STOCK MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

SHIBANI S (2303811710421147)

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 **SHIBANI S (2303811710421147)** who carried out 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]	CGB1201-JAVA PROGRAMMING Mr.MALARMANNAN A, M.E., ASSISTANT PROFESSOR [illegible]
SIGNATURE	**SIGNATURE**
Mrs.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr. A. Malarmannan, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on – 06.12.24

CGB1201-JAVA PROGRAMMING Mr. M. SARAVANAN, M.E., INTERNAL EXAMINER ASSISTANT PROFESSOR [illegible]	CGB1201-JAVA PROGRAMMING Mr.R. KARTHIK, M.E., EXTERNAL EXAMINER ASSISTANT PROFESSOR 8138-SCE, TRICHY. [illegible] 06/12/2024
INTERNAL EXAMINER	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 **BACHELOROF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING.**

Signature

[illegible]

SHIBANI S

Place: Samayapuram

Date: 6.12.24

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. A. MALARMANNAN, 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

The Stock Management System project is designed to develop an efficient, automated solution for managing and tracking inventory within businesses. It aims to optimize stock levels, manage product orders, and streamline the entire inventory management process to ensure accuracy, minimize errors, and improve operational efficiency. The system allows businesses to monitor the quantity of products, track stock movements, generate reports, and set automatic reorder alerts to avoid stockouts or overstocking.

This system includes several key features such as user-friendly dashboards, barcode scanning integration for quick product identification, supplier management, and real-time updates on stock levels. It also provides data analytics and reporting tools that offer insights into inventory trends, sales patterns, and stock performance. The system is designed to reduce the time spent on manual inventory management, ensuring that stock data is always accurate and up-to-date.

The Stock Management System can be integrated with other business operations like sales, purchasing, and accounting systems, ensuring seamless data flow across different departments. It is built to support businesses of various sizes, from small enterprises to large corporations, providing scalability and flexibility to meet specific needs.

This project seeks to deliver an intuitive, reliable, and scalable solution to help organizations improve inventory control, reduce operational costs, and enhance overall productivity. The development of this Stock Management System leverages modern technologies to create a robust and secure platform that facilitates easy management and optimization of business stock.

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 an essential software solution designed to streamline and automate the tracking, control, and management of inventory in businesses. The system is primarily aimed at helping organizations manage their stock levels, orders, sales, and deliveries with improved accuracy and efficiency. By providing real-time tracking and reporting, it minimizes the risk of stockouts, overstocking, and wastage, thus optimizing overall operational costs. The Stock Management System provides functionalities such as inventory monitoring, order management, supplier tracking, and product categorization. It also includes features like barcode scanning for easy product identification, stock replenishment alerts, and detailed reporting for analysis. Additionally, the system can integrate with other business processes, including accounting and sales management, to offer a comprehensive view of the business's performance. The system offers a user-friendly interface and ensures data accuracy by reducing human errors in stock handling and inventory updates. With its ability to automate inventory tasks, the Stock Management System significantly improves productivity, enhances decision-making, and supports the smooth functioning of businesses across various industries. In conclusion, the Stock Management System is a vital tool for businesses aiming to optimize their stock control process, reduce manual intervention, and increase profitability through better resource allocation and inventory management.	**PO1 -3** **PO2 -3** **PO3 -3** **PO5 -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 objective of the Stock Management System project is to develop a comprehensive and automated platform that enables businesses to efficiently track, manage, and optimize their inventory. The system aims to streamline stock control processes by providing real-time updates on stock levels, automating stock replenishment alerts, managing orders, and generating detailed reports for analysis. The goal is to create a reliable, scalable solution that improves business productivity and reduces operational costs associated with inventory management.

1.2 Overview

The Stock Management System is an automated software solution designed to simplify and streamline the management of inventory within organizations. It provides businesses with the tools necessary to efficiently track stock levels, manage orders, and handle supplier information, ensuring that inventory is always accurately monitored. The system helps to reduce manual intervention, errors, and inefficiencies typically associated with traditional stock management processes.

The core functionality of the system includes real-time inventory tracking, automated stock updates, order processing, and stock-level alerts to prevent both stockouts and overstocking. Users can categorize products, manage suppliers, and generate detailed reports on stock performance, sales trends, and purchasing activities. Additionally, the system integrates barcode scanning to quickly identify products, making the process faster and more accurate. Overall, the Stock Management System serves as a powerful tool to optimize inventory management, improve productivity, and support the growth and efficiency of businesses in various industries.

