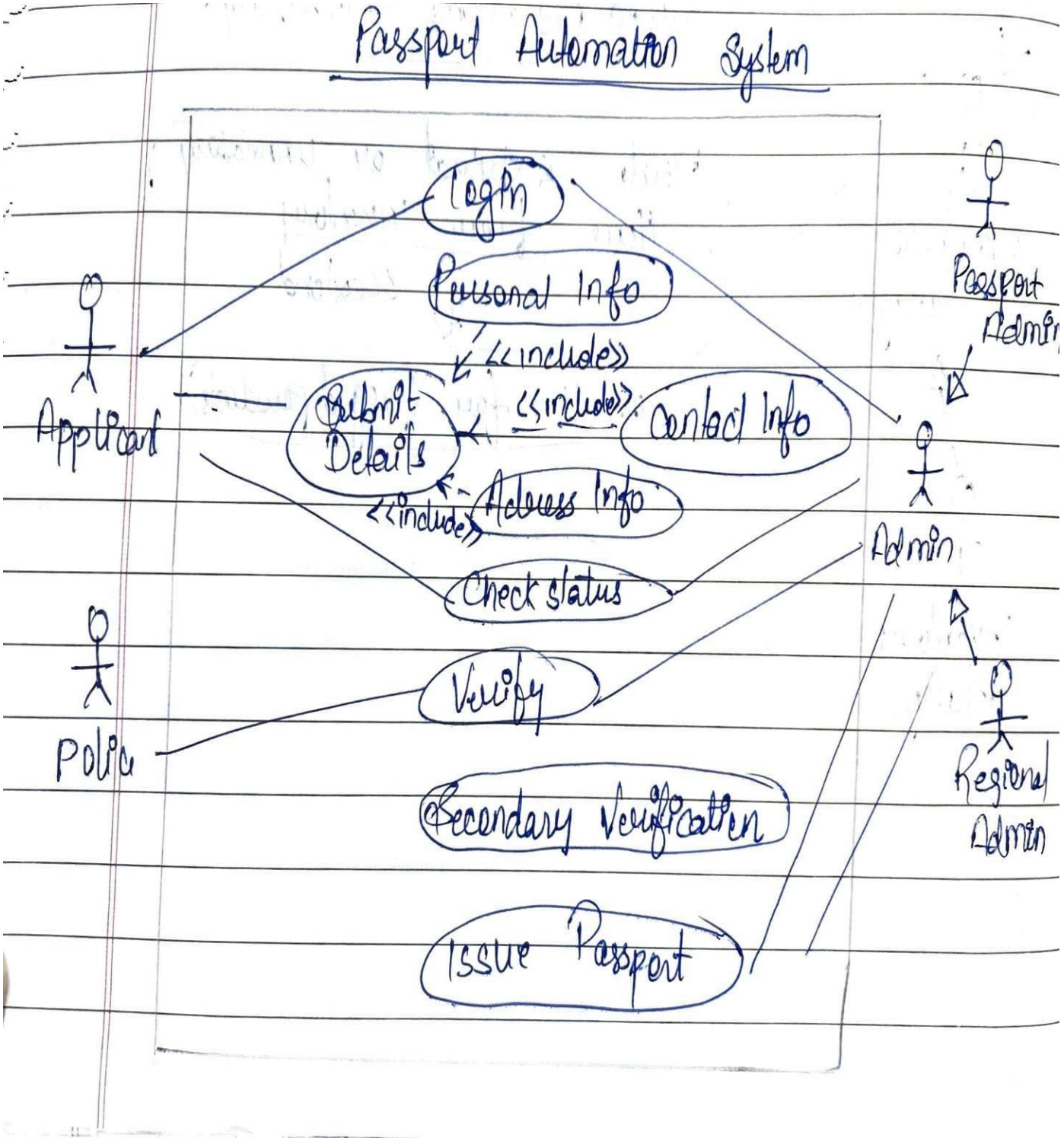

Passport Automation System
Login
Personal Info
<<include>>
Submit Details
<<include>>
Contact Info
<<include>>
Address Info
Check status
Verify
Secondary Verification
Issue Passport
Applicant
Police
Passport Admin
Admin
Regional Admin

Fig 5.9

Brief descriptions:

