STOCK MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

NIRANJAN KUMAR M (2303811710421107)

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 **NIRANJAN KUMAR M(2303811710421107)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.,),
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 03/12/2024

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 **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING.**

.

Signature

NIRANJAN KUMAR M

Place: Samayapuram

Date: 03/12/2024

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 **Mrs. K. VALLI PRIYADHARSHINI, M.E., (Ph.D.,),** 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 is a comprehensive software solution designed to streamline and optimize inventory management for businesses. Developed in Java, the system is structured around core Object-Oriented Programming (OOP) principles, utilizing encapsulation to secure product data and provide a modular, maintainable codebase. The system provides functionalities for adding new products, updating stock quantities, placing orders, and displaying the current inventory status. Users interact with the system through a Graphical User Interface (GUI) built using Java's Abstract Window Toolkit (AWT), ensuring ease of use for all stakeholders involved in inventory management.The product data is encapsulated in a Product class, where each product is defined by attributes such as ID, name, quantity, and price. Accessor and mutator methods (getters and setters) are provided for each attribute, maintaining data security and preventing direct manipulation. Through synchronized methods, the system ensures that actions such as adding products, updating stock, or placing orders are safely executed, preventing race conditions and ensuring data consistency in a multi-threaded environment.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
The Stock Management System is a Java-based desktop application that enables efficient product inventory management. It integrates features like adding products, updating stock, placing orders, and displaying inventory.Using multithreading, it ensures automatic periodic inventory saving to prevent data loss.	**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 NO.	TITLE	PAGE NO.
	ABSTRACT	VIII
1	**INTRODUCTION**	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	**PROJECT METHODOLOGY**	3
	2.1 Proposed Work	3
	2.2 Block Diagram	3
3	**MODULE DESCRIPTION**	4
	3.1 Product Module	4
	3.2 File Handling Module	4
	3.3 Auto-Save Module	4
	3.4 Stock Management Module	4
	3.5 GUI Module	5
4	**CONCLUSION & FUTURE SCOPE**	6
	4.1 Conclusion	6
	4.2 Future Scope	6
	REFERENCES	15
	APPENDIX A (SOURCE CODE)	7
	APPENDIX B (SCREENSHOTS)	13

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the Stock Management System is to provide an efficient, user-friendly solution for managing inventory in businesses. It aims to simplify tasks such as adding products, updating stock levels, placing orders, and displaying inventory status through a graphical user interface (GUI). The system ensures data security and consistency by utilizing object-oriented principles like encapsulation and synchronized methods for safe multi-user access. Additionally, it integrates file handling for persistent storage of product data and implements an auto-save feature to prevent data loss. The system is designed to be scalable and easily extendable, offering a reliable tool for businesses to streamline their inventory management processes.

1.2 Overview

The Stock Management System is a comprehensive application that facilitates efficient management of product inventory for businesses. It allows users to perform operations like adding new products, updating stock quantities, placing orders, and displaying inventory through a simple graphical user interface (GUI). The system is designed with object-oriented principles, ensuring encapsulation and data security. It leverages file handling to save and load product data, allowing persistent storage between sessions. Additionally, an auto-save feature is implemented using multithreading to automatically save the inventory at regular intervals. This system provides a robust solution for inventory control with scalable and synchronized methods for managing concurrent user interactions.

1.3 Java Programming Concepts

- Encapsulation: The Product class uses private fields and public getter/setter methods to control access to product details like id, name, quantity, and price, ensuring data security and integrity.
- Inheritance: The StockManagementSystem class extends Frame, inheriting methods and properties to create a GUI window. This enables GUI components like buttons and event handling in the application.

