

PHARMACY MANAGEMENT SYSTEM



A PROJECT REPORT

Submitted by

MUKILAN K 2303811724321072

in partial fulfillment of requirements for the award of the course

CGB1201 – JAVA PROGRAMMING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by
AICTE, New Delhi)

SAMAYAPURAM – 621 112

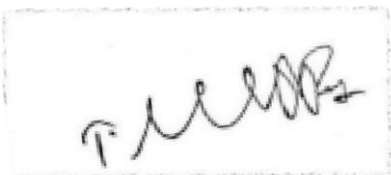
DECEMBER, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “**PHARMACY MANAGEMENT SYSTEM**”
is the bonafide work of **MUKILAN K 2303811724321072** who carried out the
project work during the academic year 2024 - 2025 under my supervision.



Signature


Dr. T. AVUDAIAPPAN M.E., Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.



Signature

Mrs. S. GEETHA M.E.,

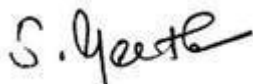
SUPERVISOR,

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 3.12.24 .



INTERNAL EXAMINER



EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**PHARMACY MANAGEMENT SYSTEM**” is the result of original work done by me and best of my knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING**.

Signature



MUKILAN K

Place: Samayapuram

Date: 3/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, **“K. Ramakrishnan College of Technology (Autonomous)”**, for providing us with the opportunity to do this project.

I extend our sincere acknowledgement and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.**, Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr.T.AVUDAIAPPAN, M.E.,Ph.D.**, Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.S.GEETHA M.E.**, Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards.

MISSION OF THE INSTITUTION

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics.

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

The Pharmacy Management System is designed to streamline pharmacy operations by providing an efficient platform for managing **inventory, prescriptions, billing, and alerts**. It allows pharmacists to add and track medicine details, such as **quantity, expiry date, and price**, while generating automated alerts for low stock and approaching expiration. The system also facilitates the creation and management of prescriptions by linking patients, doctors, and medicines. Additionally, it includes a billing module to generate detailed invoices based on prescriptions, ensuring accurate and seamless transactions. Developed using Java Programming, the project demonstrates **Object-Oriented Programming (OOP)** principles like **encapsulation and modular design**. Dynamic data handling is achieved using Java's Collections **Framework**, incorporating **ArrayList** for efficient storage and retrieval. With an intuitive, menu-driven console interface, the system ensures ease of use while maintaining simplicity. Designed with scalability in mind, the project can be enhanced with future integrations, such as **graphical interfaces or databases**, addressing the inefficiencies of manual pharmacy operations and providing a comprehensive automated solution.

TABLE OF CONTENTS

| CHAPTER No. | TITLE | PAGE No. |
|------------------------|---|---------------------|
| | ABSTRACT | viii |
| 1 | INTRODUCTION | 1 |
| | 1.1 INTRODUCTION | 1 |
| | 1.2 OBJECTIVE | 1 |
| 2 | PROJECT METHODOLOGY | 3 |
| | 2.1 PROPOSED WORK | 3 |
| | 2.2 BLOCK DIAGRAM | 4 |
| 3 | JAVA PROGRAMMING CONCEPTS | 5 |
| | 3.1 OBJECT ORIENTED PROGRAMMING LANGUAGE | 5 |
| | 3.2 GUI COMPONENTS | 5 |
| 4 | MODULE DESCRIPTION | 6 |
| | 4.1 MAIN APPLICATION MODULE | 6 |
| | 4.2 INVENTORY MODULE | 7 |
| | 4.3 PRESCRIPTION MODULE | 8 |
| | 4.4 BILLING MODULE | 9 |
| 5 | CONCLUSION | 10 |
| | REFERENCES | 10 |
| | APPENDICES | 13 |
| | Appendix A – Source code | 13 |
| | Appendix B – Screen shots | 23 |

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Pharmacy Management System is a Java-based application designed to simplify and optimize the operations of a pharmacy through an intuitive console-based interface. The system facilitates efficient management of inventory, prescriptions, billing, and alert systems, ensuring seamless workflows for pharmacists and staff. It enables users to add and track medicine details, monitor stock levels, and manage expiration dates effectively. Additionally, the system supports prescription creation by associating medicines with patients and doctors, while also generating accurate invoices based on prescription details. By leveraging Java's robust Object-Oriented Programming (OOP) principles and its Collections Framework, the application ensures efficient real-time data handling. The Pharmacy Management System serves as a comprehensive solution, eliminating manual inefficiencies and offering a scalable, automated platform for pharmacy management.