1.3 Java Programming Concepts

In the development of a Stock Management System using Java, several programming concepts and technologies are commonly employed to ensure functionality, efficiency, and maintainability. These include:

1. **Object-Oriented Programming (OOP):**
    - **Classes and Objects**: Fundamental to Java, the Stock Management System utilizes classes (e.g., Product, Order, Supplier) to define the properties and behaviors associated with each entity. Objects are instances of these classes.
    - **Encapsulation**: Data such as stock quantity, product name, and price is encapsulated within classes, and access is controlled through getter and setter methods to maintain data integrity.
    - **Inheritance**: In case of any extended functionality (e.g., PerishableProduct inherits from Product), inheritance is used to avoid code duplication and enhance reusability.
    - **Polymorphism**: Different types of products (e.g., regular products, perishable products) can be treated as objects of the base Product class but behave differently, enhancing flexibility in the system.
2. **Collections Framework:**
    - **Lists, Sets, Maps**: Java's Collections Framework is used to manage and store data, such as inventory items, orders, and suppliers. For example, an ArrayList can be used to store the list of products, and a HashMap might be used to map product IDs to product objects for quick lookup.
    - **Iterator**: The Iterator interface is used for traversing through collections, especially when managing inventory lists or supplier data.
3. **Exception Handling:**
    - Proper exception handling ensures that the system gracefully handles errors such as invalid input, stock level issues (e.g., negative quantities), and database connection failures. Java's try-catch blocks and custom exceptions improve the system's robustness.

4. **File Handling/Database Connectivity:**
    - **File I/O**: To persist data like inventory and transaction records, Java's file handling features (e.g., FileReader, BufferedReader, FileWriter) might be used for reading from and writing to text files.
    - **JDBC (Java Database Connectivity)**: For larger systems, a database is used to store stock information, order details, and transaction logs. Java's JDBC API connects the system to a database (e.g., MySQL) for executing SQL queries and updating records.
5. **User Interface (UI):**
    - **Swing/JavaFX**: For creating a graphical user interface (GUI), Java Swing or JavaFX can be used to develop interactive forms, buttons, tables, and other UI components for ease of use and efficient interaction with the system.
    - **Event Handling**: Java's event handling mechanisms (e.g., ActionListener) are used to capture and respond to user actions, such as clicking buttons or entering data into fields.
6. **Multi-threading:**
    - **Threading**: Java’s multi-threading capabilities are used to handle background tasks like updating stock levels, processing orders, or generating reports without freezing the main user interface.
7. **Java API Libraries:**
    - Various Java libraries and utilities (e.g., java.util.Date, java.text.SimpleDateFormat) are used to handle dates, time, and formatting requirements for orders and transactions.

By utilizing these Java programming concepts, the Stock Management System can be built to be efficient, maintainable, and scalable, ensuring it can handle the complexities of inventory management in real-world scenarios.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the Stock Management System involves developing an automated solution to streamline inventory tracking, order processing, and stock control for businesses. The system will be designed to monitor stock levels in real-time, manage product details, process customer orders, and generate reports for decision-making. The aim is to improve inventory efficiency, reduce errors, and optimize stock management processes, leading to better operational control and cost reduction.

2.2 Block Diagram

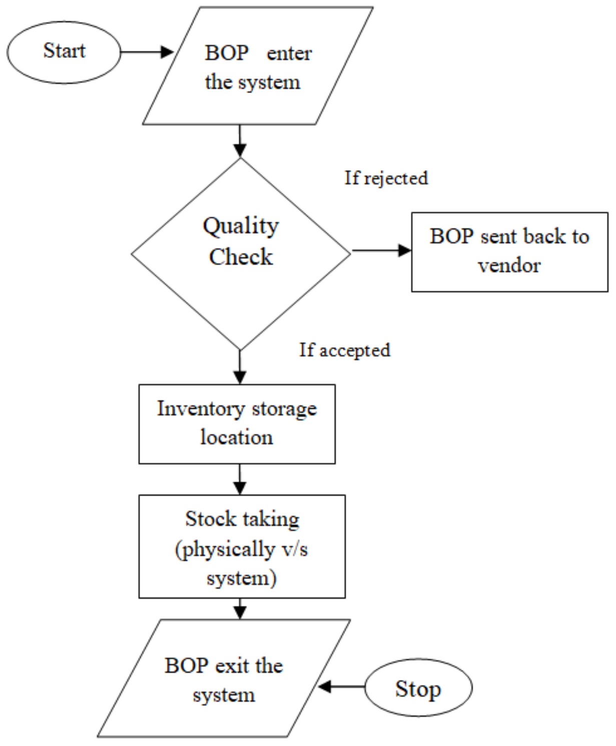


Fig – 2.1 Block Diagram

Fig – 2.1

CHAPTER 3
MODULE DESCRIPTION

3.1 Inventory Management Module

This module is the core of the system, designed to manage and track stock levels in real-time. It allows users to add, update, and delete products from the inventory. Key functionalities include maintaining product details such as name, price, stock quantity, and category