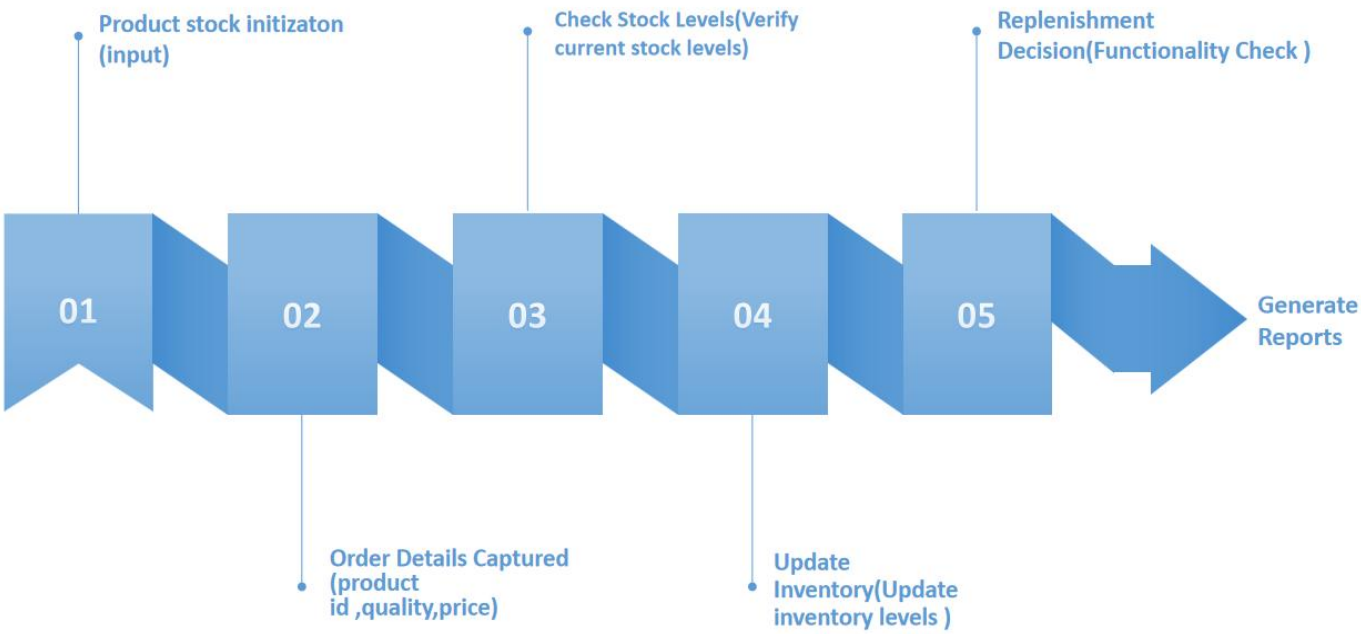
- AWT (Abstract Window Toolkit): AWT components such as Button and FlowLayout are used for creating and organizing the GUI, providing interaction with users.
- Event Handling: The code implements ActionListener for handling button clicks (e.g., adding products, placing orders) and WindowAdapter to handle window closing events (e.g., saving data on exit).
- File Handling: ObjectInputStream and ObjectOutputStream are used to save and load the product inventory to/from a file, ensuring data persistence between sessions.
- Swing: JOptionPane is used for showing input dialogs to collect data from the user in a simple, graphical manner.
- Exception Handling: Exceptions like IOException, ClassNotFoundException, and NumberFormatException are caught to handle file errors, class loading issues, and invalid input gracefully.
- Polymorphism: The toString() method is overridden in the Product class to provide a custom string representation of product objects.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The StockManagementSystem is a Java application that manages a product inventory using features like AWT-based GUI, multithreading, file handling, and encapsulation. The Product class encapsulates product details such as ID, name, quantity, and price, with getters, setters, and serialization support. The main class, StockManagementSystem, allows adding products, updating stock, placing orders, and displaying inventory, with synchronized methods for thread safety. Inventory is persistently saved and loaded using object serialization, and an auto-save feature runs every 30 seconds via a daemon thread. The AWT GUI includes buttons for operations and prompts user inputs through dialogs, ensuring usability. The program saves inventory on exit and supports a maximum number of products, defined at runtime.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Product Module:

Represents each product in the inventory with attributes: id, name, quantity, and price.Implements encapsulation by keeping fields private and providing public getters and setters for controlled access.Includes the Serializable interface to enable saving objects to a file.The toString method provides a readable string representation of product details for display.

3.2 File Handling Module:

Save Inventory: Uses ObjectOutputStream to serialize the inventory array and save it to a file (inventory.dat). This ensures data persistence between application runs.Load Inventory: Uses ObjectInputStream to deserialize the file and restore the inventory into memory. Handles missing files or format issues with appropriate fallback (starting a fresh inventory).