1.2 OBJECTIVE

The primary objective of the Pharmacy Management System is to provide a robust, scalable, and user-friendly solution that simplifies pharmacy operations and ensures accurate and efficient management of essential tasks. The system aims to:

1. Streamline inventory management: Automating the addition, tracking, and monitoring of medicines, including stock levels and expiration dates.
2. Enhance operational efficiency: Simplifying tasks such as prescription creation, billing, and alerts through an intuitive interface.
3. Demonstrate Java's capabilities: Utilizing Object-Oriented Programming (OOP), data encapsulation, and the Collections Framework to enable dynamic and efficient data management.
4. Address manual inefficiencies: Replacing traditional manual processes with a faster, more accurate, and automated system.

5. Improve user experience: Offering a simple, menu-driven interface for pharmacists to manage daily tasks with ease.
6. Provide real-time alerts: Generating notifications for low stock or medicines nearing expiration to prevent shortages or wastage.
7. Ensure scalability: Designing the system with modularity, allowing for future enhancements, such as graphical interfaces or integration with databases.
8. Promote accuracy and reliability: Ensuring error-free billing, prescription generation, and inventory tracking for better pharmacy operations.

CHAPTER 2

PROJECT METHODOLOGY

PROPOSED WORK

The Pharmacy Management System aims to streamline pharmacy operations by providing an efficient, intuitive, and user-friendly Java-based application. The project focuses on automating key processes such as inventory management, prescription handling, billing, and alerts.

1. Understanding Requirements:

- Identifying the need for functionalities like inventory tracking, prescription creation, and billing.
- Analyzing traditional pharmacy systems to identify inefficiencies and areas for improvement.

2. System Design:

- Structuring modules for inventory, prescriptions, billing, and alert systems to ensure modularity and scalability.
- Designing a console-based interface with clearly defined navigation for users.

3. Implementation:

- Developing modular Java code using Object-Oriented Programming principles.
- Utilizing Java Collections Framework components like HashMap and ArrayList for efficient real-time data handling.

