SUPER MARKET BILLING SYSTEM

A PROJECT REPORT

Submitted by

VARUNIKA S (2303811710422175)

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 **“SUPER MARKET BILLING SYSTEM”** is the bonafide work of VARUNIKA S **(2303811710422175)** 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.,
SUPERVISOR
ASSISTANTP ROFESSOR

[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 06/12/2024

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]

INTERNAL EXAMINER EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **"SUPER MARKET BILLING 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

S. Varunika

VARUNIKA S

Place: Samayapuram

Date:06/12/2024

ACKNOWLEDGEMENT

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

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

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

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

I express our deep expression and sincere gratitude to our project supervisor **MR. 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 **Supermarket Billing System** is a robust Java-based application designed to automate and streamline the billing process in a supermarket setting. Utilizing **Java Swing** for the graphical user interface (GUI), the system offers an interactive platform where users can seamlessly manage products, maintain a shopping cart, and generate detailed bills. The system leverages two primary classes: the **Product** class, which encapsulates product details such as name and price, and the **ShoppingCart** class, which provides core functionalities for adding products, updating quantities, removing items, and calculating the total amount. The GUI is designed for simplicity and functionality, featuring a product selection panel, a dynamic shopping cart table, and a billing summary. Users can add products with specified quantities to the cart, view and manage items in real time, and remove selected items if needed. A **"Generate Bill"** feature calculates and displays a detailed breakdown, including subtotal, tax (10%), and the final total. The system ensures accuracy, ease of use, and clear feedback through error handling and user-friendly components. This project demonstrates practical implementation of **object-oriented programming principles**, **event-driven programming**, and effective use of Java Swing components. It is a scalable solution ideal for small to medium-scale retail stores, showcasing real-world applications of Java programming and GUI design. This project highlights the integration of advanced Java programming techniques with practical application design to create a functional and efficient billing solution. It showcases the potential of software-driven automation in improving retail processes, reducing human error, and delivering a professional and seamless experience for users.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
The **Supermarket Billing System** is a robust Java-based application designed to automate and streamline the billing process in a supermarket setting. Utilizing **Java Swing** for the graphical user interface (GUI), the system offers an interactive platform where users can seamlessly manage products, maintain a shopping cart, and generate detailed bills. The system leverages two primary classes: the **Product** class, which encapsulates product details such as name and price, and the **ShoppingCart** class, which provides core functionalities for adding products, updating quantities, removing items, and calculating the total amount.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

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

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1Objective

The objective of the **Supermarket Billing System** project is to develop a professional and user-friendly application that simplifies the billing process in supermarkets. This system is designed to allow users to add products to a shopping cart, specify quantities, and automatically calculate the total amount, including applicable taxes, ensuring accuracy and efficiency in billing. The application provides real-time cart management, enabling users to view, update, or remove items from the cart seamlessly. It dynamically updates the total cost whenever changes are made, ensuring clarity during the checkout process. By offering an intuitive graphical interface built with Java Swing, the system aims to streamline supermarket operations, reduce human errors, and provide a smooth and hassle-free experience for both customers and staff.

1.2Overview

The **Supermarket Billing System** is a desktop application developed using Java and Swing to streamline the checkout process in supermarkets. It provides an intuitive graphical interface for adding products, managing a shopping cart, and generating bills. Users can select predefined products, specify quantities, and see real-time updates of the cart, including itemized totals and overall billing amounts. The system calculates the total cost with tax, ensuring accuracy and efficiency in the billing process. It also offers functionalities like removing items from the cart and generating a detailed bill, making it a practical solution for improving supermarket operations.

1.3 Java Programming Concepts

1. Basic Java Concepts:

- **Classes and Objects:** Fundamental building blocks are used to represent Product and manage ShoppingCart operations.

- **Control Structures:** Applied for logical operations like validating user input and iterating over cart items for calculations.

- **String Formatting:** Utilized with String.format() to present totals, product details, and bills in a clear, user-friendly format.

2. OOP Concepts:

- **Encapsulation:** Sensitive fields in the Product class are kept private, and controlled access is provided through getter methods.
- **Abstraction:** Key functionalities such as adding/removing items and calculating totals are abstracted within the ShoppingCart class.
- **Overriding:** The toString() method in the Product class is overridden to provide meaningful product descriptions for GUI display.