3.3 Auto-Save Module:

Runs a daemon thread in the background to periodically save the inventory every 30 seconds.Ensures changes are saved automatically, reducing the risk of data loss during unexpected exits.Uses Thread.sleep for scheduling, with interruption handling to ensure smooth operation.

3.4 Stock Management Module:

Add Product: Adds a new product to the inventory if space is available, updating the inventory array and count.Update Stock: Updates the quantity of an existing product by matching its ID. Displays a message if the product is not found.Place Order: Checks stock availability and deducts the ordered quantity if sufficient stock exists. Otherwise, informs the user of insufficient stock or invalid product ID.Display Inventory: Prints the details of all products in the inventory or a message if the inventory is empty.

3.5 GUI Module:

Built using AWT (Abstract Window Toolkit) for a graphical interface.Provides buttons for actions: "Add Product", "Update Stock", "Place Order", "Display Inventory", and "Exit".Uses JOptionPane dialogs for user input and feedback, ensuring an interactive experience.Listens to button clicks with ActionListener to execute corresponding operations.Handles window-closing events to save inventory data and exit gracefully.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The StockManagementSystem is a well-designed application that effectively integrates Java's object-oriented principles, file handling, multithreading, and GUI capabilities to provide a robust and user-friendly inventory management solution. By encapsulating product details in the Product class and ensuring data persistence with serialization, the program guarantees both data security and continuity across sessions. Features like synchronized methods and a daemon thread for auto-saving demonstrate thoughtful implementation to handle concurrency and prevent data loss. The ability to seamlessly add, update, and manage stock, coupled with validation for operations like placing orders, ensures that the application meets the essential requirements of a practical inventory system.Furthermore, the AWT-based GUI makes the system accessible to users with minimal technical expertise. Buttons for core functionalities and input dialogs simplify user interaction, while clean integration between the interface and the underlying logic ensures a seamless experience.

4.2 FUTURE SCOPE

The **future scope** of the above Stock Management System includes enhancements to improve usability, scalability, and integration. First, the system can be expanded to support database integration instead of relying on local file storage, enabling efficient inventory management for large-scale businesses. Incorporating a relational database like MySQL or PostgreSQL would facilitate querying, data security, and remote access. Additionally, the AWT-based user interface could be upgraded to a modern framework like JavaFX or Swing for a more responsive, user-friendly experience with advanced UI features such as charts and dashboards.Another potential development is the integration of the system with web or mobile applications to allow real-time monitoring and management of inventory. Features such as generating reports, notifications for low stock, and automated ordering can be added to make the system more intelligent.

APPENDIX A
(SOURCE CODE)

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.JOptionPane;
import java.util.Scanner;
class Product implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    private String name;
    private int quantity;
    private double price;
    public Product(int id, String name, int quantity, double price) {
        this.id = id;
        this.name = name;
        this.quantity = quantity;
        this.price = price;
    }
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
```

```
    public double getPrice() {
        return price;
    }
    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Quantity: " + quantity + ", Price: $" + price;
    }
}
// Main Stock Management System
public class StockManagementSystem extends Frame {
    private Product[] inventory;
    private int productCount = 0;
    private final int maxProducts;
    private static final String FILE_NAME = "inventory.dat";
    public StockManagementSystem(int maxProducts) {
        this.maxProducts = maxProducts;
        inventory = new Product[maxProducts];
        loadInventory();
        setupGUI();
        startAutoSave();
    }
    // Add Product
    public synchronized void addProduct(int id, String name, int quantity, double price) {
        if (productCount >= inventory.length) {
            System.out.println("Inventory is full. Cannot add more products.");
            return;
        }
        inventory[productCount++] = new Product(id, name, quantity, price);
        System.out.println("Product added successfully!");
    }
    // Update Stock
    public synchronized void updateStock(int productId, int newQuantity) {
        for (int i = 0; i < productCount; i++) {
            if (inventory[i].getId() == productId) {
```

