

INVENTORY MANAGEMENT SYSTEM



A PROJECT REPORT

Submitted by

NANTHINI S (2303811710422101)

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 “ **INVENTORY MANAGEMENT SYSTEM**” is the bonafide work of **NANTHINI S (2303811710422101)** 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

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
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.),
SUPERVISOR
ASSISTANT PROFESSOR

SIGNATURE

Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.),

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 03.12.2024

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

INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Mrs.K.PRIYARUNA PRIYA, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8104-DSEC, PERAMBALUR.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**INVENTORY 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



NANTHINI S

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 Inventory Management System is a dynamic and modular software designed to streamline the complex processes of inventory management, order processing, customer interaction, and reporting. The inventory module efficiently manages stock levels, enabling users to add, update, and monitor product availability. Customers are managed through a module that tracks their information and maintains a detailed history of their orders. The order management module enables seamless order creation and processing. Payments are integrated into the system, allowing users to generate invoices and track transactions. The delivery management module facilitates the tracking of orders from shipment to delivery, offering real-time status updates to enhance operational efficiency and customer satisfaction. The reporting and analytics module generates detailed insights, including inventory reports and sales. Designed with scalability and user-friendliness in mind, it serves as an essential tool for businesses of all sizes aiming to optimize their processes, improve resource utilization, and deliver exceptional service to customers.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
<p>An Inventory Management System is a comprehensive solution for managing stock, streamlining order processing, and optimizing inventory control. It consists of modules for user, product, stock tracking, order processing, supplier management, reporting, and billing. These modules work together to ensure real-time updates, minimize errors, and maintain stock availability. This system automates processes like invoice generation and payment tracking. Scalable and efficient, IMS is essential for businesses to enhance operational efficiency and stay competitive.</p>	<p>PO1 -3 PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3</p>	<p>PSO1 -3 PSO2 -3 PSO3 -3</p>

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

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
No.		No.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	PROJECT METHODOLOGY	4
	2.1 Proposed Work	4
	2.2 Block Diagram	4
3	MODULE DESCRIPTION	5
	3.1 User Interface	5
	3.2 Inventory Management module	5
	3.3 Order Management module	5
	3.4 Customer Management module	6
	3.5 Payment module	6
	3.6 Delivery Management module	6
	3.7 Report and Analysis module	7
4	CONCLUSION & FUTURE SCOPE	8
	4.1 Conclusion	8
	4.2 Future Scope	8
	APPENDIX A (SOURCE CODE)	10
	APPENDIX B (SCREENSHOTS)	22

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the Inventory Management System is to efficiently manage inventory, orders, and customer interactions. It ensures role-based access for secure operations and automates tasks like stock tracking and order processing. The system integrates payment tracking and generates reports for better decision-making. It reduces manual errors, enhances operational efficiency, and provides a user-friendly interface. Its ultimate goal is to optimize resource utilization and improve customer satisfaction.

1.2 Overview

The Inventory Management System is a comprehensive software solution designed to manage various business operations such as inventory tracking, order processing, and customer management. The system is organized into modules that handle distinct business functions: user management, inventory control, order processing, customer management, supplier coordination, delivery tracking, and reporting. Through the user module, the system ensures secure access and role-based functionality for different types of users, such as admins and staff. The inventory module allows for real-time stock updates, tracking product quantities, and managing product data efficiently. Orders are processed seamlessly with stock verification and order creation, while customer details and their order history are stored for easy reference. The system also integrates supplier management to track restocking needs and delivery management to monitor the status of shipments. Additionally, the reporting module generates valuable insights, helping users make informed decisions related to stock levels, sales, and reorder requirements. Payment details are tracked, ensuring a smooth transition from

order to payment. The system increases efficiency, reduces operational errors, and enables effective business decision-making.

1.3 Java Programming Concepts

1. Object-Oriented Programming (OOP) Concepts:

- **Encapsulation:** The inventory data (products, quantities, and prices) is stored in private collections (HashMap) and managed through public methods.
- **Abstraction:** The system provides specific methods like addStock to interact with the inventory without exposing internal details.

2. AWT (Abstract Window Toolkit):

- **GUI Components:** Used Frame, TextField, Label, Button, and TextArea to create a graphical user interface.
- **Event Handling:**
 - **ActionListener:** Used to perform actions when buttons like "Login," "Add Stock," or "View Inventory" are clicked.
- **Layout Management:** Utilized FlowLayout for arranging GUI components in a simple, left-to-right flow.

3. Collections Framework:

- **HashMap:** Used to store inventory data (product names as keys, quantities, and prices as values). It helps manage the dynamic addition and retrieval of stock information.

4. Control Flow:

- **Conditional Statements:** if and else are used to validate login credentials and manage actions like enabling or disabling inventory controls.
- **Loops:** While not directly used here, loops could be added for extended functionality (e.g., menu-driven interfaces).

5. Data Handling and Parsing:

- **Parsing Input:** Converted text input (TextField.getText()) into numeric types

like int and double using Integer.parseInt and Double.parseDouble.

6. Access Modifiers:

- private: Used to restrict access to stock and prices HashMaps within the inventory system.
- public: Used to allow controlled access to methods for interacting with the inventory.

7. Error Handling Concepts:

- Basic Validation: Checked for valid login credentials and ensured data like quantity and price are entered before processing.

8. Static Methods and Variables:

- Static Main Method: public static void main(String[] args) serves as the entry point for the program.
- Static Fields: GUI components (TextField, Button, etc.) and inventory-related collections (stock, prices) are declared static for shared access.