3. Advanced Java Concepts:

- **Swing Framework:** Java Swing is used to create an interactive GUI, with components like JFrame, JPanel, JButton, and JTable forming the core interface.
- **Collections Framework:** The HashMap in the ShoppingCart class stores product details and quantities efficiently, enabling seamless cart management.
- **Event Handling:** Action listeners (ActionListener) are implemented to respond to user actions, such as adding products, removing items, or generating bills dynamically.

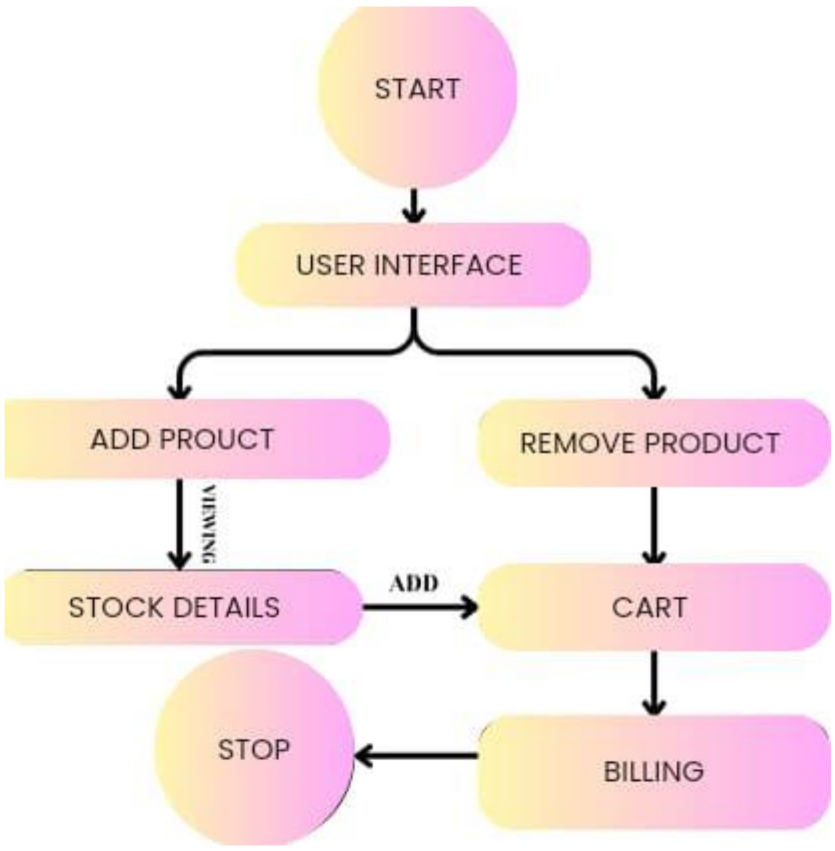
By combining these concepts, the proposed system ensures an efficient, interactive, and accurate billing process, demonstrating the application of core and advanced Java programming techniques.

CHAPTER 2
PROJECT METHODOLOGY

2.1Proposed Work

The proposed work for the **Supermarket Billing System** focuses on designing and implementing a robust application to streamline the billing process in supermarkets. The system will feature a graphical user interface using Java Swing to enable easy product selection, quantity management, and real-time cart updates. It will allow users to add products to a shopping cart, dynamically calculate totals, and generate detailed bills, including tax computations. The application will incorporate features for removing items and displaying a final summary of the bill. By leveraging object-oriented programming principles, efficient data structures, and intuitive design, the system aims to enhance the accuracy, efficiency, and user experience of the supermarket checkout process.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Product Management Module

This module defines the Product class to represent individual products. It encapsulates product details like name and price and includes methods for retrieving product information. It also ensures proper management of product attributes using OOP principles such as encapsulation and overriding (toString() method).

3.2 Shopping Cart Module

The ShoppingCart class manages the user's cart. It handles adding products with quantities, removing products, and calculating the total cost. This module uses the Java Collections Framework (HashMap) for efficient storage and retrieval of cart data and supports real-time updates to reflect changes in the cart.

