STOCK MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

DIVYA P(2303811710422035)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

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

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"STOCK MANAGEMENT SYSTEM"** is the bonafide work of **DIVYA P (2303811710422035)** who carriedout the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

[illegible]

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

CGB1201-JAVA PROGRAMMING
Mr. M. SARAVANAN, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR

[illegible]

SIGNATURE

Mr. M. Saravanan, M.E.,

SUPERVISOR

ASSISTANTP ROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 02.12.2024

CGB1201-JAVA PROGRAMMING
Mr.MALARMANNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

[illegible]

INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Dr.K.SENTHAMILSELVI, M.E., Ph.D.,
EXTERNAL EXAMINER
PROFESSOR
8138-SCE, TRICHY.

[illegible]

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“STOCK MANAGEMENT SYSTEM”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201- JAVA PROGRAMMING.**

.

P. Divya.

Signature

DIVYA P

Place: Samayapuram

Date: 02.12.2024

ACKNOWLEDGEMENT

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

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for 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 to our project and offering adequate duration in completing our project.

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

I whole heartily thanks to **Dr A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr. M. SARAVANAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

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

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

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.

ABSTRACT

A Stock Management System is a Java-based software application designed to streamline and automate inventory management processes for businesses such as retail stores, warehouses, or any organization dealing with physical goods. This system enables real-time tracking of stock levels, ensuring accurate monitoring to prevent overstocking or stockouts. It includes features for managing product details, supplier information, and sales records, providing businesses with a comprehensive solution for inventory control. The application is developed using Java for the backend, ensuring robust, platform-independent performance, and utilizes JDBC to interact with a relational database for efficient data storage. The user interface, built with Java Swing or JavaFX, ensures a user-friendly experience for non-technical users. Additional functionalities such as low-stock alerts, secure user authentication, and sales tracking enhance operational efficiency and decision-making. This system offers a scalable foundation, with the potential to integrate advanced features like barcode scanning and demand forecasting to meet evolving business needs.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
A Stock Management System is a Java-based software application designed to streamline and automate inventory management processes for businesses such as retail stores, warehouses, or any organization dealing with physical goods. This system enables real-time tracking of stock levels, ensuring accurate monitoring to prevent overstocking or stockouts. It includes features for managing product details, supplier information, and sales records, providing businesses with a comprehensive solution for inventory control. The application is developed using Java for the backend, ensuring robust, platform-independent performance, and utilizes JDBC to interact with a relational database for efficient data storage.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of this project is to design and develop a comprehensive stock management system using Java programming language. The system aims to provide an efficient and user-friendly platform for managing stock inventory, tracking stock movements, and generating reports. The system should be able to handle a large number of stock items, track stock levels in real-time, and provide alerts and notifications when stock levels reach a certain threshold. Additionally, the system should provide a secure and scalable solution for managing stock inventory, and should be able to integrate with other systems and applications.The stock management system is a Java-based application that enables users to manage and monitor stock inventory levels. The system allows users to add, update, and delete stock items, as well as track stock movements, such as purchases, sales, and stock transfers. The system also generates reports on stock levels, stock movements, and other relevant information. The system consists of several modules, including stock management, report generation, and user management. The system uses a relational database management system, such as MySQL, to store and retrieve stock data. The system also uses Java Swing and JavaFX for designing and implementing the system's graphical user interface (GUI). The system is designed to be scalable, secure, and user-friendly, and provides a comprehensive solution for managing stock inventory.

1.2 Overview

The stock management system utilizes several key Java programming concepts, including Object-Oriented Programming (OOP) concepts, Java Collections Framework (JCF), Java Database Connectivity (JDBC), Java Swing, and JavaFX. The system uses OOP concepts, such as encapsulation, inheritance, and polymorphism, to design and implement the system's classes and objects. The system uses JCF to store and manage stock items, including Array List, HashMap, and Tree Set. The system uses JDBC to interact with a relational database management system, such as MySQL, to store and retrieve stock data. The system also uses Java Swing and JavaFX for designing and implementing the system's graphical user interface (GUI).

1.3 Java Programming Concepts

1. Object-Oriented Programming (OOP)

- ✓ **Encapsulation:** Protect and manage stock-related data using private fields and public methods.
- ✓ **Inheritance:** Create specialized stock item types (e.g., perishable, electronics) using base and derived classes.
- ✓ **Polymorphism:** Implement method overriding for behavior customization (e.g., calculating expiry for perishable items).
- ✓ **Abstraction:** Define abstract classes or interfaces for generic operations like adding, deleting, or updating stock items.