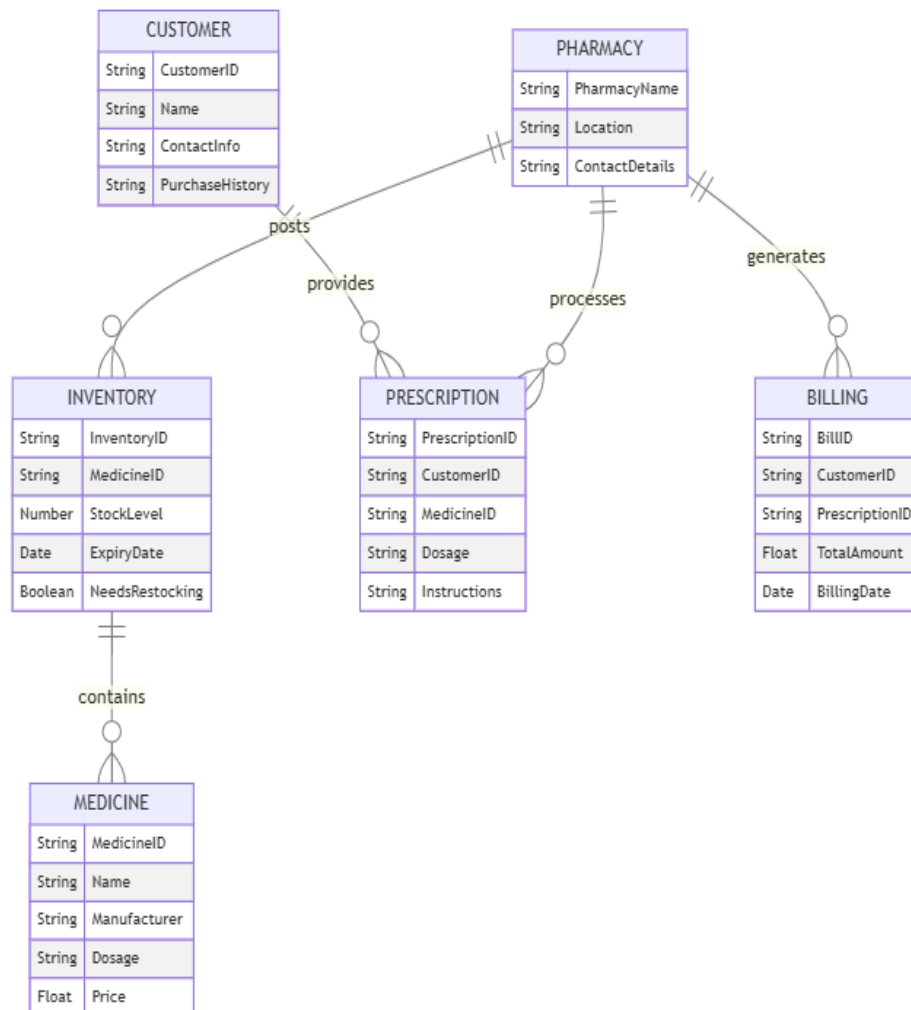
4. Testing and Validation:

- Conducting independent testing of each module to verify functionality and integration.
- Ensuring error-free data processing and accurate outputs, including alerts and billing details.

5. Deployment and Feedback:

- Delivering a functional console-based interface for end user

2.1 BLOCK DIAGRAM



CHAPTER 3

JAVA PROGRAMMING CONCEPTS

3.1 OBJECT-ORIENTED PROGRAMMING (OOP):

- **Classes and Objects:** Represent various entities like medicines, prescriptions, billing details, and user roles (e.g., pharmacy staff, customers) to model the system's functionalities.
- **Encapsulation:** Securely manage data by keeping sensitive information such as medicine details, stock levels, and billing information in private fields, accessed and modified via public getters and setters.
- **Inheritance:** Use inheritance to extend common functionalities for different user roles (e.g., pharmacist and customer), enhancing modularity and code reusability, and making the system extensible for future upgrades.

3.2 JAVA COLLECTIONS FRAMEWORK:

- **HashMap:** Efficiently stores data such as medicine details, user credentials, and prescription records, allowing fast lookup and management of key-value pairs.
- **ArrayList:** Dynamically manages lists of medicines, prescriptions, and job history, ensuring flexibility in adding, removing, and displaying data as the system scales.

3.3 CONTROL FLOW MECHANISMS:

- **Loops and Conditional Statements:** Facilitate smooth user interactions, enabling navigation between system menus and operations based on the user's choices and input validation.
- **Switch Case:** Streamlines the user interface by handling different choices in the menu, such as adding medicines, viewing prescriptions, and generating bills, in an organized and readable manner.

3.4 INPUT HANDLING WITH SCANNER:

- **Scanner:** Captures and validates user inputs for tasks such as adding new medicines, registering prescriptions, generating bills, and receiving alerts, ensuring that the system processes user actions accurately and securely.

CHAPTER 4

MODULE DESCRIPTION

4.1 MAIN APPLICATION MODULE

4.1.1 Objective

The Main Application Module serves as the entry point for the Pharmacy Management System. It provides an intuitive user interface for accessing core functionalities such as inventory management, prescription management, billing, and alert system. This module ensures smooth navigation, helping users easily transition between different sections of the system based on their role (pharmacy staff or customer).

4.1.2 Design

- **Graphical Components Used:**

- Frame: The main container that holds the entire user interface for the pharmacy management system.
- Button: Interactive elements for selecting options such as "Inventory Management," "Prescription Management," "Billing," and "Alerts."
- Label: Displays static text providing instructions and guiding users through the available options.