9. String Manipulation:

- Used String operations such as concatenation (+) to display messages and construct inventory details.

10. Event-Driven Programming:

- GUI functionality is inherently event-driven, responding to user actions like button clicks (ActionEvent).

11. Modularity:

- Code is organized into methods like enableStockActions, which isolates functionality for reusability and readability.

12. Basic Input and Output:

- Captured user input through TextField and displayed results using TextArea.

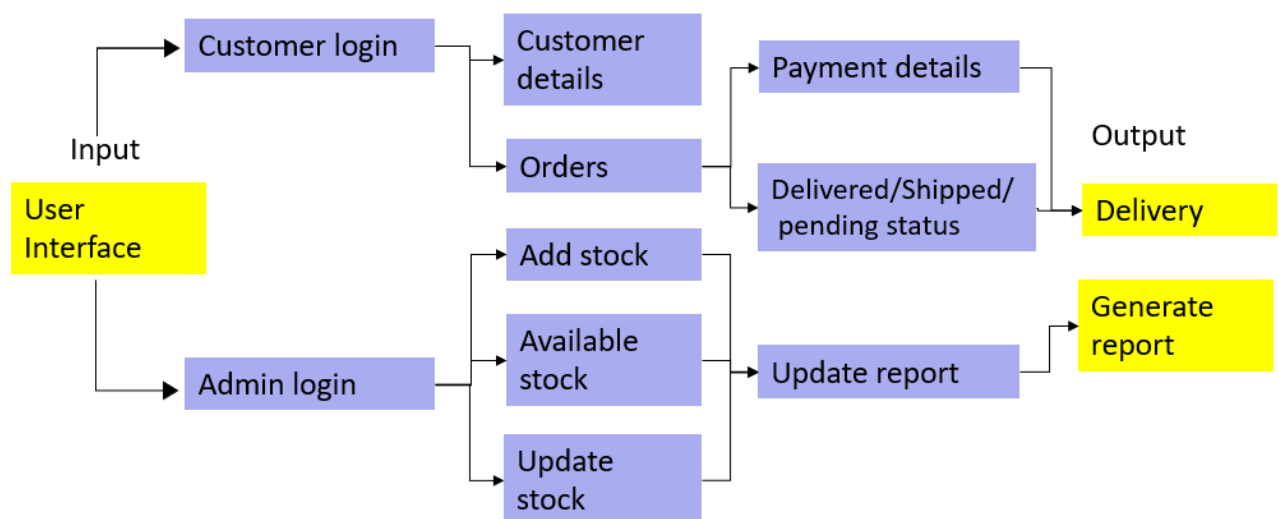
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the Simple Inventory Management System using AWT focuses on developing an intuitive graphical application for managing inventory, orders, and customer interactions. It integrates user authentication for secure access, enabling administrators to handle stock updates and customers to place orders seamlessly. The system facilitates efficient inventory management by tracking product quantities and prices while automatically adjusting stock levels upon order processing. It incorporates features for simulating payment confirmations, maintaining customer profiles, and storing order histories for better organization. Additionally, the system supports generating inventory and sales reports to provide valuable insights into business operations. Built using Java's AWT, it offers a clean, interactive interface with event-driven functionality, catering to small-scale businesses by streamlining operations, minimizing errors, and enhancing overall productivity.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 User Interface

The User Module is responsible for managing user authentication and access control within the system. It ensures that only authorized personnel can interact with the application by validating usernames and passwords. The module supports role-based access, distinguishing between administrators, staff, and customers, which allows for different levels of permissions. For instance, administrators can manage inventory and view reports, while customers can place orders and view their order history. This module acts as the gateway to the system, ensuring secure and controlled access.

3.2. Inventory Management Module

This module oversees the core inventory operations, such as adding new products, updating stock quantities, and setting prices for each item. It allows users to view detailed information about current stock levels and pricing. By providing an up-to-date inventory, it ensures that the system reflects real-world product availability, helping to avoid issues like overstocking or stock shortages. This module is integral to maintaining an efficient supply chain and supports seamless integration with other modules like order and reporting.

3.3. Order Management Module

The Order Management Module streamlines the process of creating and managing customer orders. It enables users to input order details, validate stock availability, and calculate the total price of items based on quantity and pricing. Once an order is confirmed, the module updates the inventory to reflect the reduced stock. This integration ensures consistency and accuracy in inventory data while delivering a smooth ordering experience for customers. The module is critical for maintaining order accuracy and enhancing customer satisfaction.

3.4. Customer Management Module

This module focuses on handling customer-related data and interactions. It stores customer information, such as names and contact details, and maintains a history of all their past orders. By organizing customer data, it enables personalized service and streamlined order tracking. This module enhances customer relationships by making it easier to address their needs and inquiries efficiently. It also supports better communication and order transparency, contributing to improved customer loyalty.

3.5. Payment Module

The Payment Module is designed to handle the financial aspects of order processing. It calculates the total cost of an order based on the selected products and quantities and provides a confirmation of the payment. Although it doesn't include real-world payment gateway integration, it simulates the process by displaying payment success messages. This module is crucial for ensuring that transactions are acknowledged, adding a layer of professionalism to the system while recording payment details for reference.