2. Collections Framework

- ✓ Use dynamic data structures like ArrayList, HashMap, or Set to manage stock items, categories, or supplier details.
- ✓ Iterate and filter collections using for-each loops, iterators, or streams for efficient processing.

3. Exception Handling

- ✓ Handle errors gracefully using try-catch blocks.
- ✓ Create custom exceptions for inventory-specific scenarios, such as invalid quantity or duplicate entries.

4. File Handling

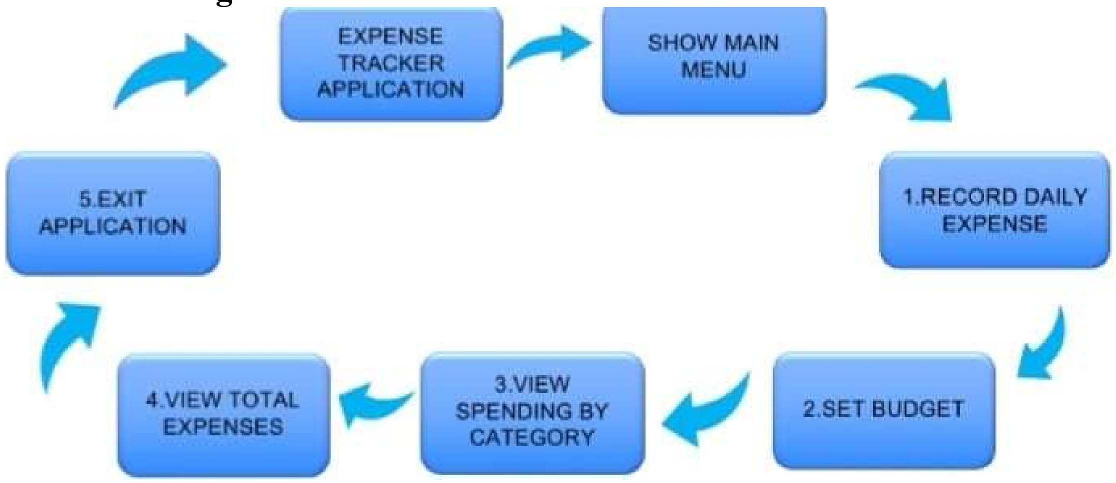
- ✓ Use file I/O operations to save or retrieve inventory data when a database is unavailable.
- ✓ Implement serialization to save object states for later use.
- ✓ methods to access them, ensuring controlled access to the data.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed Stock Management System aims to streamline inventory operations by automating the processes of stock tracking, management, and reporting. The system will address common challenges such as overstocking, stockouts, and manual errors, ensuring efficient inventory control and accurate data management. It will include features like stock entry and updates, real-time inventory tracking, low-stock alerts, and order management to facilitate seamless handling of stock movements. Additionally, the system will generate detailed reports and analytics to provide insights into inventory trends, aiding in better decision-making and demand forecasting.To ensure secure and efficient operations, the system will incorporate role-based access, allowing only authorized users to manage stock data. Technologies such as a user-friendly frontend (using HTML, CSS, and JavaScript), a robust backend (using Node.js, Python, or Java), and a reliable database (MySQL, PostgreSQL, or MongoDB) will be employed. The solution will leverage cloud services for scalability and reliability, integrating with other systems like accounting or e-commerce platforms if required. By automating inventory operations, the proposed system is expected to reduce manual errors, enhance operational efficiency, and improve stock visibility, resulting in better resource utilization and customer satisfaction.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Add stock item

The Add Stock Item functionality allows users to input detailed information about new inventory items. This includes essential data such as the item name, Stock Keeping Unit (SKU), category, initial quantity, unit price, supplier details, and any other relevant attributes. Once entered, this information is validated and securely stored in the database, ensuring the addition of accurate and consistent records. This feature helps businesses keep track of every new product introduced into their inventory, forming the foundation of the stock management system.

3.2 View stock item

The View Stock Item functionality provides users with a comprehensive and real-time overview of all stock items currently in the system. It includes information such as item names, SKUs, categories, quantities available, unit prices, supplier details, and stock status (e.g., low stock, out of stock). This feature also incorporates search, filter, and sorting options, enabling users to quickly locate specific stock items based on various criteria.