```
            inventory[i].setQuantity(newQuantity);
            System.out.println("Stock updated successfully for: " + inventory[i].getName());
            return;
        }
    }
    System.out.println("Product not found.");
}
// Place Order
public synchronized void placeOrder(int productId, int orderQuantity) {
    for (int i = 0; i < productCount; i++) {
        if (inventory[i].getId() == productId) {
            if (inventory[i].getQuantity() >= orderQuantity) {
                inventory[i].setQuantity(inventory[i].getQuantity() - orderQuantity);
                System.out.println("Order placed successfully for: " + inventory[i].getName());
            } else {
                System.out.println("Insufficient stock for: " + inventory[i].getName());
            }
            return;
        }
    }
    System.out.println("Product not found.");
}
public void displayInventory() {
    if (productCount == 0) {
        System.out.println("Inventory is empty.");
        return;
    }
    System.out.println("Inventory Status:");
    for (int i = 0; i < productCount; i++) {
        System.out.println(inventory[i]);
    }
}
// File Handling: Save Inventory
private synchronized void saveInventory() {
```

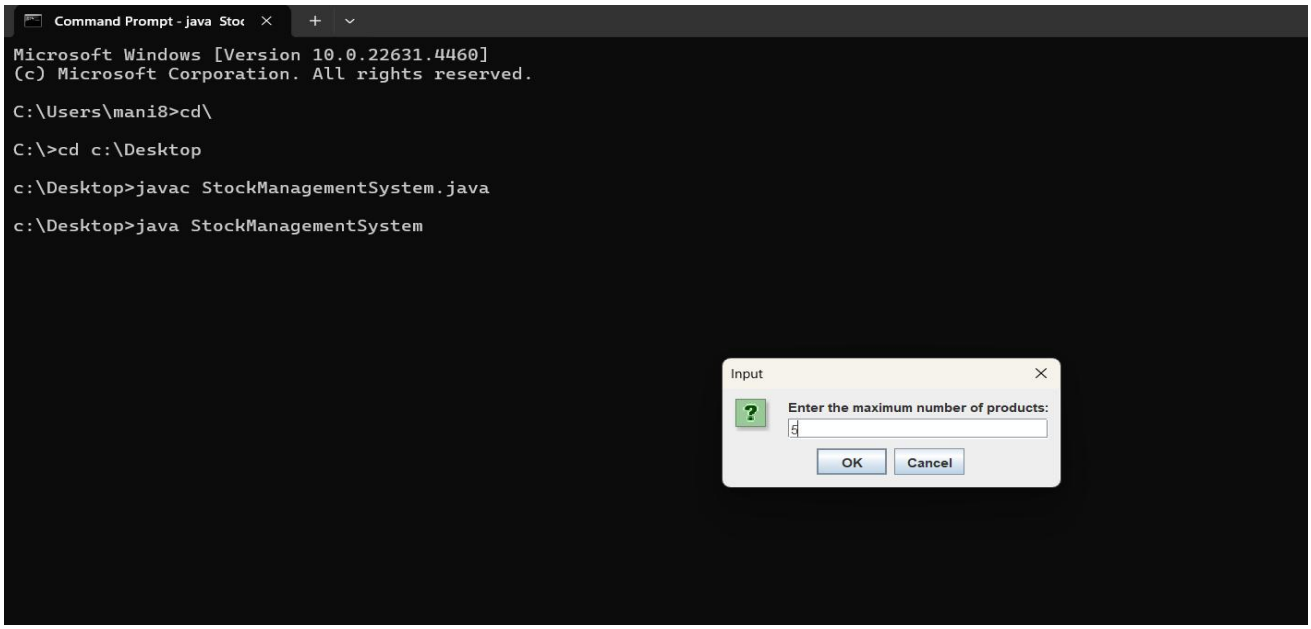
```
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            oos.writeObject(inventory);
            oos.writeInt(productCount);
            System.out.println("Inventory saved to file.");
        } catch (IOException e) {
            System.out.println("Error saving inventory: " + e.getMessage());
        }
    }
    // File Handling: Load Inventory