3.2 Order Management Module

The order management module helps track and manage customer orders. It allows users to create new orders, update existing ones, and monitor the order status. This module integrates with the inventory management system to update stock quantities automatically upon order fulfillment.

3.3 Supplier Management Module

This module enables the management of supplier details, including contact information, pricing, and order history. Users can add, update, and remove suppliers, ensuring a smooth process for ordering new stock. The module can also be linked to the order management module to automatically place orders with suppliers when stock levels are low.

3.4 Reporting and Analytics Module

The reporting module generates detailed reports and analytics to support decision-making. It includes inventory reports, order summaries, sales trends, and low-stock alerts. Users can customize reports based on date ranges, product categories, or suppliers.

3.5 User Authentication and Role Management Module

This module ensures secure access to the system by implementing user authentication and role-based access control. Users are required to log in with a username and password, and their access rights are defined based on their roles (e.g., Admin, Sales, Inventory Manager).

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Stock Management System project aims to offer a robust, automated solution for businesses to efficiently manage their inventory, streamline order processing, and enhance decision-making. By utilizing Java and applying key programming concepts such as Object-Oriented Programming (OOP), database connectivity, and modular design, the system ensures seamless integration and smooth operation of all stock management processes.

Throughout the development of the system, key features were incorporated to meet the core objectives of efficiency and accuracy in inventory management. These features include real-time tracking of stock levels, automated stock alerts to prevent understocking or overstocking, and the integration of barcode scanning to simplify product identification and entry. Additionally, the system offers detailed reporting and analytics capabilities, which enable businesses to gain insights into inventory performance, sales trends, and product demand. This data-driven approach aids in making informed decisions related to stock replenishment, supplier selection, and pricing strategies.

The system’s modular design ensures scalability and adaptability, allowing businesses of all sizes—whether small enterprises or large corporations—to customize and scale the system according to their specific needs. Modules such as order management, supplier management, and user authentication further enhance the system's functionality, making it a comprehensive solution for inventory control. Integration with databases (using JDBC) ensures that the system can handle large amounts of data

In conclusion, the Stock Management System not only meets the fundamental requirements of modern inventory management but also provides businesses with the tools needed to optimize their operations. It fosters better resource allocation, reduces operational costs, and enhances inventory control. By leveraging technology, this system supports businesses in maintaining accurate inventory levels, improving customer satisfaction, and ultimately driving growth and profitability. The successful implementation of this system represents a significant advancement in inventory management practices, helping businesses remain competitive in an increasingly fast-paced market environment.

4.2 FUTURE SCOPE

The future scope of the Stock Management System involves several enhancements to increase its functionality, scalability, and adaptability to emerging business needs. Integrating the system with e-commerce platforms would allow businesses to manage inventory across multiple sales channels in real-time, preventing stockouts or overselling. Incorporating advanced analytics and AI could provide predictive insights into demand forecasting, enabling better inventory planning and decision-making. Additionally, developing a mobile app for on-the-go inventory management and adopting cloud-based deployment would improve accessibility, flexibility, and scalability. The system could also be integrated with other business tools like ERP, CRM, and accounting systems to streamline operations. Further improvements such as multi-currency and multi-language support, RFID integration for real-time tracking, and blockchain for enhanced transparency could enhance global operations and security. Other innovations like automated stock replenishment, supplier performance monitoring, and user interface enhancements would contribute to greater efficiency and a better user experience. Ultimately, these advancements would make the Stock Management System more robust, efficient, and aligned with modern business practices, helping businesses optimize inventory management, reduce costs, and increase profitability.