3.6. Delivery Management Module

This module manages the delivery process of customer orders. It allows administrators or staff to update the status of deliveries, marking them as "Pending," "Shipped," or "Delivered." By tracking delivery statuses, it provides transparency and keeps customers informed about the progress of their orders. This module is essential for maintaining customer trust and ensuring a smooth delivery workflow. It also serves as a communication bridge between the business and its customers regarding the fulfillment of orders.

3.7. Report and Analytics Module

The Report and Analytics Module generates comprehensive reports on inventory levels and sales activities. It produces inventory reports that detail stock levels and pricing, helping administrators make informed decisions about restocking. Additionally, it provides sales reports summarizing order histories and customer interactions, offering insights into business performance. By analyzing these reports, businesses can identify trends, improve operational efficiency, and make data-driven decisions to enhance overall performance.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The **Inventory Management System using Java and AWT** is an integrated solution designed to enhance business operations through its seven core modules. The **User Module** handles secure login and role-based access, ensuring that only authorized personnel can perform certain actions. The **Inventory Management Module** tracks product stock levels and pricing, making it easy to monitor inventory and adjust as necessary. The **Order Module** processes customer orders and checks stock availability, while the **Payment Module** manages transaction processing, ensuring smooth financial operations. The **Customer Module** stores customer information and their order history, improving customer relationship management. The **Delivery Module** updates and tracks the delivery status of orders, providing timely and accurate delivery information. Finally, the **Reporting Module** generates detailed reports on inventory and sales data, providing key insights for decision-making. This system combines efficiency, security, and scalability, making it suitable for businesses aiming to optimize their operations and customer service.

4.2 FUTURE SCOPE

The Inventory Management System has a promising future scope with several enhancements. Advanced user roles can be implemented for better access control and security. Integration with cloud platforms can enable real-time inventory updates across multiple locations, improving scalability and global accessibility. E-commerce platform integration would streamline online and offline inventory synchronization. AI-based analytics and reporting tools can be added for sales forecasting and trend analysis. A mobile app with features like barcode scanning would provide on-the-go

management. IoT and RFID integration could enable real-time stock tracking, while machine learning can enhance fraud detection and customer behavior insights. Payment gateway integration would streamline transactions, and multi-language or multi-currency support can expand market reach. Blockchain technology can be used for secure transaction logs, making the system robust and future-proof.

APPENDIX A

(SOURCE CODE)

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.util.List;
// User Module
class User {
    private String username;
    private String password;
    private String role;

    public User(String username, String password, String role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }

    public boolean login(String inputUsername, String inputPassword) {
        return inputUsername.equals(username) && inputPassword.equals(password);
    }
}

// Inventory Management Module
class Inventory {
    private HashMap<String, Integer> stock = new HashMap<>();
    private HashMap<String, Double> prices = new HashMap<>();
}
```

```

public void addStock(String productName, int quantity, double price) {
    stock.put(productName, stock.getDefault(productName, 0) + quantity);
    prices.put(productName, price);
}

public int checkStock(String productName) {
    return stock.getDefault(productName, 0);
}

public void updateStock(String productName, int quantity) {
    if (stock.containsKey(productName)) {
        stock.put(productName, quantity);
    }
}

public double getPrice(String productName) {
    return prices.getDefault(productName, 0.0);
}

public String viewStock() {
    StringBuilder sb = new StringBuilder("Current Inventory:\n");
    for (String product : stock.keySet()) {
        sb.append("Product: ").append(product)
          .append(", Quantity: ").append(stock.get(product))
          .append(", Price: Rs").append(prices.get(product)).append("\n");
    }
    return sb.toString();
}
}

```

```
// Order Management Module
```

```
class Order {
```

```
    private String orderId;
```

```
    private String productName;
```

```
    private int quantity;
```

```
    private double totalPrice;
```

```
    public Order(String orderId, String productName, int quantity, double totalPrice) {
```

```
        this.orderId = orderId;
```

```
        this.productName = productName;
```

```
        this.quantity = quantity;
```

```
        this.totalPrice = totalPrice;
```

```
    }
```

```
    public void processOrder(Inventory inventory) {
```

```
        if (inventory.checkStock(productName) >= quantity) {
```

```
            inventory.updateStock(productName, inventory.checkStock(productName) -  
quantity);
```

```
        }
```

```
    }
```

```
    public String getOrderId() {
```

```
        return orderId;
```

```
    }
```

```
    public double getTotalPrice() {
```

```
        return totalPrice;
```

```
    }
```

```
}
```

```
// Customer Module
```

```
class Customer {
```

```
    private String name;
```

```
    private String email;
```

```
    private List<Order> orderHistory = new ArrayList<>();
```

```
    public Customer(String name, String email) {
```

```
        this.name = name;
```

```
        this.email = email;
```

```
    }
```

```
    public void addOrder(Order order) {
```

```
        orderHistory.add(order);
```

```
    }
```

```
    public String viewOrderHistory() {
```

```
        StringBuilder sb = new StringBuilder("Order History for  
").append(name).append(":\n");
```

```
        for (Order order : orderHistory) {
```

```
            sb.append("Order ID: ").append(order.getId())
```

```
                .append(", Total Price: Rs").append(order.getTotalPrice()).append("\n");
```

```
        }
```

```
        return sb.toString();
```

```
    }
```

```
}
```

```
// Payment Module
```

```

class Payment {
    public String processPayment(double amount) {
        return "Payment of Rs" + amount + " processed successfully.";
    }
}

```

// Main Class