- **Navigation Workflow:**

1. User launches the application.
2. The main menu displays various options (e.g., Inventory, Prescription, Billing, Alerts).
3. Based on the user's selection, the corresponding module (Inventory, Prescription, Billing, or Alerts) is loaded.

4.1.3 Usage

This module acts as the first point of interaction for users, offering a simple, straightforward navigation system. It ensures that users can easily understand and access all the features of the pharmacy management system by selecting their required functionality.

4.2 INVENTORY MODULE

4.2.1 Objective

The Inventory Module allows pharmacy staff to manage the stock of medicines, including adding new medicines, updating quantities, and monitoring expiry dates. This module ensures that the pharmacy's inventory is well-organized, and stock levels are accurately tracked.

4.2.2 Submodules

1. Add Medicine Submodule:

- Objective: Enables pharmacy staff to add new medicines to the system with detailed information.
- Features:
 - Captures details such as Medicine ID, Name, Quantity, Expiry Date, and Price.
 - Ensures that the medicine ID is unique and validates the expiry date.
- Workflow:
 1. User enters details (ID, name, quantity, expiry date, price) in the form.
 2. The system validates the data.
 3. Data is saved to the inventory database.

2. View Inventory Submodule:

- Objective: Provides an overview of the entire inventory, including available medicines.
- Features:
 - Displays a list of all medicines with details such as ID, Name, Quantity, Expiry Date, and Price.
- Workflow:
 1. User clicks on the "View Inventory" option.
 2. The system retrieves and displays the inventory list.

3. Stock Alert Submodule:

- Objective: Alerts the pharmacy staff about low stock levels or medicines nearing expiration.
- Features:
 - Alerts are triggered when a medicine's stock reaches a predefined threshold or when a medicine is approaching its expiry date.
- Workflow:
 1. The system regularly checks the stock levels and expiry dates.
 2. If any medicine meets the alert criteria, a notification is displayed.

4.3 PRESCRIPTION MODULE

4.3.1 Objective

The Prescription Module enables pharmacy staff to create, manage, and view prescriptions associated with patients and doctors. It ensures efficient tracking of medication prescriptions.

4.3.2 Submodules

1. Create Prescription Submodule:

- Objective: Allows the creation of new prescriptions by associating medicines with patients and doctors.
- Features:
 - Captures patient details, doctor information, and prescribed medicines.
 - Validates the medicines being prescribed to ensure availability in the inventory.
- Workflow:
 1. User enters patient and doctor details, and selects medicines from the inventory.
 2. The system validates the medicines and creates a prescription record.

2. View Prescription Submodule:

- Objective: Provides a list of all prescriptions, showing relevant details about the patient, doctor, and prescribed medicines.
- Features:

- Displays patient and doctor information along with the associated medicines.
- Workflow:
 1. User selects the "View Prescriptions" option.
 2. The system retrieves and displays a list of all prescriptions.

4.4 BILLING MODULE

4.4.1 Objective

The Billing Module allows pharmacy staff to generate bills for customers based on the medicines prescribed and their respective prices.

4.4.2 Submodules

1. Generate Bill Submodule:

- Objective: Generates a detailed bill based on the prescription and associated medicines.
- Features:
 - Calculates the total cost based on the medicines listed in the prescription.
 - Displays the breakdown of costs, including any applicable taxes.
- Workflow:
 1. User selects a prescription from the database.
 2. The system retrieves medicine details and calculates the total cost.
 3. The bill is generated and displayed.

2. View Bill Submodule:

- Objective: Displays past bills for reference.
- Features:
 - Allows the user to search for bills by patient name, date, or prescription ID.
- Workflow:
 1. User selects "View Bill" and enters search criteria.
 2. The system retrieves and displays the matching bills.