CHAPTER 5 APPENDIX A

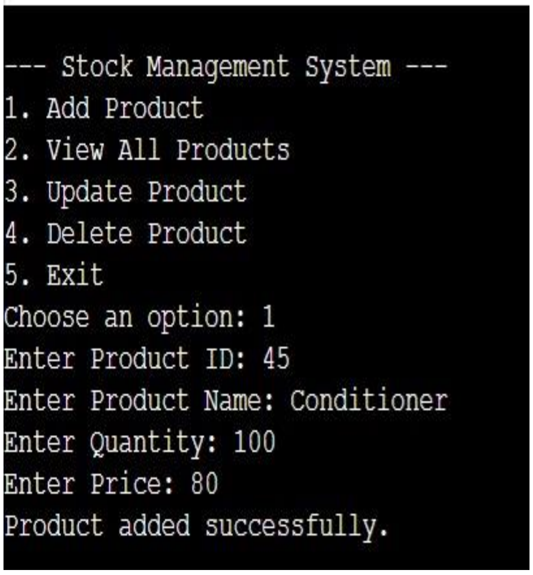
(SOURCE CODE)

```
import java.util.ArrayList;
import java.util.Scanner;

class Product {
    private String id;
    private String name;
    private int quantity;
    private double price;

    public Product(String id, String name, int quantity, double price) {
        this.id = id;
        this.name = name;
        this.quantity = quantity;
        this.price = price;
    }

    public String getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public int getQuantity() {
        return quantity;
    }

    public double getPrice() {
        return price;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Quantity: " + quantity + ", Price: $" + price;
    }
}

public class StockManagementSystem {
    private static ArrayList<Product> stock = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        while (true) {
            System.out.println("\n--- Stock Management System ---");
            System.out.println("1. Add Product");
            System.out.println("2. View All Products");
            System.out.println("3. Update Product");
            System.out.println("4. Delete Product");
            System.out.println("5. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    addProduct();
                    break;
                case 2:
                    viewProducts();
                    break;
                case 3:
                    updateProduct();
                    break;
                case 4:
                    deleteProduct();
                    break;
                case 5:
                    System.out.println("Exiting... Goodbye!");
                    return;
                default:
                    System.out.println("Invalid option. Try again.");
            }
        }
    }

    private static void addProduct() {
        System.out.print("Enter Product ID: ");
        String id = scanner.nextLine();
        System.out.print("Enter Product Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Quantity: ");
        int quantity = scanner.nextInt();
        System.out.print("Enter Price: ");
        double price = scanner.nextDouble();
        scanner.nextLine(); // consume newline

        stock.add(new Product(id, name, quantity, price));
        System.out.println("Product added successfully.");
    }

    private static void viewProducts() {
        if (stock.isEmpty()) {
            System.out.println("No products in stock.");
        } else {
            for (Product product : stock) {
                System.out.println(product);
            }
        }
    }

    private static void updateProduct() {
        System.out.print("Enter Product ID to update: ");
        String id = scanner.nextLine();
        for (Product product : stock) {
            if (product.getId().equals(id)) {
                System.out.print("Enter new quantity: ");
                int quantity = scanner.nextInt();
                System.out.print("Enter new price: ");
                double price = scanner.nextDouble();
                scanner.nextLine(); // consume newline

                product.setQuantity(quantity);
                product.setPrice(price);
                System.out.println("Product updated successfully.");
                return;
            }
        }
        System.out.println("Product not found.");
    }

    private static void deleteProduct() {
        System.out.print("Enter Product ID to delete: ");
        String id = scanner.nextLine();
        for (int i = 0; i < stock.size(); i++) {
            if (stock.get(i).getId().equals(id)) {
                stock.remove(i);
                System.out.println("Product deleted successfully.");
```

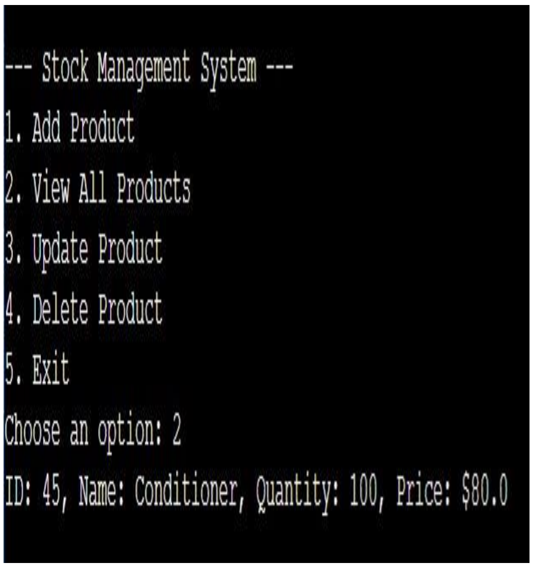
```
                return;
            }
        }
        System.out.println("Product not found.");
    }
}
```