3.3 Update stock item

The Update Stock Item functionality is critical for maintaining accurate inventory records. Users can modify existing stock details, such as updating quantities to reflect sales or new purchases, revising pricing to account for market changes, or editing product descriptions and categories to improve record accuracy. This feature ensures that the system always reflects the current state of inventory, minimizing discrepancies and enhancing operational efficiency.

3.4 Delete stock item

The Delete Stock Item functionality enables users to remove outdated, discontinued, or incorrect inventory records from the system. Before deletion, the system prompts users for confirmation and may also check dependencies, such as whether the item is part of an ongoing order or transaction, to avoid accidental removal of critical data. This feature helps maintain a clean and relevant inventory database, reducing clutter and improving system performance.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The development of a Stock Management System is a critical step towards ensuring efficient inventory control and streamlined business operations. By automating key processes such as stock addition, updates, tracking, and reporting, the system minimizes manual errors, reduces operational inefficiencies, and enhances decision-making. It provides real-time visibility into inventory levels, ensuring that stockouts or overstocking are effectively avoided, which ultimately contributes to better resource utilization and customer satisfaction. The integration of robust programming concepts like object-oriented design, database connectivity, exception handling, and user-friendly interfaces ensures that the system is both scalable and easy to maintain. Additionally, advanced features like real-time alerts, analytics, and role-based access control enhance the system's utility and security. Over all, the Stock Management System not only simplifies inventory management but also lays a strong foundation for data-driven decision-making, helping businesses remain competitive in a dynamic market.

4.2 FUTURE SCOPE

The future scope of the Stock Management System lies in its ability to evolve with emerging technologies and business requirements. Integrating advanced tools like Artificial Intelligence (AI) and Machine Learning (ML) can enable the system to predict demand trends, optimize inventory levels, and provide actionable insights. The incorporation of Internet of Things (IoT) devices, such as RFID tags and smart sensors, can facilitate real-time tracking of stock conditions and automated updates. Transitioning to a cloud-based architecture can enhance accessibility, scalability, and reliability, while ensuring data security through disaster recovery and backup solutions.

Mobile application development can extend the system’s functionality, enabling staff to manage inventory on the go with features like barcode scanning and instant stock updates. Enhanced security measures, such as multi-factor authentication and data encryption, can safeguard sensitive inventory information. Integration with e-commerce platforms allows for real-time synchronization of stock levels, preventing overselling and improving customer satisfaction. Additionally, advanced analytics and reporting tools can offer dynamic dashboards and customizable insights, supporting better decision-making.

Expanding the system to include automated supply chain management and ERP integration can streamline operations across procurement, finance, and logistics. Furthermore, global features like multi-currency and multi-language support can make the system suitable for businesses operating in diverse markets. The adoption of blockchain technology can enhance transparency and traceability, particularly for industries requiring strict compliance, such as pharmaceuticals or food. Overall, these advancements position the Stock Management System as a robust, scalable, and indispensable tool for future business needs.

REFERENCES

Books

1. **"Head First Java"** by Kathy Sierra and Bert Bates
    - A beginner-friendly book for understanding Java programming concepts used in stock management systems.
2. **"Database System Concepts"** by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan
    - Covers database design and management, crucial for the backend of inventory systems.
3. **"Software Engineering: A Practitioner's Approach"** by Roger S. Pressman
    - Provides a comprehensive overview of software development processes applicable to building systems like this.
4. **"Design Patterns: Elements of Reusable Object-Oriented Software"** by Erich Gamma et al.
    - Helpful for implementing scalable design patterns in a stock management system.

Web Resources

1. **Java Tutorials by Oracle**
    - https://docs.oracle.com/javase/tutorial/
      Offers official Java tutorials covering topics like OOP, collections, and JDBC.

2. **GeeksforGeeks – Inventory Management System Tutorials**
    - https://www.geeksforgeeks.org/
      Provides programming guides and examples related to inventory and stock management systems.
3. **W3Schools Java and SQL Tutorials**
    - https://www.w3schools.com/
      Introduces basic and advanced concepts of Java programming and SQL databases.
4. **Stack Overflow**
    - https://stackoverflow.com/
      A platform for resolving technical issues and learning from real-world coding examples.

Research Papers

1. "Inventory Management System: A Study of the Impact of Automation in Small Businesses" – Available on **ResearchGate** or similar academic repositories.
2. "The Role of Technology in Modern Inventory Control Systems" – Provides insights into technologies like IoT and AI in inventory systems.