    private void loadInventory() {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
            inventory = (Product[]) ois.readObject();
            productCount = ois.readInt();
            System.out.println("Inventory loaded from file.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("No previous inventory found. Starting fresh.");
        }
    }
    private void startAutoSave() {
        Thread autoSaveThread = new Thread(() -> {
            while (true) {
                try {
                    Thread.sleep(30000); // Save every 30 seconds
                    saveInventory();
                } catch (InterruptedException e) {
                    System.out.println("Auto-save interrupted.");
                }
            }
        });
        autoSaveThread.setDaemon(true);
        autoSaveThread.start();
    }
    private void setupGUI() {
```

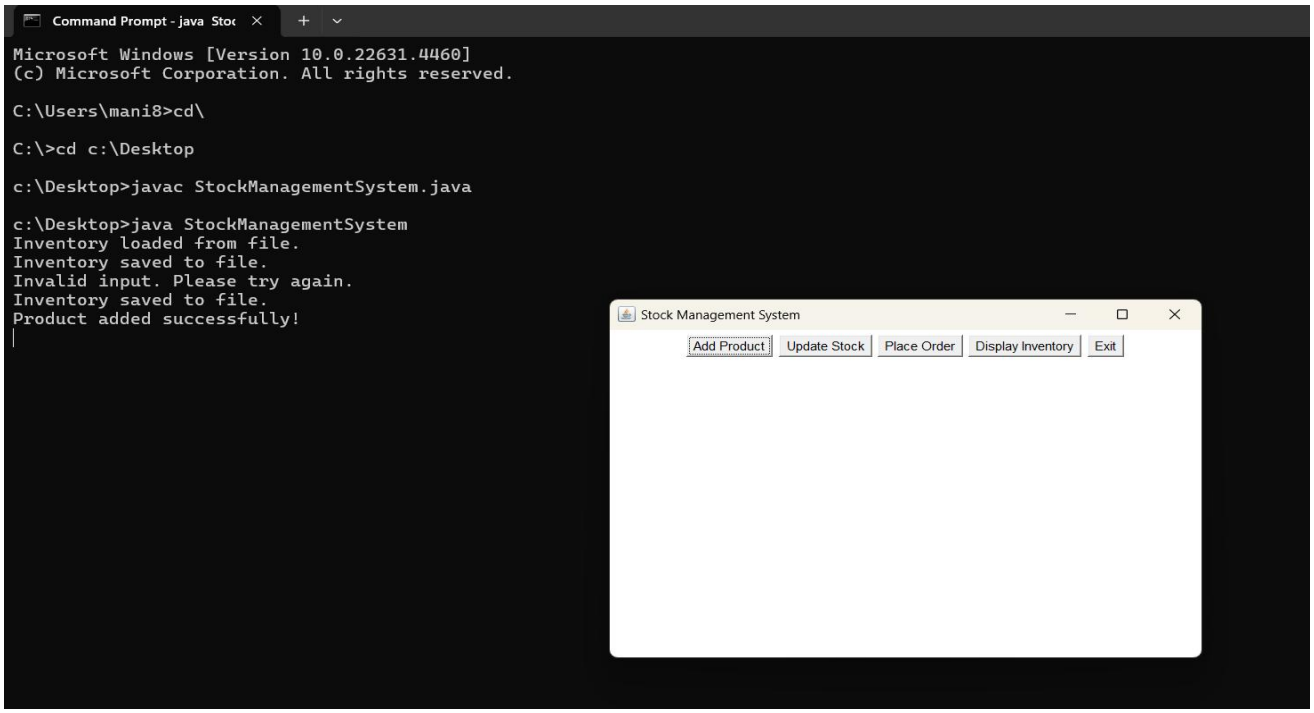
```
setTitle("Stock Management System");
setSize(400, 300);
setLayout(new FlowLayout());
Button addButton = new Button("Add Product");
Button updateButton = new Button("Update Stock");
Button orderButton = new Button("Place Order");
Button displayButton = new Button("Display Inventory");
Button exitButton = new Button("Exit");
add(addButton);
add(updateButton);
add(orderButton);
add(displayButton);
add(exitButton);
addButton.addActionListener(e -> {
   try {
      int id = Integer.parseInt(prompt("Enter Product ID:"));
      String name = prompt("Enter Product Name:");
      int quantity = Integer.parseInt(prompt("Enter Product Quantity:"));
      double price = Double.parseDouble(prompt("Enter Product Price:"));