CHAPTER 5

CONCLUSION

SUMMARY OF ACHIEVEMENTS

The Pharmacy Management System has successfully met the needs of pharmacy staff and patients through the following:

1. For Pharmacy Staff:

- **Efficient Inventory Management:** Pharmacy staff can add, view, and manage medicine stock, ensuring that stock levels are accurately tracked.
- **Prescription Management:** Allows for the creation, tracking, and management of prescriptions, ensuring accurate medicine dispensing.
- **Billing and Transactions:** Facilitates the generation of bills based on prescriptions, ensuring a smooth transaction process.

2. For Patients:

- **Prescription Tracking:** Patients' prescribed medicines are easily accessible and can be reviewed at any time.
- **Accurate Billing:** Transparent billing based on prescribed medicines and their prices.

3. Overall System:

- **Scalable and User-Friendly Design:** The system is designed to be modular and scalable for future updates.
- **Data Security:** Sensitive patient and prescription data are handled securely with appropriate access controls.

LIMITATIONS

1. **No Persistent Data Storage:** The system currently uses in-memory data structures, meaning data is lost when the application is restarted.
2. **Basic User Interface:** Due to the limitations of the chosen GUI framework, the aesthetic design could be further improved.

3. **Limited Reporting Features:** The current version lacks advanced reporting capabilities for inventory and billing analysis.

FUTURE SCOPE

1. **Database Integration:** Transition to a database (SQL or NoSQL) for persistent data storage and better scalability.
2. **User Authentication and Role Management:** Implement secure login and role-based access (e.g., for pharmacists, patients, and administrators).
3. **Advanced Inventory Features:** Include automatic restocking notifications based on inventory levels and expiry date tracking.
4. **Mobile Application:** Extend the system for Android or iOS platforms for on-the-go pharmacy management using JavaFX, Kotlin, or Flutter.
5. **Analytics and Reporting:** Add detailed analytics for sales, inventory, and prescription trends to provide insights for better management.

REFERENCES:

The development of this project involved the use of several resources to understand Java programming and apply best practices. Below are the references that were used:

Books:

1. **Core Java Volume I: Fundamentals** by Cay S. Horstmann and Gary Cornell
 - A comprehensive guide to Java fundamentals, object-oriented programming, and collections.
2. **Effective Java** by Joshua Bloch
 - Practical programming techniques and best practices for Java.

Online Tutorials:

1. **W3Schools Java Tutorials:** <https://www.w3schools.com/java/>
 - For understanding basic to advanced Java programming concepts.
2. **GeeksforGeeks Java Programming:** <https://www.geeksforgeeks.org/java/>
 - Detailed explanations of Java collections, object-oriented principles, and practical examples.

Documentation:

1. **Oracle Java Documentation:** <https://docs.oracle.com/javase/tutorial/>
 - Official documentation for Java programming and the Collections Framework.
2. **Java SE 17 API Documentation:** <https://docs.oracle.com/en/java/javase/17/docs/api/>
 - Reference for Java APIs used in the project.

Videos:

1. **Programming with Mosh – Java Full Course (YouTube):**
<https://www.youtube.com/watch?v=grEKMHGYYns>
 - Comprehensive video tutorials covering Java basics to advanced topics.
2. **CodeWithHarry – Java for Beginners (YouTube):**
<https://www.youtube.com/CodeWithHarry>
 - Simplified explanation of Java programming concepts.

APPENDICES

APPENDIX A – SOURCE CODE

MEDICIEN CLASS

```
package pharmacy.system;
import java.text.SimpleDateFormat;
import java.util.Date;
public class Medicine {
    private int id;
    private String name;
    private int quantity;
    private Date expiryDate;
    private double price;
    private static final int LOW_STOCK_THRESHOLD = 5;
    private static final int EXPIRY_THRESHOLD_DAYS = 30;
    public Medicine(int id, String name, int quantity, Date expiryDate, double price) {
        this.id = id;
        this.name = name;
        this.quantity = quantity;
        this.expiryDate = expiryDate;
        this.price = price;
    }
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public int getQuantity() {
        return quantity;
    }
    public Date getExpiryDate() {
        return expiryDate;
    }
    public double getPrice() {
        return price;
    }
    public boolean isLowStock() {
        return this.quantity <= LOW_STOCK_THRESHOLD;
    }
    public boolean isNearExpiry() {
        long millisInDay = 1000 * 60 * 60 * 24;
        long difference = (expiryDate.getTime() - new Date().getTime()) / millisInDay;
```

```

        return difference <= EXPIRY_THRESHOLD_DAYS;
    }
    @Override
    public String toString() {
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        return "Medicine ID: " + id + ", Name: " + name + ", Quantity: " + quantity + ", Expiry
Date: " + dateFormat.format(expiryDate) + ", Price: $" + price;
    }
}