```

public class InventoryManagementSystemAWT extends Frame {
    private User admin;
    private Inventory inventory;
    private Customer customer;
    private Payment payment;

    private CardLayout cardLayout;
    private Panel cardPanel;
    private TextArea outputArea;

    public InventoryManagementSystemAWT() {
        admin = new User("admin", "admin123", "Admin");
        inventory = new Inventory();
        payment = new Payment();

        // Main frame setup
        setTitle("Inventory Management System");
        setSize(600, 400);
        setLayout(new BorderLayout());

        // Output area
        outputArea = new TextArea();
    }
}

```



```

outputArea.setEditable(false);
add(outputArea, BorderLayout.CENTER);

// Navigation buttons
Panel navigationPanel = new Panel(new FlowLayout());
Button loginButton = new Button("Login");
Button stockButton = new Button("Stock");
Button customerButton = new Button("Customer");
Button orderButton = new Button("Order");
Button reportButton = new Button("Report");
Button logoutButton = new Button("Logout");

navigationPanel.add(loginButton);
navigationPanel.add(stockButton);
navigationPanel.add(customerButton);
navigationPanel.add(orderButton);
navigationPanel.add(reportButton);
navigationPanel.add(logoutButton);

add(navigationPanel, BorderLayout.SOUTH);

// Card layout for content
cardLayout = new CardLayout();
cardPanel = new Panel(cardLayout);
add(cardPanel, BorderLayout.NORTH);

// Panels for each module
Panel loginPanel = createLoginPanel();
Panel stockPanel = createStockPanel();

```

```

Panel customerPanel = createCustomerPanel();
Panel orderPanel = createOrderPanel();
Panel reportPanel = createReportPanel();

cardPanel.add("Login", loginPanel);
cardPanel.add("Stock", stockPanel);
cardPanel.add("Customer", customerPanel);
cardPanel.add("Order", orderPanel);
cardPanel.add("Report", reportPanel);

// Button actions
loginButton.addActionListener(e -> cardLayout.show(cardPanel, "Login"));
stockButton.addActionListener(e -> cardLayout.show(cardPanel, "Stock"));
customerButton.addActionListener(e -> cardLayout.show(cardPanel,
"Customer"));
orderButton.addActionListener(e -> cardLayout.show(cardPanel, "Order"));
reportButton.addActionListener(e -> cardLayout.show(cardPanel, "Report"));
logoutButton.addActionListener(e -> outputArea.setText("Logged out
successfully.));

// Window close
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        dispose();
    }
});

setVisible(true);
}

```

```

private Panel createLoginPanel() {
    Panel panel = new Panel(new GridLayout(3, 2));
    TextField usernameField = new TextField();
    TextField passwordField = new TextField();
    passwordField.setEchoChar('*');
    Button loginButton = new Button("Login");

    loginButton.addActionListener(e -> {
        String username = usernameField.getText();
        String password = passwordField.getText();
        if (admin.login(username, password)) {
            outputArea.setText("Login successful!");
            cardLayout.show(cardPanel, "Stock");
        } else {
            outputArea.setText("Login failed. Invalid username or password.");
        }
    });

    panel.add(new Label("Username:"));
    panel.add(usernameField);
    panel.add(new Label("Password:"));
    panel.add(passwordField);
    panel.add(new Label());
    panel.add(loginButton);

    return panel;
}

```

```

private Panel createStockPanel() {
    Panel panel = new Panel(new GridLayout(4, 2));
    TextField productField = new TextField();
    TextField quantityField = new TextField();
    TextField priceField = new TextField();
    Button addStockButton = new Button("Add Stock");

    addStockButton.addActionListener(e -> {
        String product = productField.getText();
        int quantity = Integer.parseInt(quantityField.getText());
        double price = Double.parseDouble(priceField.getText());
        inventory.addStock(product, quantity, price);
        outputArea.setText("Stock added for " + product);
    });

    panel.add(new Label("Product:"));
    panel.add(productField);
    panel.add(new Label("Quantity:"));
    panel.add(quantityField);
    panel.add(new Label("Price:"));
    panel.add(priceField);
    panel.add(new Label());
    panel.add(addStockButton);

    return panel;
}

```

```

private Panel createCustomerPanel() {
    Panel panel = new Panel(new GridLayout(3, 2));

```

```

TextField nameField = new TextField();
TextField emailField = new TextField();
Button createCustomerButton = new Button("Create Customer");

createCustomerButton.addActionListener(e -> {
    String name = nameField.getText();
    String email = emailField.getText();
    customer = new Customer(name, email);
    outputArea.setText("Customer created: " + name);
});

panel.add(new Label("Name:"));
panel.add(nameField);
panel.add(new Label("Email:"));
panel.add(emailField);
panel.add(new Label());
panel.add(createCustomerButton);

return panel;
}

private Panel createOrderPanel() {
    Panel panel = new Panel(new GridLayout(3, 2));
    TextField productField = new TextField();
    TextField quantityField = new TextField();
    Button placeOrderButton = new Button("Place Order");

    placeOrderButton.addActionListener(e -> {
        if (customer == null) {

```

```

        outputArea.setText("No customer exists. Create one first.");
        return;
    }

