Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Semester: (Fall, Year: 2024), B.Sc. in CSE (Day)

Pharmaceutical Management System

Course Title: Algorithm

Course Code: CSE 208

Section: 231 D1

Students Details

Name	ID
Asfiqul Alam Emran	231902049

Submission Date: 15/12/2024

Course Teacher's Name: Fatema Tuj Johora

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	Signature:
Comments:	Date:

Contents

Chapter 1

Introduction

1.1 Overview

The Pharmaceutical Management System, implemented in java, is a user-friendly solution designed for efficient pharmacy inventory management. Through a menu-driven console application, users can easily add, update, and access detailed information about medicines while checking stock availability. The project's documentation is well-organized, covering essential aspects such as project introduction, design, implementation, and future scope, contributing to a comprehensive pharmaceutical management system.

1.2 Motivation

The focus on the Pharmaceutical Management System stems from the increasing complexity and critical importance of pharmaceutical operations in today's healthcare landscape. Efficient management is vital for patient care, safety, and regulatory compliance. The project is motivated by factors such as its impact on healthcare, ensuring regulatory compliance, addressing inventory challenges, improving operational efficiency through automation, and offering educational value by applying programming skills to real-world healthcare challenges.

1.3 Problem Definition

1.3.1 Problem Statement

The pharmaceutical industry's complexity and stringent regulations underscore the need for an efficient Pharmaceutical Management System. Pharmacies grapple with challenges such as inadequate inventory control, leading to stock outs or overstock situations. Manual data entry errors in medicine information pose risks to patient safety and regulatory compliance. Addressing regulatory compliance challenges, pharmacies require a system to ensure accurate record-keeping and reporting.

1.3.2 Complex Engineering Problem

The challenge lies in designing a system that can manage these various data sets in real-time, ensure accuracy in resource allocation, and maintain scalability as the scope of a Pharmacy management System . One of the key engineering problems in this project is the integration of diverse data sources, while maintaining data consistency and integrity across multiple entities involved in the relief process. Additionally, the system must handle concurrent updates, support rapid data retrieval, and offer a robust mechanism for conflict resolution when different agencies input overlapping or contradictory information. This requires an optimized database structure that can support complex relationships, handle large-scale queries efficiently, and ensure the system remains responsive under heavy load during disaster scenarios.

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	Explain how to address
P1: Depth of knowledge required	The system requires a deep understanding of pharmaceutical management principles, and java programming.
P2: Range of conflicting requirements	Address by conducting thorough requirement analysis and stakeholder consultations to identify conflicting requirements early in the development process.
P3: Depth of analysis required	Address by breaking down complex problems into smaller, more manageable components and conducting systematic analysis at each level.

P4: Familiarity of issues	The team ensured familiarity with pharmaceutical management challenges through extensive literature review, consultations with domain experts, and studying existing systems.
P5: Extent of applicable codes	Address by conducting a thorough review of applicable codes, standards, regulations, and industry guidelines relevant to the project.
P6: Extent of stakeholder involvement and conflicting requirements	Address by establishing effective communication channels and collaboration framework to engage stakeholders throughout the project life cycle.
P7: Interdependence	Address by identifying and mapping interdependence between project components, tasks, and stakeholders.

1.4 Design Goals/Objectives

The Pharmaceutical Management System is designed with the following goals and objectives in mind.

1. Error-Free Data Management: Minimize manual data entry errors through robust validation mechanisms and user-friendly interfaces.

2. Regulatory Compliance: Facilitate adherence to pharmaceutical regulations and standards with features for accurate record-keeping and security measures.

3. User-Friendly Interface: Create an intuitive interface for easy navigation and user adoption.

4. Efficient Reporting Mechanism: Implement a robust reporting mechanism for monitoring and analyzing pharmaceutical data.

5. Modular and Scalable Architecture: Design a modular and scalable architecture for future enhancements and adaptability to evolving pharmacy requirements.

1.5 Application

The Pharmaceutical Management System optimizes medicine inventory and ensures regulatory compliance through automated reporting. Its user-friendly interface streamlines daily tasks for pharmacy staff, while the modular architecture accommodates future enhancements, making it a valuable solution for efficient pharmaceutical management.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

Start the section with a general discussion of the project [1] [2] [3].

2.2 Project Details

The Pharmaceutical Management System is a comprehensive solution designed to streamline the operations of pharmacies or pharmaceutical companies. This system aims to efficiently manage inventory, customer records, employee details, and sales processes while ensuring compliance with healthcare regulations. Key features include inventory management to track stock levels and expiry dates, a customer module to maintain prescription and purchase history, and an employee module to manage staff details. The system also automates billing, generates detailed invoices, and provides sales and inventory reports for auditing and compliance. it ensures data consistency and accuracy, connecting tables like Medicines, Customers, Sales, and Employees through relational schemas. By integrating modern technologies and optional user interfaces, this project enhances efficiency, reduces errors, and modernizes traditional pharmacy operations into a digital framework.