```

INVENTORY CLASS

```

package pharmacy.system;

import java.util.ArrayList;
import java.util.List;

public class Inventory {

    private List<Medicine> medicines;

    public Inventory() {
        this.medicines = new ArrayList<>();
    }

    public void addMedicine(Medicine medicine) {
        medicines.add(medicine);
    }

    public List<Medicine> getMedicines() {
        return medicines;
    }

    public Medicine findMedicineById(int id) {
        for (Medicine medicine : medicines) {
            if (medicine.getId() == id) {
                return medicine;
            }
        }
        return null;
    }

    public void checkLowStockAndExpiry() {
        System.out.println("\n** Stock Alerts **");
        for (Medicine medicine : medicines) {

```

```

        if (medicine.isLowStock()) {
            System.out.println("Low stock alert for Medicine: " + medicine.getName() + "
(ID: " + medicine.getId() + ")");
        }
        if (medicine.isNearExpiry()) {
            System.out.println("Expiry alert for Medicine: " + medicine.getName() + " (ID: "
+ medicine.getId() + ")");
        }
    }
}
}

```

PRESCRIPTION CLASS

```
package pharmacy.system;
```

```
import java.util.Date;
import java.util.List;
```

```
public class Prescription {
```

```

    private int id;
    private int patientId;
    private int doctorId;
    private Date date;
    private List<Medicine> medicines;

```

```

    public Prescription(int id, int patientId, int doctorId, Date date, List<Medicine>
medicines) {
        this.id = id;
        this.patientId = patientId;
        this.doctorId = doctorId;
        this.date = date;
        this.medicines = medicines;
    }

```

```

    public int getId() {
        return id;
    }

```

```

    public double calculateTotalCost() {
        double totalCost = 0;
        for (Medicine medicine : medicines) {
            totalCost += medicine.getPrice();
        }
    }

```



```

        return totalCost;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("Prescription ID: ").append(id)
            .append(", Patient ID: ").append(patientId)
            .append(", Doctor ID: ").append(doctorId)
            .append("\nDate: ").append(date)
            .append("\nMedicines:\n");
        for (Medicine medicine : medicines) {
            sb.append(medicine).append("\n");
        }
        sb.append("Total Cost: $").append(calculateTotalCost());
        return sb.toString();
    }
}

```

PHARMACY MANAGEMENT SYSTEM

```

package pharmacy.system;
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.swing.*;

public class PharmacySystemAWT extends Frame {
    private Inventory inventory;
    private List<Prescription> prescriptions;
    private TextArea displayArea;
    public PharmacySystemAWT() {
        this.inventory = new Inventory();
        this.prescriptions = new ArrayList<>();
        init();
        prepareGUI();
    }
    private void prepareGUI() {
        setTitle("Pharmacy Management System");
        setSize(600, 400);
        setLayout(new BorderLayout());
        Label titleLabel = new Label("Pharmacy Management System", Label.CENTER);
    }
}