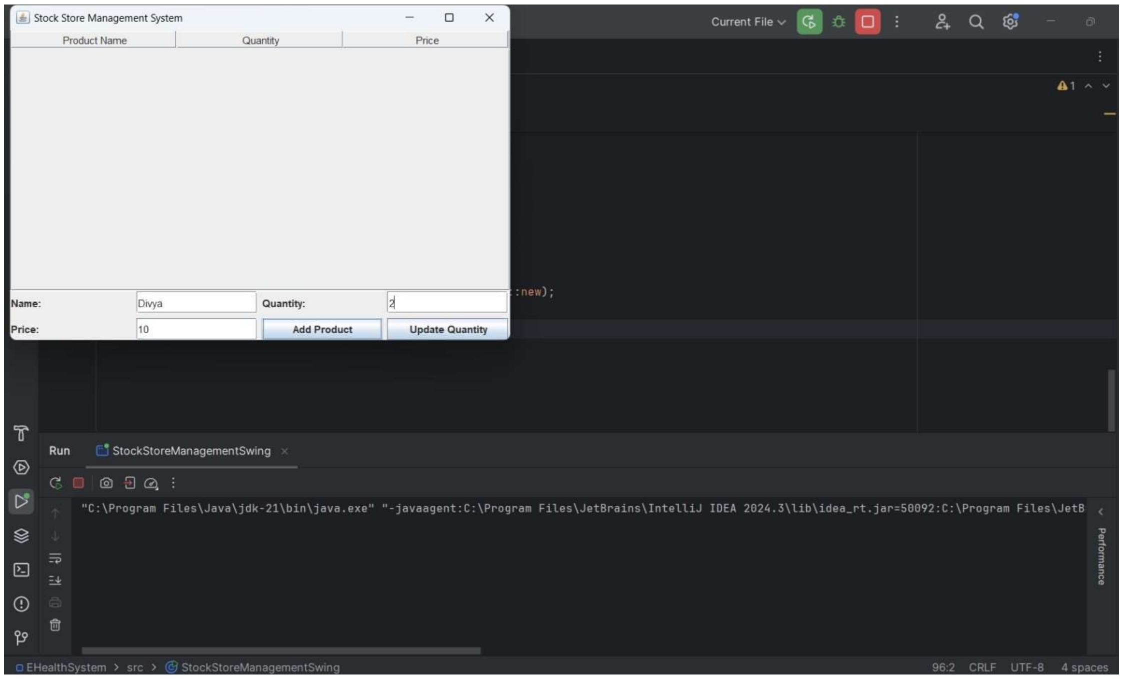
Tools and Framework Documentation

1. **MySQL Documentation**
    - https://dev.mysql.com/doc/
      Official documentation for database setup and operations.
2. **Apache Maven Documentation**
    - https://maven.apache.org/guides/
      Guide for managing dependencies in Java projects.
3. **Spring Framework Documentation**
    - https://spring.io/guides
      A valuable resource for building robust backend systems using Spring Boot.

APPENDIX A (SOURCE CODE)

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class StockStoreManagementSwing {
    private DefaultTableModel tableModel;

    public StockStoreManagementSwing() {
        // Main frame
        JFrame frame = new JFrame("Stock Store Management System");
        frame.setSize(600, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        // Table for displaying products
        String[] columns = {"Product Name", "Quantity", "Price"};
        tableModel = new DefaultTableModel(columns, 0);
        JTable table = new JTable(tableModel);
        JScrollPane tableScrollPane = new JScrollPane(table);
        frame.add(tableScrollPane, BorderLayout.CENTER);

        // Input panel
        JPanel inputPanel = new JPanel(new GridLayout(2, 4, 5, 5));
        JTextField nameField = new JTextField();
        JTextField quantityField = new JTextField();
        JTextField priceField = new JTextField();
        JButton addButton = new JButton("Add Product");
        JButton updateButton = new JButton("Update Quantity");
inputPanel.add(new JLabel("Name:"));
        inputPanel.add(nameField);
        inputPanel.add(new JLabel("Quantity:"));
        inputPanel.add(quantityField);
        inputPanel.add(new JLabel("Price:"));
        inputPanel.add(priceField);
        inputPanel.add(addButton);
        inputPanel.add(updateButton);

        frame.add(inputPanel, BorderLayout.SOUTH);