- **Login:** Allows users (Applicants, Admin, Passport Admin, Regional Admin) to authenticate and access the system.
- **Submit Details:** The core use case where applicants provide personal, contact, and address information for their passport application.
- **Personal Info:** Includes entering personal details like name, date of birth, and nationality.
- **Contact Info:** Includes entering contact details like phone number and email address.
- **Address Info:** Includes entering current and permanent address details.
- **Check Status:** Allows applicants to track the status of their passport application.
- **Verify:** The process of verifying the applicant's information and documents.
- **Secondary Verification:** An optional extension of the verification process, potentially involving police checks or background investigations.
- **Issue Passport:** The final step is where the passport is issued to the applicant.

SEQUENCE DIAGRAM

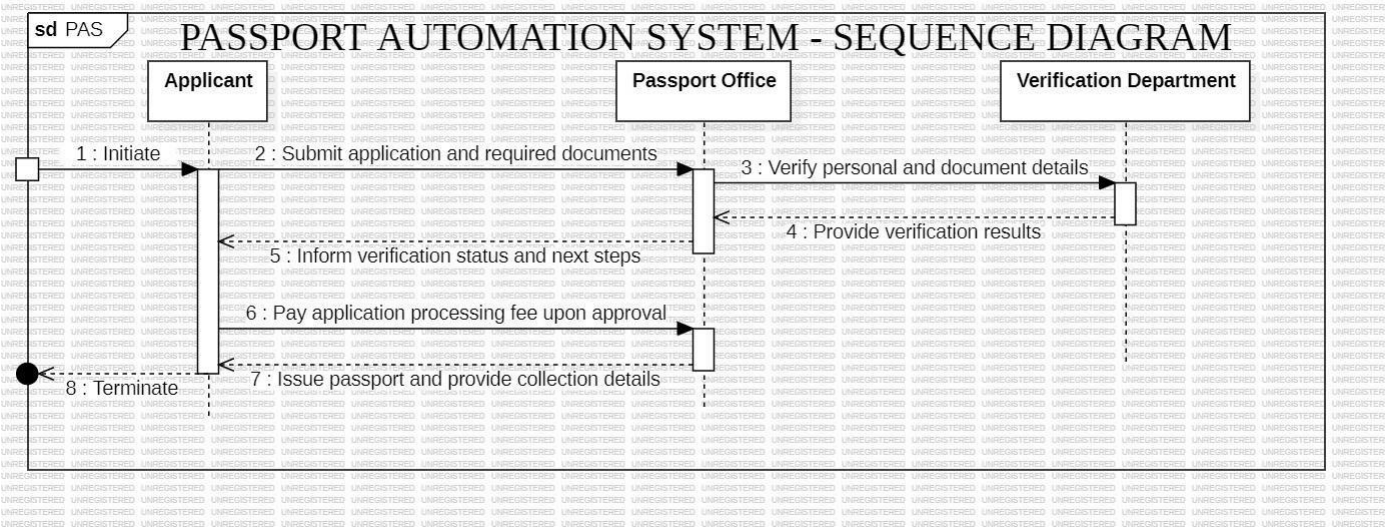


Fig 5.10

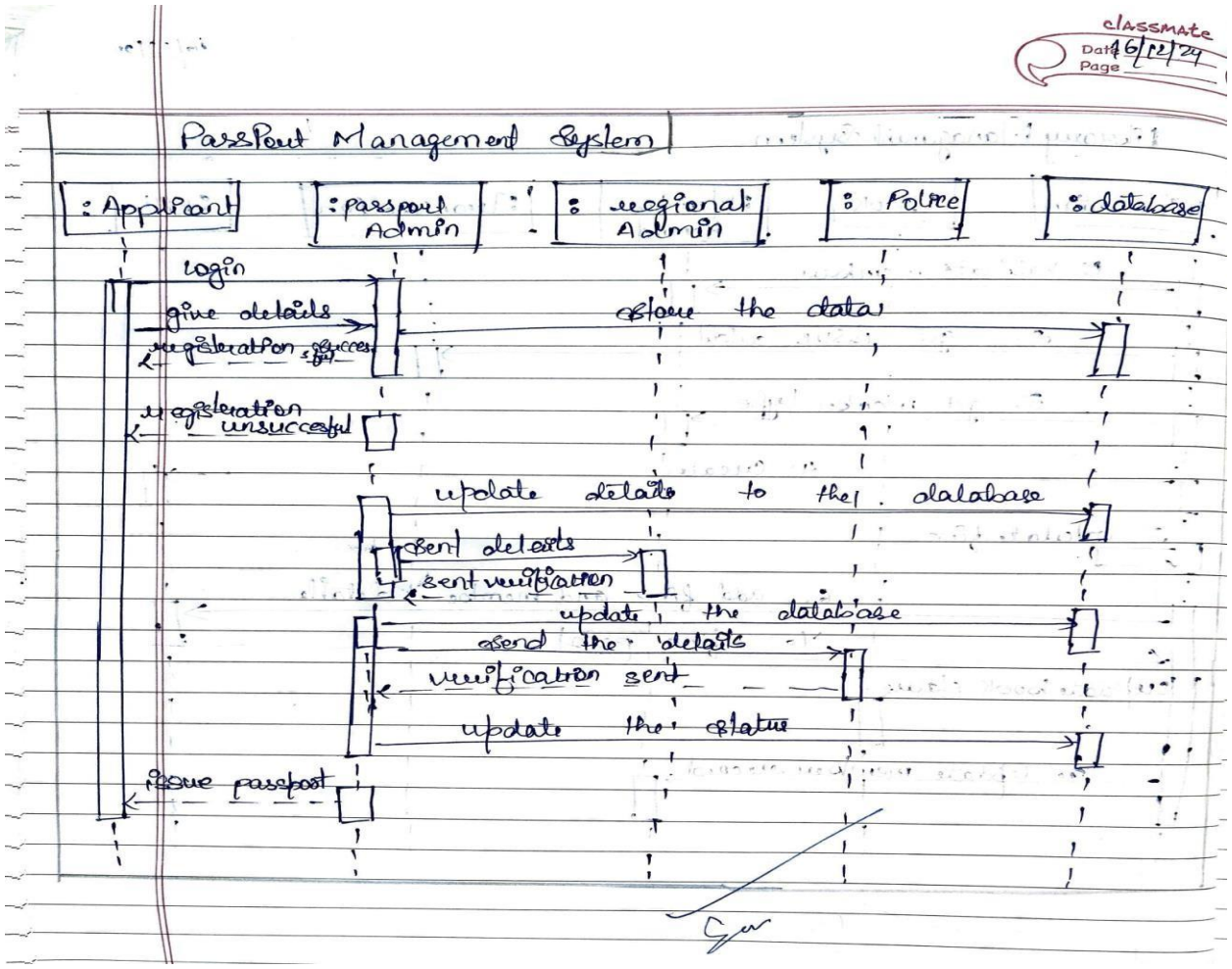


Fig 5.11

Brief descriptions:

1. **Initiate:** The applicant initiates the passport application process.
2. **Submit application and required documents:** The applicant submits the passport application form and required supporting documents to the Passport Office.
3. **Verify personal and document details:** The Passport Office receives the application and forwards it to the Verification Department for verification.
4. **Provide verification results:** The Verification Department verifies the applicant's personal information and documents and sends the results back to the Passport Office.
5. **Inform verification status and next steps:** The Passport Office informs the applicant about the verification status and the next steps in the process.
6. **Pay the application processing fee upon approval:** If the application is approved, the applicant is notified and asked to pay the application processing fee.
7. **Issue passport and provide collection details:** After the fee is paid, the Passport Office processes the application and issues the passport. The applicant is notified about the passport issuance and provided with details on how to collect it.
8. **Terminate:** The application process is completed.