2.3 Implementation

```
package pharmaceutical;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

class Medicine {
   String name;
   String manufacturer;
   float price;
   int quantity;

   Medicine(String name, String manufacturer, float price, int quantity) {
      this.name = name;
```

```
        this.manufacturer = manufacturer;
        this.price = price;
        this.quantity = quantity;
    }
}

public class PharmaceuticalManagementSystem {

    private List<Medicine> medicines = new ArrayList<>();
    private JFrame frame;
    private JTable table;
    private DefaultTableModel tableModel;

    public PharmaceuticalManagementSystem() {
        initializeGUI();
        addDefaultMedicines();
        updateTable();
    }

    private void initializeGUI() {
        frame = new JFrame("Pharmaceutical Management System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);

        // Table
        tableModel = new DefaultTableModel(new String[]{"Name", "Manufacturer",
"Price", "Quantity"}, 0);
        table = new JTable(tableModel);
        JScrollPane scrollPane = new JScrollPane(table);

        // Buttons
        JButton addButton = new JButton("Add Medicine");
        JButton updateButton = new JButton("Update Medicine");
        JButton checkStockButton = new JButton("Check Stock");
        JButton sortByNameButton = new JButton("Sort by Name");
        JButton sortByPriceButton = new JButton("Sort by Price (Merge Sort)");
        JButton selectionSortButton = new JButton("Sort by Manufacturer (Selection
Sort)");
        JButton quickSortButton = new JButton("Sort by Quantity (Quick Sort)");

        // Button Actions
        addButton.addActionListener(e -> addMedicine());
        updateButton.addActionListener(e -> updateMedicine());
        checkStockButton.addActionListener(e -> checkStock());
```

```
        sortByNameButton.addActionListener(e -> {
            sortMedicinesByName();
            updateTable();
        });
        sortByPriceButton.addActionListener(e -> {
            sortMedicinesByPrice();
            updateTable();
        });
        selectionSortButton.addActionListener(e -> {
            selectionSortByManufacturer();
            updateTable();
        });
        quickSortButton.addActionListener(e -> {
            quickSortByQuantity(0, medicines.size() - 1);
            updateTable();
        });

        JPanel buttonPanel = new JPanel();
        buttonPanel.add(addButton);
        buttonPanel.add(updateButton);
        buttonPanel.add(checkStockButton);
        buttonPanel.add(sortByNameButton);
        buttonPanel.add(sortByPriceButton);
        buttonPanel.add(selectionSortButton);
        buttonPanel.add(quickSortButton);

        frame.add(scrollPane, BorderLayout.CENTER);
        frame.add(buttonPanel, BorderLayout.SOUTH);
        frame.setVisible(true);
    }

    private void addDefaultMedicines() {
        medicines.add(new Medicine("Paracetamol", "XYZ Pharmaceuticals", 10.5f, 100));
        medicines.add(new Medicine("Ibuprofen", "ABC Pharma", 8.75f, 50));
        medicines.add(new Medicine("Aspirin", "XYZ Pharmaceuticals", 7.25f, 75));
    }

    private void addMedicine() {
        String name = JOptionPane.showInputDialog(frame, "Enter medicine name:");
        String manufacturer = JOptionPane.showInputDialog(frame, "Enter
manufacturer:");
        String priceStr = JOptionPane.showInputDialog(frame, "Enter price:");
        String quantityStr = JOptionPane.showInputDialog(frame, "Enter quantity:");
```

```
        try {
            float price = Float.parseFloat(priceStr);
            int quantity = Integer.parseInt(quantityStr);
            medicines.add(new Medicine(name, manufacturer, price, quantity));
            updateTable();
            JOptionPane.showMessageDialog(frame, "Medicine added successfully.");
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(frame, "Invalid input for price or quantity.");
        }
    }

    private void updateMedicine() {
        String name = JOptionPane.showInputDialog(frame, "Enter the name of the medicine to update:");
        for (Medicine med : medicines) {
            if (med.name.equalsIgnoreCase(name)) {
                String quantityStr = JOptionPane.showInputDialog(frame, "Enter the new quantity:");
                try {
                    med.quantity = Integer.parseInt(quantityStr);
                    updateTable();
                    JOptionPane.showMessageDialog(frame, "Medicine updated successfully.");
                    return;
                } catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(frame, "Invalid input for quantity.");
                    return;
                }
            }
        }
        JOptionPane.showMessageDialog(frame, "Medicine not found.");
    }

    private void checkStock() {
        String name = JOptionPane.showInputDialog(frame, "Enter the name of the medicine to check stock:");
        for (Medicine med : medicines) {
            if (med.name.equalsIgnoreCase(name)) {
                JOptionPane.showMessageDialog(frame, "Stock for " + med.name + ": " + med.quantity);
                return;
            }
        }
        JOptionPane.showMessageDialog(frame, "Medicine not found.");
```

```
    }

    private void sortMedicinesByName() {
        medicines.sort(Comparator.comparing(med -> med.name));
    }

    private void sortMedicinesByPrice() {
        mergeSort(medicines, 0, medicines.size() - 1, Comparator.comparingDouble(med
-> med.price));
    }

    private void mergeSort(List<Medicine> list, int left, int right, Comparator<Medicine>
comparator) {
        if (left < right) {
            int mid = left + (right - left) / 2;
            mergeSort(list, left, mid, comparator);
            mergeSort(list, mid + 1, right, comparator);
            merge(list, left, mid, right, comparator);
        }
    }

    private void merge(List<Medicine> list, int left, int mid, int right,
Comparator<Medicine> comparator) {
        List<Medicine> leftList = new ArrayList<>(list.subList(left, mid + 1));
        List<Medicine> rightList = new ArrayList<>(list.subList(mid + 1, right + 1));

        int i = 0, j = 0, k = left;
        while (i < leftList.size() && j < rightList.size()) {
            if (comparator.compare(leftList.get(i), rightList.get(j)) <= 0) {
                list.set(k++, leftList.get(i++));
            } else {
                list.set(k++, rightList.get(j++));
            }
        }

        while (i < leftList.size()) {
            list.set(k++, leftList.get(i++));
        }

        while (j < rightList.size()) {
            list.set(k++, rightList.get(j++));
        }
    }
```