        // Add product button action
        addButton.addActionListener(e -> {
            String name = nameField.getText();
            String quantityText = quantityField.getText();
            String priceText = priceField.getText();

            if (name.isEmpty() || quantityText.isEmpty() || priceText.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "All fields are required!", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }
try {
            int quantity = Integer.parseInt(quantityText);
            double price = Double.parseDouble(priceText);
            tableModel.addRow(new Object[]{name, quantity, price});
            nameField.setText("");
            quantityField.setText("");
            priceField.setText("");
            JOptionPane.showMessageDialog(frame, "Product added successfully!");
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(frame, "Invalid quantity or price format!",
"Error", JOptionPane.ERROR_MESSAGE);
        }
    });

    // Update product quantity button action
    updateButton.addActionListener(e -> {
        int selectedRow = table.getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(frame, "Please select a product to update!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        String newQuantityText = quantityField.getText();
        if (newQuantityText.isEmpty()) {
            JOptionPane.showMessageDialog(frame, "Please enter a new quantity!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
try {
            int newQuantity = Integer.parseInt(newQuantityText);
            tableModel.setValueAt(newQuantity, selectedRow, 1);
            quantityField.setText("");
            JOptionPane.showMessageDialog(frame, "Quantity updated successfully!");
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(frame, "Invalid quantity format!", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    });

    // Display the frame
    frame.setVisible(true);
  }

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

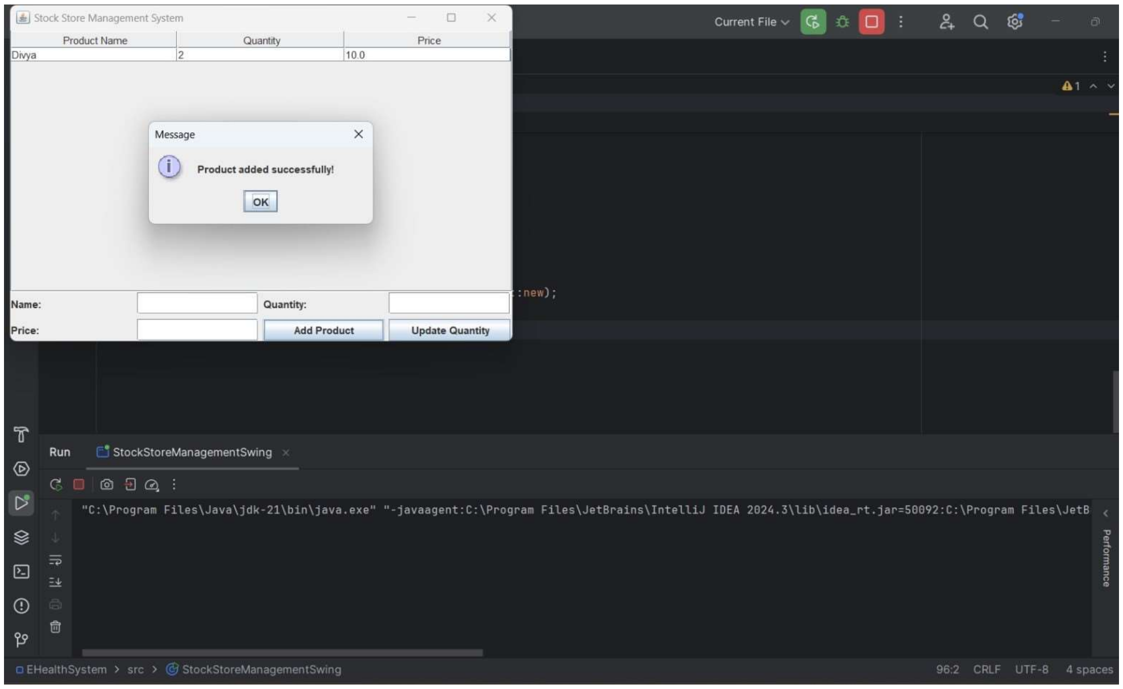
APPENDIX B(SCREENSHOTS)


Stock Store Management System
Current File
Add Product
Update Quantity
Run
StockStoreManagementSwing
Performance
EHealthSystem > src > StockStoreManagementSwing
96:2 CRLF UTF-8 4 spaces

Product Name	Quantity	Price

Name: Divya
Quantity: 2
Price: 10

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3\lib\idea_rt.jar=50092:C:\Program Files\JetB
```


Stock Store Management System
Current File
Message
OK
Add Product
Update Quantity
Run
StockStoreManagementSwing
Performance
EHealthSystem > src > StockStoreManagementSwing
96:2 CRLF UTF-8 4 spaces

Product Name	Quantity	Price
Divya	2	10.0

Product added successfully!

Name:
Quantity:
Price:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3\lib\idea_rt.jar=50092:C:\Program Files\JetB
```