ACTIVITY DIAGRAM

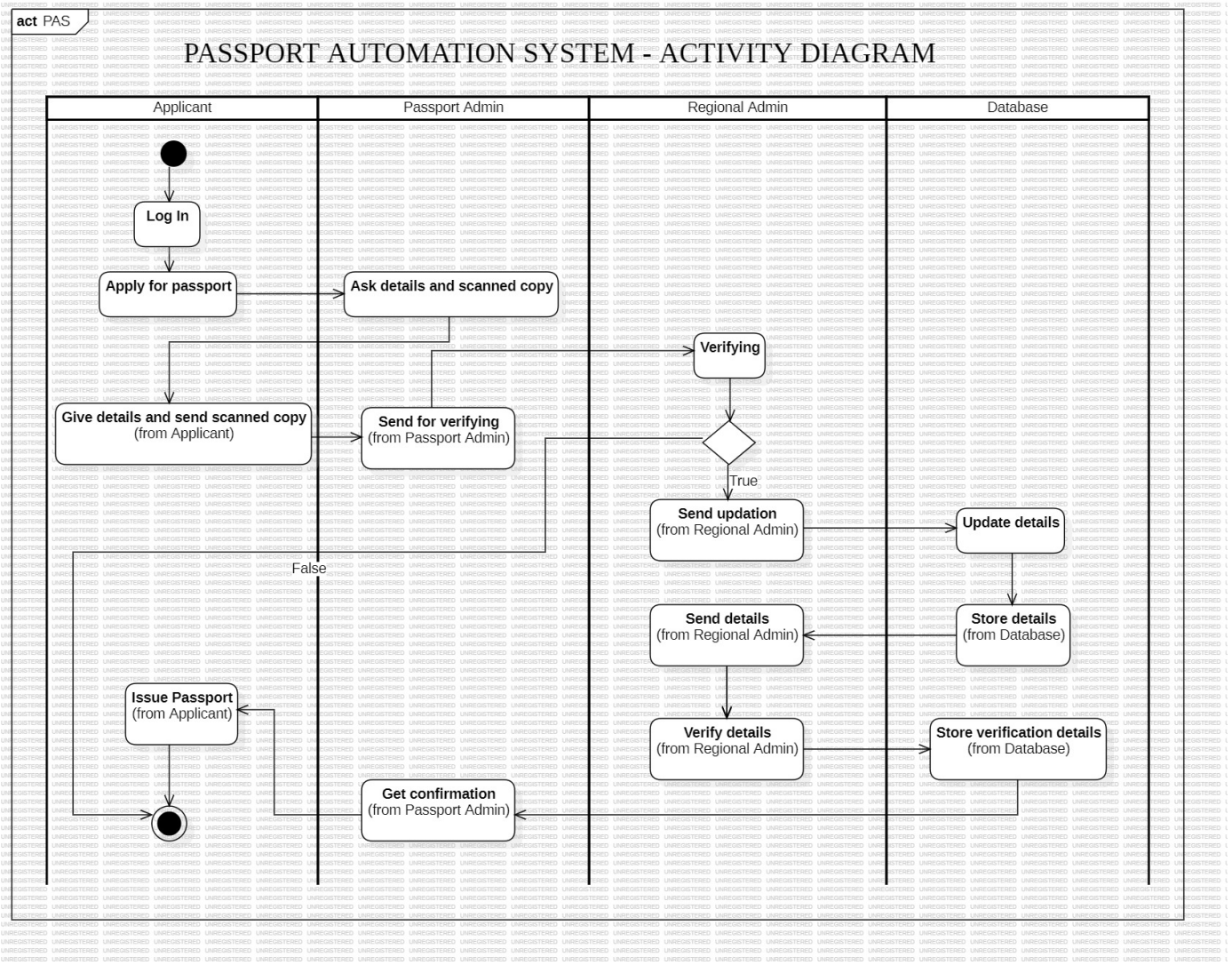


Fig 5.12

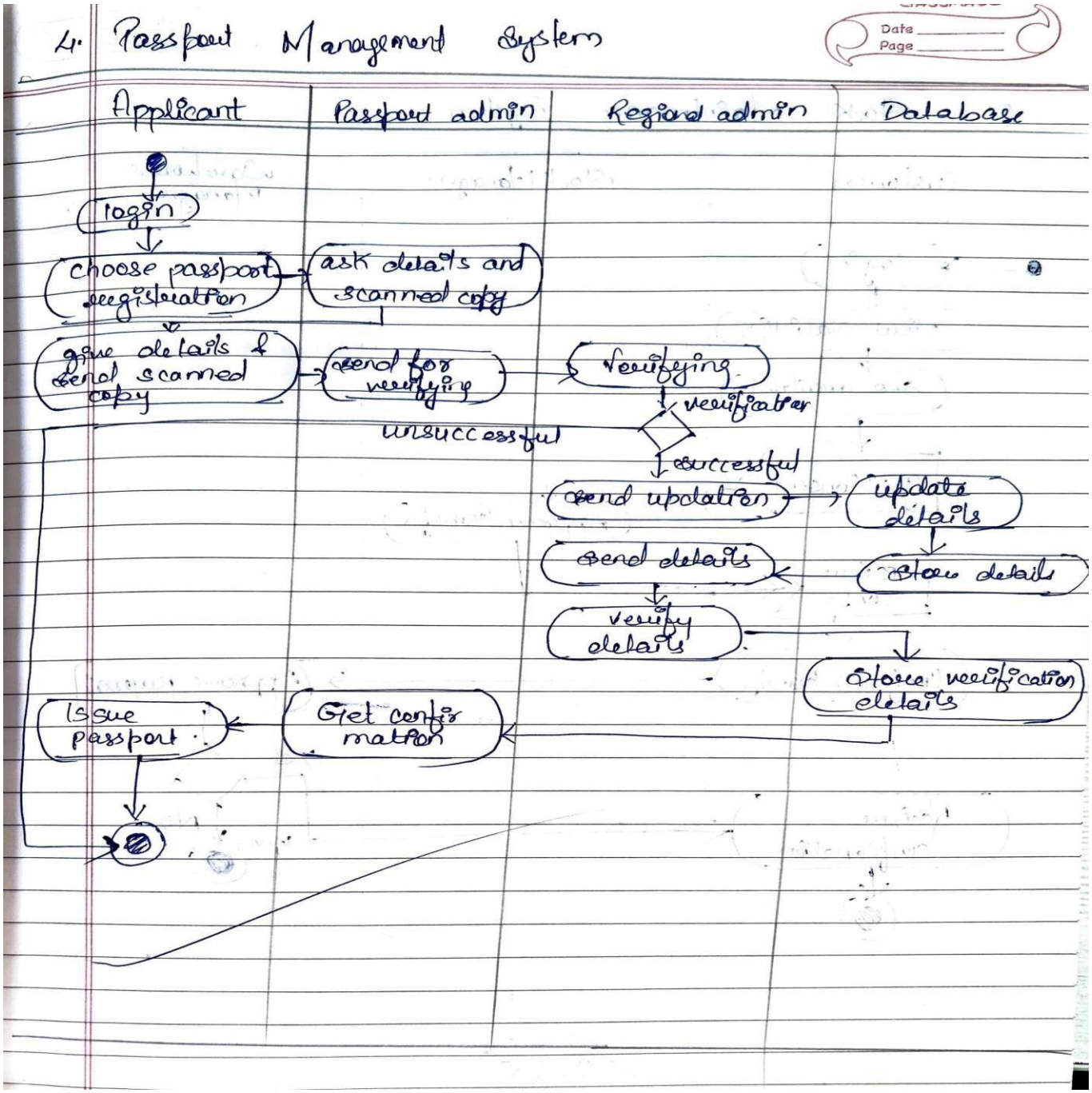

4. Passport Management System
Applicant
Passport admin
Regional admin
Database
login
choose passport registeration
ask details and scanned copy
give details & send scanned copy
send for verifying
verifying
verification
unsuccessful
successful
send updation
update details
send details
store details
verify details
store verification details
issue passport
Get confirmation

Fig 5.13

Brief descriptions:

- **Log In:** The Applicant initiates the process by logging into the system.
- **Apply for Passport:** The Applicant applies for a passport.
- **Ask for details and scanned copy:** The Passport Admin asks the Applicant for details and scanned copies of required documents.
- **Give details and send scanned copies:** The Applicant provides the details and sends the scanned copies to the Passport Admin.
- **Send for verifying:** The Passport Admin sends the details and documents to the Regional Admin for verification.
- **Verifying:** The Regional Admin verifies the details and documents.
- **Send updation:** If the verification is successful (True), the Regional Admin sends an update to the Database.
- **Send details:** If the verification is unsuccessful (False), the Regional Admin sends the details back to the Passport Admin.
- **Store details:** The Database stores the updated details or the details returned by the Regional Admin.
- **Verify details:** The Regional Admin verifies the details again.
- **Store verification details:** The Database stores the verification details.
- **Get confirmation:** The Passport Admin receives confirmation from the Regional Admin.
- **Issue Passport:** The Passport Admin issues the passport to the Applicant.