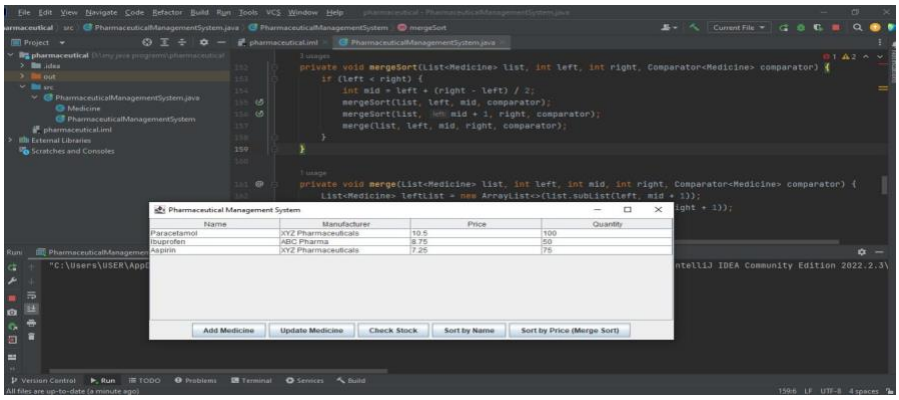
```
    private void selectionSortByManufacturer() {
        int n = medicines.size();
        for (int i = 0; i < n - 1; i++) {
            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if
(medicines.get(j).manufacturer.compareTo(medicines.get(minIndex).manufacturer) <
0) {
                    minIndex = j;
                }
            }
            Medicine temp = medicines.get(minIndex);
            medicines.set(minIndex, medicines.get(i));
            medicines.set(i, temp);
        }
    }

    private void quickSortByQuantity(int low, int high) {
        if (low < high) {
            int pi = partition(low, high);
            quickSortByQuantity(low, pi - 1);
            quickSortByQuantity(pi + 1, high);
        }
    }

    private int partition(int low, int high) {
        int pivot = medicines.get(high).quantity;
        int i = (low - 1);
        for (int j = low; j < high; j++) {
            if (medicines.get(j).quantity <= pivot) {
                i++;
                Medicine temp = medicines.get(i);
                medicines.set(i, medicines.get(j));
                medicines.set(j, temp);
            }
        }
        Medicine temp = medicines.get(i + 1);
        medicines.set(i + 1, medicines.get(high));
        medicines.set(high, temp);
        return i + 1;
    }

    private void updateTable() {
        tableModel.setRowCount(0);
```