    String product = productField.getText();
    int quantity = Integer.parseInt(quantityField.getText());

    if (inventory.checkStock(product) >= quantity) {
        double price = inventory.getPrice(product) * quantity;
        Order order = new Order(UUID.randomUUID().toString(), product,
quantity, price);
        order.processOrder(inventory);
        customer.addOrder(order);
        outputArea.setText("Order placed. Total: Rs" + price);
    } else {
        outputArea.setText("Insufficient stock for " + product);
    }
});

panel.add(new Label("Product:"));
panel.add(productField);
panel.add(new Label("Quantity:"));
panel.add(quantityField);
panel.add(new Label());
panel.add(placeOrderButton);

return panel;
}

```

```

private Panel createReportPanel() {
    Panel panel = new Panel(new GridLayout(2, 1));
    Button viewStockButton = new Button("View Inventory");
    Button viewOrderHistoryButton = new Button("View Order History");

    viewStockButton.addActionListener(e -> {
        outputArea.setText(inventory.viewStock());
    });

    viewOrderHistoryButton.addActionListener(e -> {
        if (customer == null) {
            outputArea.setText("No customer exists.");
        } else {
            outputArea.setText(customer.viewOrderHistory());
        }
    });

    panel.add(viewStockButton);
    panel.add(viewOrderHistoryButton);