APPENDIX B

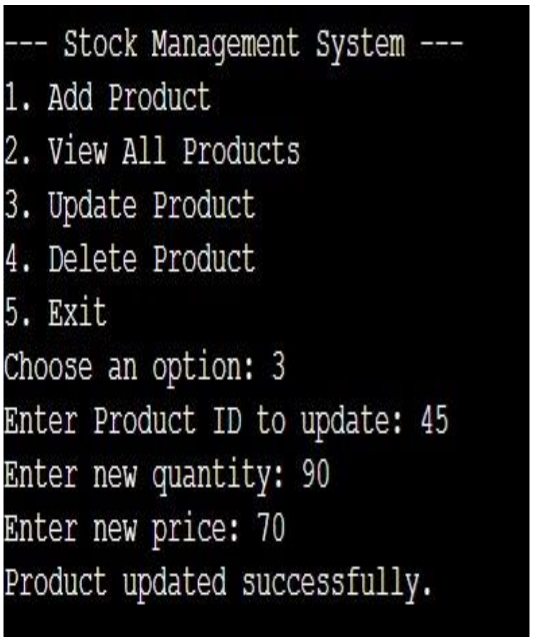
(SCREENSHOTS)



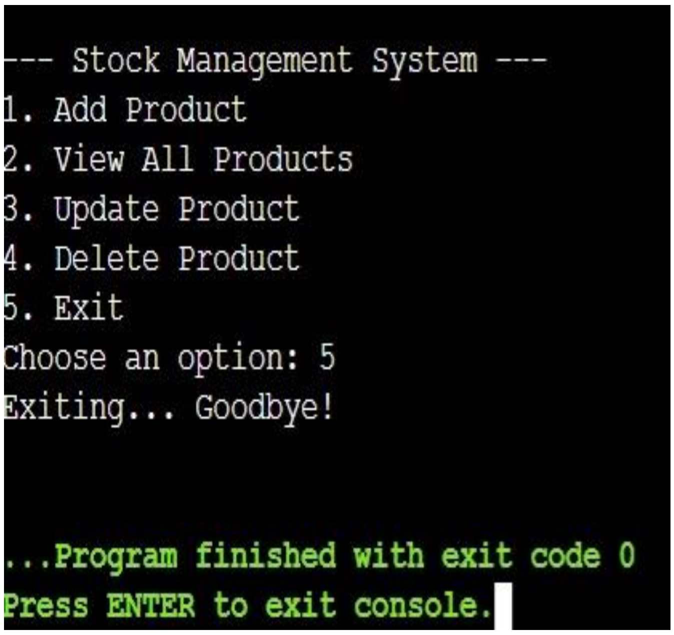
```
--- Stock Management System ---
1. Add Product
2. View All Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 1
Enter Product ID: 45
Enter Product Name: Conditioner
Enter Quantity: 100
Enter Price: 80
Product added successfully.
```



```
--- Stock Management System ---
1. Add Product
2. View All Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 2
ID: 45, Name: Conditioner, Quantity: 100, Price: $80.0
```



```
--- Stock Management System ---
1. Add Product
2. View All Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 3
Enter Product ID to update: 45
Enter new quantity: 90
Enter new price: 70
Product updated successfully.
```



```
--- Stock Management System ---
1. Add Product
2. View All Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 5
Exiting... Goodbye!


...Program finished with exit code 0
Press ENTER to exit console.
```

REFERENCES

Here are some references that could be useful when working on a Stock Management System with Java AWT or similar inventory management systems:

Websites:

1. **Oracle Java Documentation**
   Java AWT Documentation
   Official documentation for AWT, providing in-depth information on AWT classes and components.
2. **GeeksforGeeks: AWT in Java**
   AWT in Java – GeeksforGeeks
   This article explains the basics of AWT, components, layout managers, and event handling in Java.
3. **Java2s.com - Java AWT Examples**
   AWT Examples
   A collection of examples demonstrating various uses of AWT components and layout management in Java, perfect for building your own Stock Management System.

1. **Java Programming Tutorials** – Java Programming Tutorials on YouTube
   These tutorials cover Java programming fundamentals, including GUI development with AWT and Swing, which can help you understand how to build a Stock Management System interface.
2. **TutorialsPoint - Java AWT Tutorial**
   Java AWT Tutorial - TutorialsPoint
   A beginner-friendly tutorial for Java AWT, including detailed explanations and code examples.