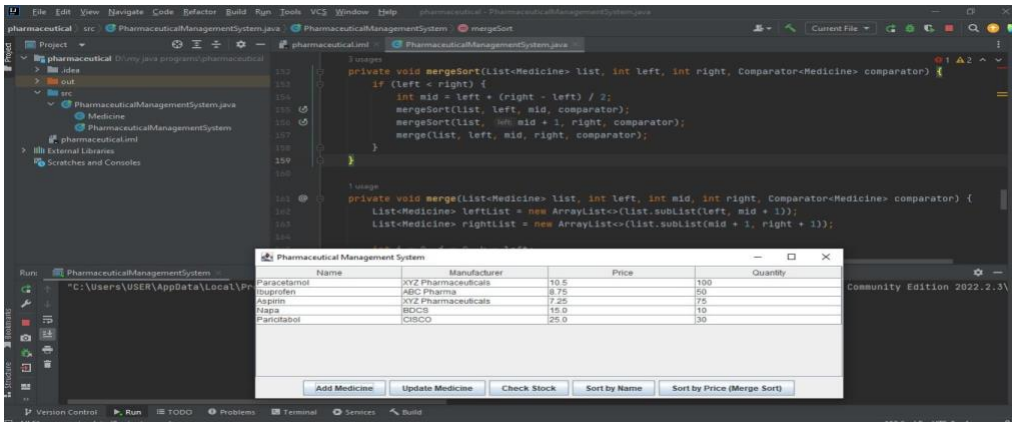
```
        for (Medicine med : medicines) {
            tableModel.addRow(new Object[]{med.name, med.manufacturer, med.price,
med.quantity});
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(PharmaceuticalManagementSystem::new);
    }
}
```

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure





3.2 Results Analysis/Testing

3.3 Results Overall Discussion

The Pharmaceutical Management System effectively addresses the critical needs of modern pharmacies by ensuring streamlined operations, improved accuracy, and enhanced efficiency. Its ability to manage inventory, track expiration dates, and maintain detailed records of customers and employees reduces manual effort and minimizes errors. The automated billing and reporting features provide insights into sales and inventory trends, enabling informed decision-making. By integrating a robust database structure and modern technologies, the system ensures data consistency, security, and scalability. Overall, the project demonstrates the potential to transform traditional pharmaceutical management practices into a more organized and digital approach, catering to the growing demands of the healthcare industry.

Chapter 4

Conclusion

4.1 Discussion

The development of the Pharmaceutical Management System highlights the importance of automating and digitizing traditional pharmacy operations to meet modern demands. The system addresses key challenges such as inefficient inventory tracking, manual record-keeping, and inaccuracies in billing. By implementing modules for inventory management, sales tracking, and employee and customer record maintenance, the system ensures operational efficiency and accuracy. Furthermore, the integration of relational databases enables seamless data management, reducing redundancy and ensuring data consistency. The ability to generate detailed reports and maintain compliance with healthcare standards adds significant value to the system. This project not only enhances the day-to-day operations of pharmacies but also provides a scalable and secure solution for future

4.2 Limitations

Despite its numerous advantages, the Pharmaceutical Management System has some limitations. One major challenge is the initial setup, which requires time and technical expertise for database design and integration. The system heavily relies on accurate data entry, and any incorrect or incomplete information could lead to errors in inventory management or billing. Additionally, the system may require regular maintenance and updates to address evolving healthcare regulations and business needs. For small-scale pharmacies, the cost of implementation and training might be a barrier. Finally, the absence of advanced features like real-time analytics, mobile application support, or integration with external healthcare systems limits its adaptability in more complex or large-scale environments.

4.3 Scope of Future Work

The Pharmaceutical Management System has significant potential for future enhancements and scalability. Integrating real-time analytics and predictive tools can provide insights into sales trends, demand forecasting, and inventory optimization. Adding mobile and cloud-based support would enable remote access and enhance the system's usability for both employees and customers. Advanced features like integration with external healthcare systems, such as electronic medical records (EMR) or telemedicine platforms, can further improve efficiency and interoperability. Incorporating AI-powered tools for automated data validation and chatbot support for customer queries can enhance user experience and reduce errors. Additionally, expanding the system's capabilities to support multi-branch pharmacies and incorporating multi-language support can make it adaptable for larger-scale or international operations. These enhancements would make the system more robust, user-friendly, and future-ready.

References

[1] Uthayasankar Sivarajah, Muhammad Mustafa Kamal, Zahir Irani, and Vishanth Weerakkody. Critical analysis of big data challenges and analytical methods. *Journal of Business Research*, 70:263–286, 2017.

[2] Douglas Laney. 3d data management: controlling data volume, velocity and variety. gartner, 2001.

[3] MSWindows NT kernel description. http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm. Accessed Date: 2010-09-30.