    return panel;
}

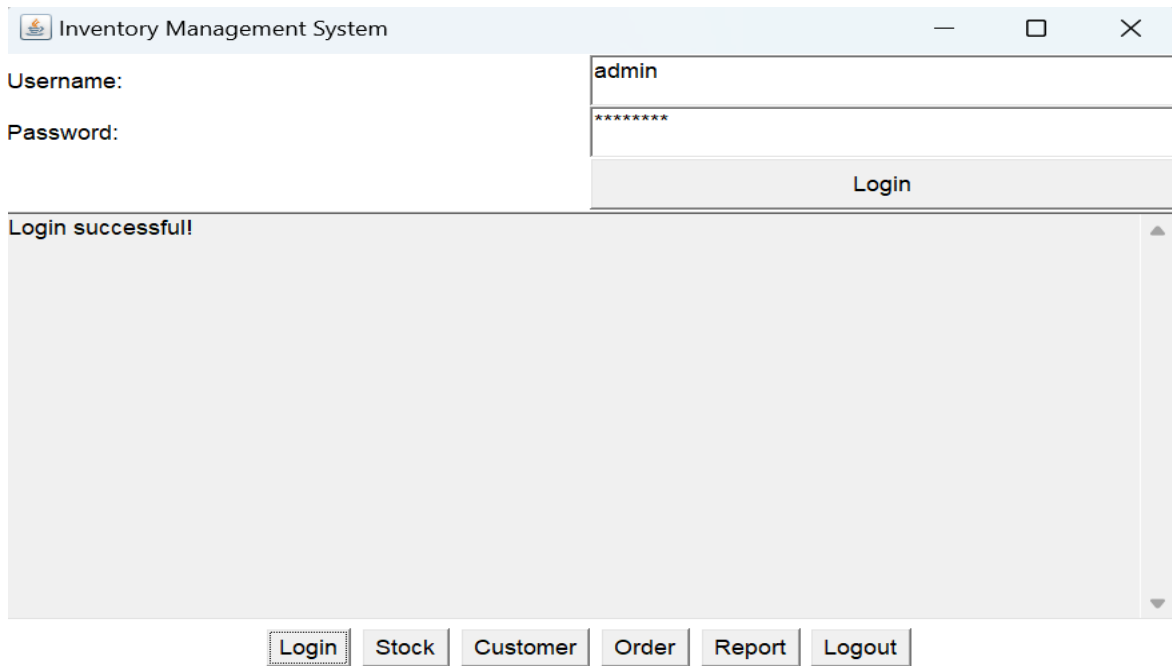
public static void main(String[] args) {
    new InventoryManagementSystemAWT();
}
}

```

APPENDIX B

(SCREENSHOTS)

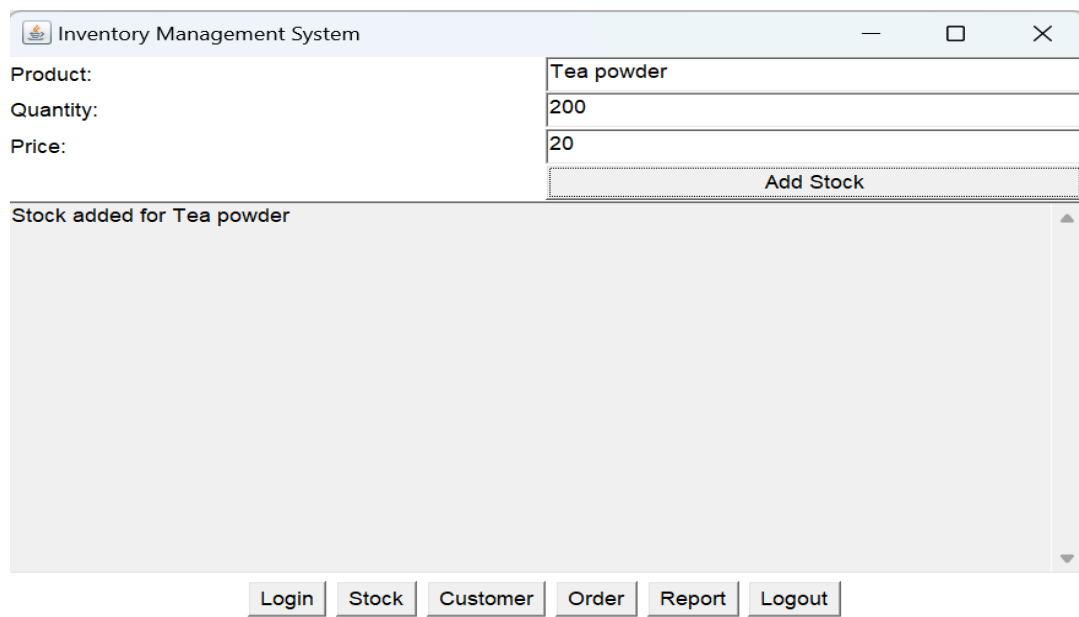
Login:



The screenshot shows a web application window titled "Inventory Management System". It features a login form with two input fields: "Username:" containing the text "admin" and "Password:" containing seven asterisks "*****". Below these fields is a "Login" button. A message "Login successful!" is displayed in a large, light gray area below the form. At the bottom of the window, there is a navigation bar with six buttons: "Login", "Stock", "Customer", "Order", "Report", and "Logout".

img 1

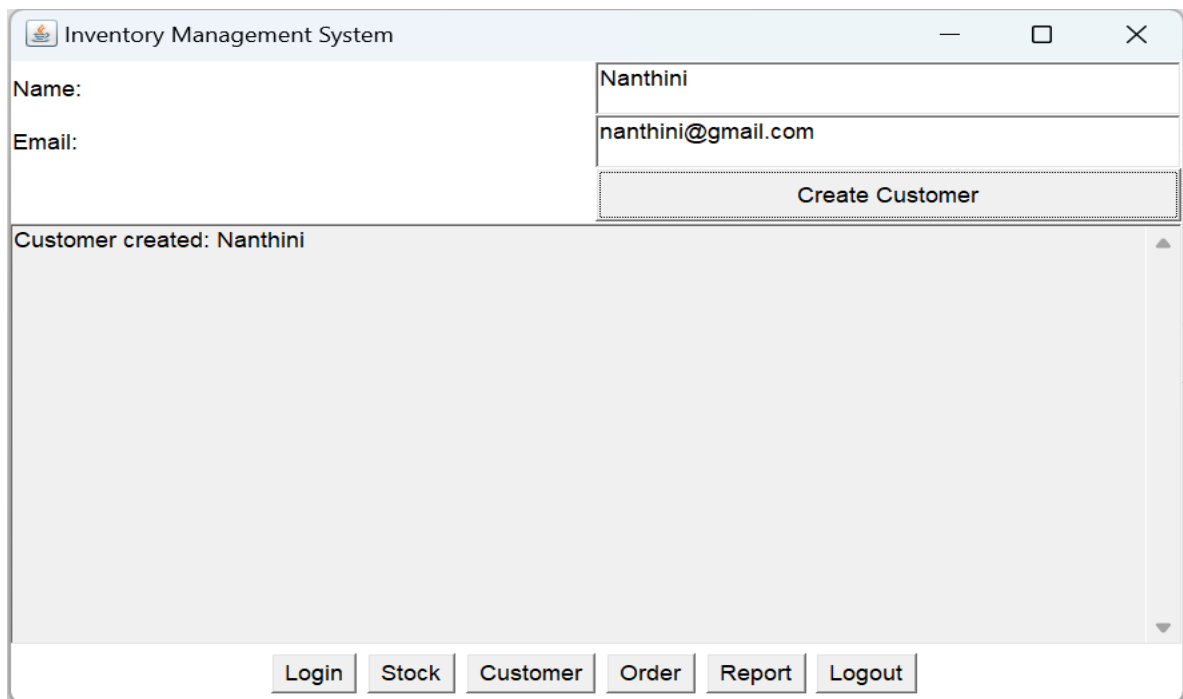
Adding stock:



The screenshot shows the same "Inventory Management System" window, but now in the "Adding stock" section. The form has three input fields: "Product:" containing "Tea powder", "Quantity:" containing "200", and "Price:" containing "20". Below these fields is an "Add Stock" button. A message "Stock added for Tea powder" is displayed in a large, light gray area below the form. The navigation bar at the bottom remains the same with buttons for "Login", "Stock", "Customer", "Order", "Report", and "Logout".

img 2

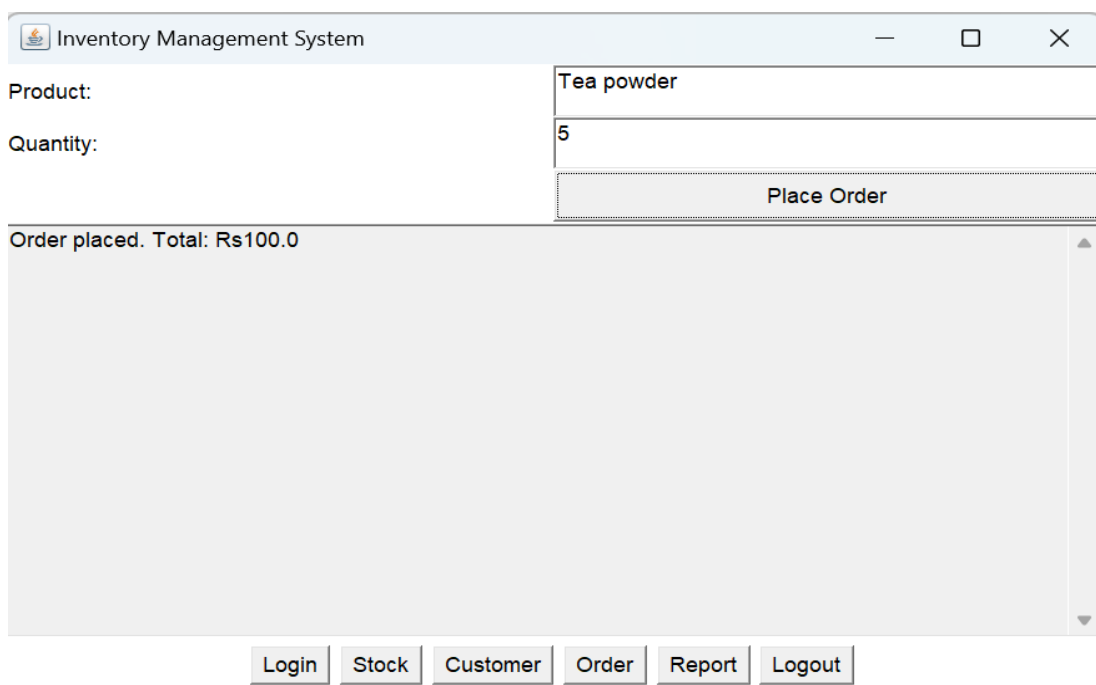
Customer details:



The screenshot shows a web application window titled "Inventory Management System". It features a form for creating a new customer. The "Name:" field contains "Nanthini" and the "Email:" field contains "nanthini@gmail.com". A "Create Customer" button is positioned below the email field. A message box at the bottom of the form area states "Customer created: Nanthini". At the bottom of the window, there is a navigation bar with buttons for "Login", "Stock", "Customer", "Order", "Report", and "Logout".

img 3

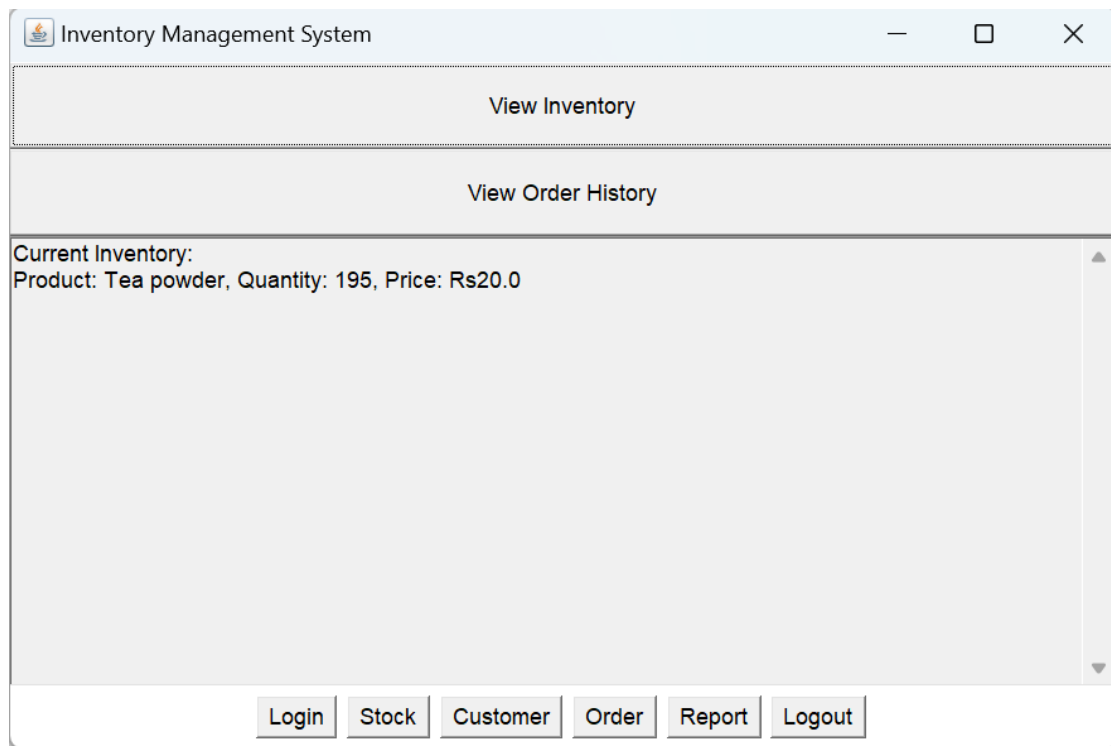
Order details:



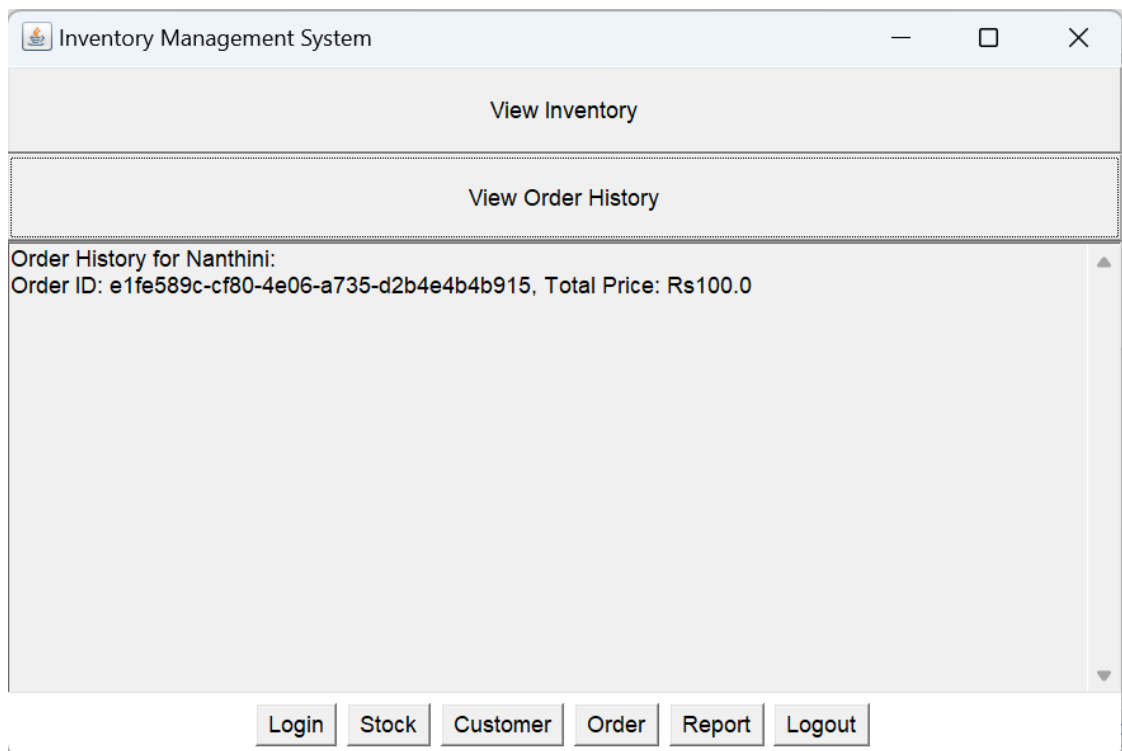