      addProduct(id, name, quantity, price);
   } catch (NumberFormatException ex) {
      System.out.println("Invalid input. Please try again.");
   }
});
updateButton.addActionListener(e -> {
   try {
      int id = Integer.parseInt(prompt("Enter Product ID to update:"));
      int newQuantity = Integer.parseInt(prompt("Enter new stock quantity:"));
      updateStock(id, newQuantity);
   } catch (NumberFormatException ex) {
      System.out.println("Invalid input. Please try again.");
   }
});
orderButton.addActionListener(e -> {
```

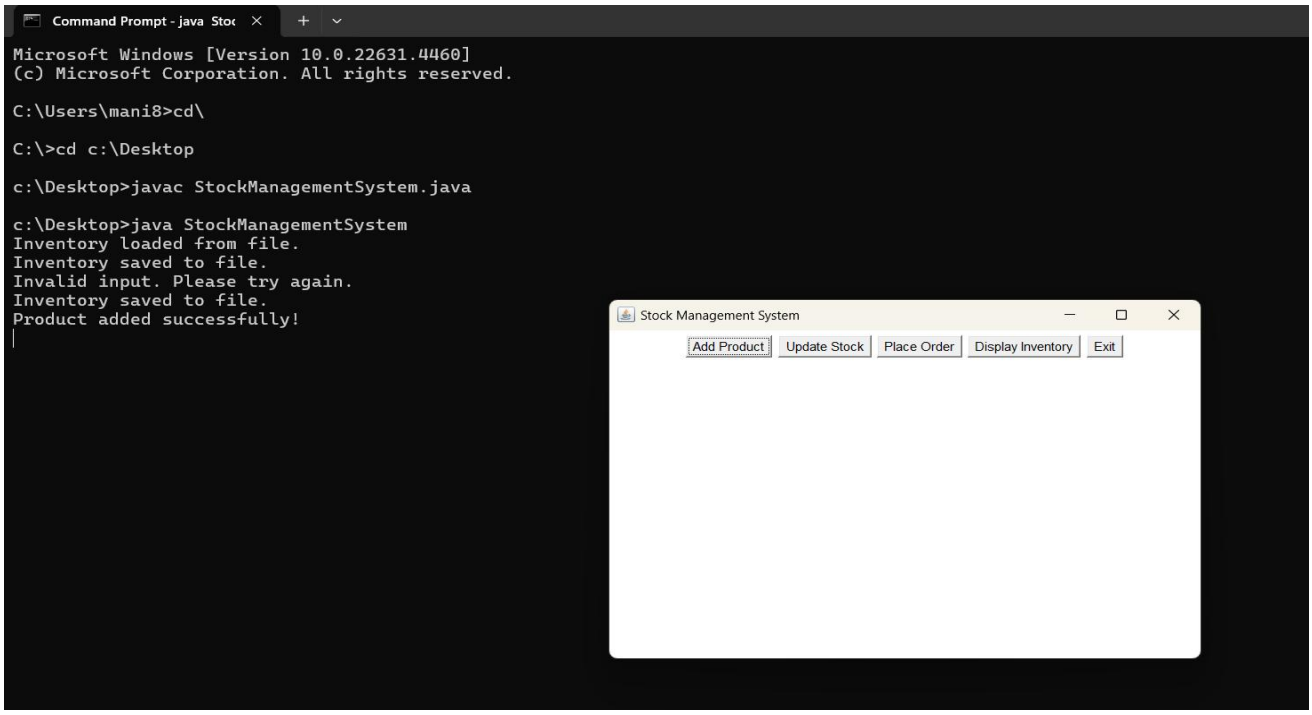
```
            try {
                int id = Integer.parseInt(prompt("Enter Product ID to order:"));
                int quantity = Integer.parseInt(prompt("Enter order quantity:"));
                placeOrder(id, quantity);
            } catch (NumberFormatException ex) {
                System.out.println("Invalid input. Please try again.");
            }
        });
        displayButton.addActionListener(e -> displayInventory());
        exitButton.addActionListener(e -> {
            saveInventory();
            System.out.println("Exiting the system. Goodbye!");
            System.exit(0);
        });
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                saveInventory();
                System.exit(0);
            }
        });
        setVisible(true);
    }
    private String prompt(String message) {
        return JOptionPane.showInputDialog(this, message);
    }
    public static void main(String[] args) {
        int maxProducts = Integer.parseInt(JOptionPane.showInputDialog("Enter the maximum
number of products:"));