```

```

titleLabel.setFont(new Font("Arial", Font.BOLD, 28));
titleLabel.setBackground(Color.DARK_GRAY);
titleLabel.setForeground(Color.WHITE);
add(titleLabel, BorderLayout.NORTH);
displayArea = new TextArea("\n\n\tWelcome to Pharmacy Management
System.\n\tPlease select an option from the right panel to proceed.");
displayArea.setFont(new Font("Serif", Font.BOLD, 20));
displayArea.setEditable(false);
displayArea.setBackground(Color.LIGHT_GRAY);
displayArea.setForeground(Color.DARK_GRAY);
add(displayArea, BorderLayout.CENTER);
Panel buttonPanel = new Panel();
buttonPanel.setLayout(new GridLayout(6, 1, 10, 10));
Button addMedicineButton = createButton("Add Medicine", Color.ORANGE);
Button viewInventoryButton = createButton("View Inventory", Color.CYAN);
Button addPrescriptionButton = createButton("Add Prescription", Color.MAGENTA);
Button viewPrescriptionsButton = createButton("View Prescriptions", Color.GREEN);
Button generateBillButton = createButton("Generate Bill", Color.YELLOW);
Button checkAlertsButton = createButton("Check Stock/Expiry Alerts", Color.PINK);
buttonPanel.add(addMedicineButton);
buttonPanel.add(viewInventoryButton);
buttonPanel.add(addPrescriptionButton);
buttonPanel.add(viewPrescriptionsButton);
buttonPanel.add(generateBillButton);
buttonPanel.add(checkAlertsButton);

add(buttonPanel, BorderLayout.EAST);
Panel exitPanel = new Panel();
Button exitButton = new Button("Exit");
exitButton.setFont(new Font("Arial", Font.BOLD, 16));
exitButton.setBackground(Color.RED);
exitButton.setForeground(Color.WHITE);

exitButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        displayArea.setText("\n\n\tWelcome to Pharmacy Management System.\n\tPlease
select an option from the right panel to proceed.");
    }
});
exitPanel.add(exitButton);
add(exitPanel, BorderLayout.SOUTH);
addMedicineButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        addMedicine();
    }
}

```

```

});
viewInventoryButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        viewInventory();
    }
});
addPrescriptionButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        addPrescription();
    }
});
viewPrescriptionsButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        viewPrescriptions();
    }
});
generateBillButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        generateBill();
    }
});
checkAlertsButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        checkAlerts();
    }
});
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent windowEvent) {
        System.exit(0);
    }
});
setVisible(true);
}
private Button createButton(String label, Color color) {
    Button button = new Button(label);
    button.setFont(new Font("Arial", Font.BOLD, 16));
    button.setBackground(color);
    button.setForeground(Color.BLACK);
    return button;
}
private void init() {
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
    try {
        inventory.addMedicine(new Medicine(1, "Paracetamol", 5, dateFormat.parse("2024-12-21"), 10.0));
    }
}

```

```

        inventory.addMedicine(new Medicine(2, "Ibuprofen", 3, dateFormat.parse("2025-01-10"), 15.0));
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void addMedicine() {
    TextField idField = new TextField();
    TextField nameField = new TextField();
    TextField quantityField = new TextField();
    TextField priceField = new TextField();
    TextField expiryDateField = new TextField();

    Object[] message = {
        "Medicine ID:", idField,
        "Medicine Name:", nameField,
        "Quantity:", quantityField,
        "Price:", priceField,
        "Expiry Date (yyyy-MM-dd):", expiryDateField
    };

    int option = JOptionPane.showConfirmDialog(this, message, "Add Medicine",
JOptionPane.OK_CANCEL_OPTION);
    if (option == JOptionPane.OK_OPTION) {
        try {
            int id = Integer.parseInt(idField.getText());
            String name = nameField.getText();
            int quantity = Integer.parseInt(quantityField.getText());
            double price = Double.parseDouble(priceField.getText());
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
            Date expiryDate = dateFormat.parse(expiryDateField.getText());

            Medicine medicine = new Medicine(id, name, quantity, expiryDate, price);
            inventory.addMedicine(medicine);
            displayArea.setText("Medicine added successfully:\n" + medicine);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Invalid input. Please try again.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
}

private void addPrescription() {
    TextField prescriptionIdField = new TextField();
    TextField patientIdField = new TextField();