The screenshot shows a web application window titled "Inventory Management System". It features a form for placing an order. The "Product:" field contains "Tea powder" and the "Quantity:" field contains "5". A "Place Order" button is positioned below the quantity field. A message box at the bottom of the form area states "Order placed. Total: Rs100.0". At the bottom of the window, there is a navigation bar with buttons for "Login", "Stock", "Customer", "Order", "Report", and "Logout".

img 4

Generating Report:

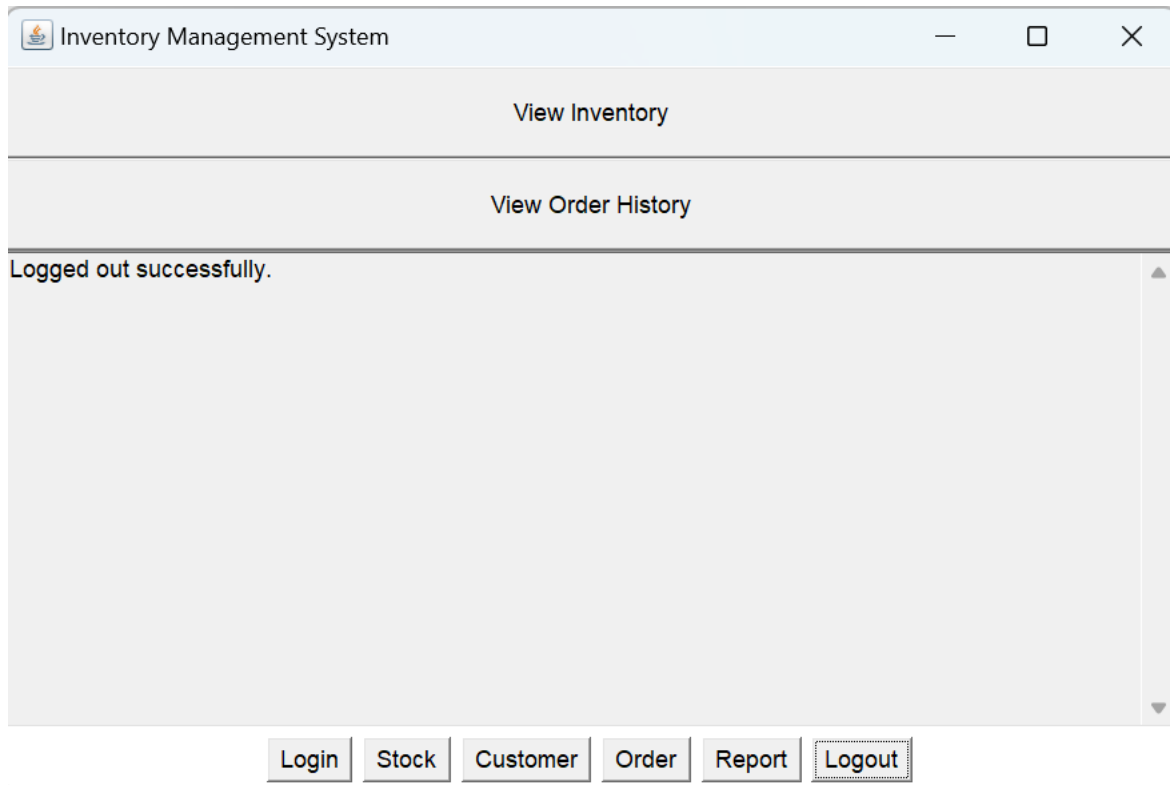


img 5



img 6

Log out:



img 7

REFERENCES:

Websites:

- <https://chatgpt.com/c/66f2446a-2754-8005-b342-2bd624e346ac>
- <https://github.com/AsjadIqbal/InventoryManagementSystem>
- <https://codewithcurious.com/projects/inventory-management-system-java/>

Youtube links:

- <https://www.youtube.com/watch?v=QoBzRvut0HA>
- <https://www.youtube.com/watch?v=rwzFrj5cWl4>

Java books:

- Java 1 crore projects

Java : The Comprehensive guide