        new StockManagementSystem(maxProducts);
    }
}
```

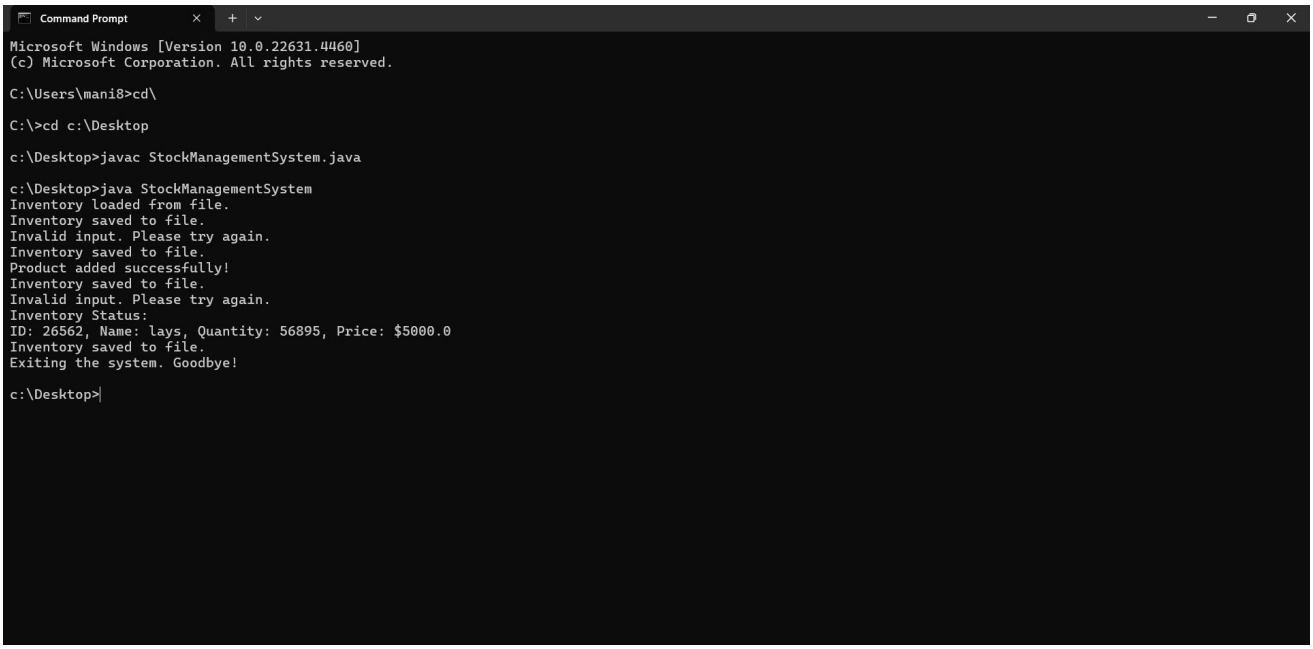
APPENDIX B

(SCREENSHOTS)






Command Prompt - java Stoc
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.
C:\Users\mani8>cd\
C:\>cd c:\Desktop
c:\Desktop>javac StockManagementSystem.java
c:\Desktop>java StockManagementSystem
Inventory loaded from file.
Inventory saved to file.
Invalid input. Please try again.
Inventory saved to file.
Product added successfully!
Stock Management System
Add Product
Update Stock
Place Order
Display Inventory
Exit


Command Prompt
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.
C:\Users\mani8>cd\
C:\>cd c:\Desktop
c:\Desktop>javac StockManagementSystem.java
c:\Desktop>java StockManagementSystem
Inventory loaded from file.
Inventory saved to file.
Invalid input. Please try again.
Inventory saved to file.
Product added successfully!
Inventory saved to file.
Invalid input. Please try again.
Inventory Status:
ID: 26562, Name: lays, Quantity: 56895, Price: $5000.0
Inventory saved to file.
Exiting the system. Goodbye!
c:\Desktop>

REFERENCES

- **Books**:

  Head First Java by Kathy Sierra and Bert Bates.

  Java: The Complete Reference by Herbert Schildt.

- **Websites**:

  Oracle Java Documentation - Tutorials on AWT, serialization, and multithreading.

  GeeksforGeeks - Examples of file handling and synchronization.

  Baeldung - Guides on Java serialization and file handling.

- **YouTube**:

  Programming with Mosh - Java tutorials and advanced concepts.

  Telusko - AWT, threading, and file handling explanations.

  CodeWithHarry - Beginner-friendly Java tutorials.