```

```

TextField doctorIdField = new TextField();
TextField numMedicinesField = new TextField();

Object[] message = {
    "Prescription ID:", prescriptionIdField,
    "Patient ID:", patientIdField,
    "Doctor ID:", doctorIdField,
    "Number of Medicines:", numMedicinesField
};

int option = JOptionPane.showConfirmDialog(this, message, "Add Prescription",
JOptionPane.OK_CANCEL_OPTION);
if (option == JOptionPane.OK_OPTION) {
    try {
        int prescriptionId = Integer.parseInt(prescriptionIdField.getText());
        int patientId = Integer.parseInt(patientIdField.getText());
        int doctorId = Integer.parseInt(doctorIdField.getText());
        int numMedicines = Integer.parseInt(numMedicinesField.getText());

        List<Medicine> medicines = new ArrayList<>();
        for (int i = 0; i < numMedicines; i++) {
            TextField medicineIdField = new TextField();
            Object[] medicineMessage = {
                "Medicine ID " + (i+1) + ":", medicineIdField
            };
            int medicineOption = JOptionPane.showConfirmDialog(this, medicineMessage,
"Enter Medicine ID", JOptionPane.OK_CANCEL_OPTION);
            if (medicineOption == JOptionPane.OK_OPTION) {
                int medicineId = Integer.parseInt(medicineIdField.getText());
                Medicine medicine = inventory.findMedicineById(medicineId);
                if (medicine != null) {
                    medicines.add(medicine);
                } else {
                    JOptionPane.showMessageDialog(this, "Medicine not found. Skipping.",
"Warning", JOptionPane.WARNING_MESSAGE);
                }
            }
        }

        Date date = new Date();
        Prescription prescription = new Prescription(prescriptionId, patientId, doctorId,
date, medicines);
        prescriptions.add(prescription);
        displayArea.setText("Prescription added successfully:\n" + prescription);
    } catch (Exception e) {

```

```

        JOptionPane.showMessageDialog(this, "Invalid input. Please try again.", "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void viewInventory() {
    StringBuilder sb = new StringBuilder();
    sb.append("Current Inventory:\n");
    for (Medicine medicine : inventory.getMedicines()) {
        sb.append(medicine).append("\n");
    }
    displayArea.setText(sb.toString());
}
private void viewPrescriptions() {
    StringBuilder sb = new StringBuilder();
    sb.append("Prescriptions:\n");
    for (Prescription prescription : prescriptions) {
        sb.append(prescription).append("\n");
    }
    displayArea.setText(sb.toString());
}
private void generateBill() {
    TextField prescriptionIdField = new TextField();

    Object[] message = {
        "Prescription ID:", prescriptionIdField
    };
    int option = JOptionPane.showConfirmDialog(this, message, "Generate Bill",
JOptionPane.OK_CANCEL_OPTION);
    if (option == JOptionPane.OK_OPTION) {
        try {
            int prescriptionId = Integer.parseInt(prescriptionIdField.getText());
            for (Prescription prescription : prescriptions) {
                if (prescription.getId() == prescriptionId) {
                    StringBuilder sb = new StringBuilder();
                    sb.append("Bill for Prescription ID: ").append(prescriptionId).append("\n");
                    sb.append(prescription);
                    displayArea.setText(sb.toString());
                    return; }
            }
            JOptionPane.showMessageDialog(this, "Prescription not found.", "Error",
JOptionPane.ERROR_MESSAGE);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Invalid input. Please try again.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

```

private void checkAlerts() {
    StringBuilder sb = new StringBuilder();
    sb.append("Stock and Expiry Alerts:\n");
    for (Medicine medicine : inventory.getMedicines()) {
        if (medicine.getQuantity() < 5) {
            sb.append("Low stock alert: ").append(medicine).append("\n");
        }
        if (medicine.getExpiryDate().before(new Date())) {
            sb.append("Expired: ").append(medicine).append("\n"); } }
    displayArea.setText(sb.toString()); }
public static void main(String[] args) {
    new PharmacySystemAWT();
}
}

```

APPENDIX B - SCREENSHOTS