3.3 User Interface Module

This module is implemented using Java Swing to create an interactive and visually appealing graphical interface. It includes components such as JFrame for the main window, JPanel for organizing different sections, JTable for displaying cart contents, and buttons for performing actions like adding products, removing items, and generating bills. Layout managers like GridLayout and BorderLayout are used to arrange components systematically, ensuring ease of use and a clean design.

3.4 Event Handling Module

This module manages user interactions by implementing event listeners (ActionListener). It handles actions such as adding products to the cart, removing selected items, and generating the final bill. It ensures that the system responds dynamically to user inputs.

3.5 Billing and Summary Module

This module calculates the total amount of the cart, applies tax, and generates a detailed bill summary. It uses formatted strings to present the subtotal, tax, and final total clearly. It also integrates with the user interface to display the bill details through dialogs.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In conclusion, the **Supermarket Billing System** effectively simplifies the billing process by offering a user-friendly, accurate, and efficient platform for managing supermarket transactions. With features like real-time cart updates, tax calculations, and bill generation, the system ensures a seamless and error-free checkout experience. Developed using Java Swing, it demonstrates the potential of modern software solutions to improve operational efficiency and customer satisfaction in retail environments. This project serves as a practical tool for supermarkets while showcasing a robust application of object-oriented programming and graphical user interface design.

4.2 FUTURE SCOPE

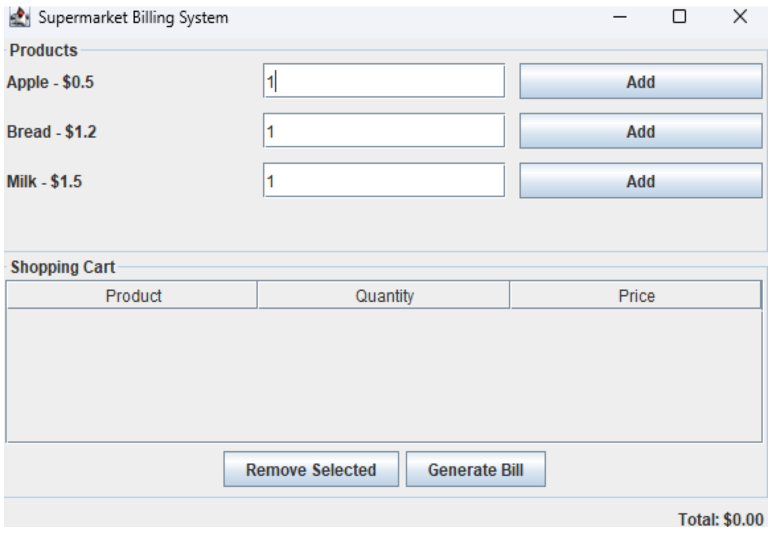
The Supermarket Billing System has significant potential for future enhancements and scalability to meet evolving requirements in the retail sector. The system can be expanded to include advanced features such as integration with a barcode scanning system for faster product entry and real-time inventory management to track stock levels. Additional functionalities like customer accounts, loyalty programs, and payment gateway integration can enhance the overall user experience. The system can also be adapted for multi-language support and deployed on other platforms, such as mobile devices or web applications, for broader accessibility. Furthermore, incorporating analytics and reporting features could provide valuable insights into sales trends and customer behavior, making it a comprehensive solution for modern supermarkets. These enhancements would not only improve operational efficiency but also make the system more versatile and competitive in the market.

APPENDIX A
(SOURCE CODE)

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

// Product Class
class Product {
    private String name;
    private double price;

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

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }

    @Override
    public String toString() {
        return name + " - $" + price;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Product product = (Product) obj;
        return name.equals(product.name);
    }

    @Override
    public int hashCode() {
        return name.hashCode();
    }
}

// ShoppingCart Class
class ShoppingCart {
    private Map<Product, Integer> cartItems = new HashMap<>();

    public void addProduct(Product product, int quantity) {
        cartItems.put(product, cartItems.getOrDefault(product, 0) + quantity);
    }

    public void removeProduct(Product product) {
        cartItems.remove(product);
    }

    public Map<Product, Integer> getCartItems() {
        return cartItems;
    }
    public double calculateTotal() {
        double total = 0;
        for (Map.Entry<Product, Integer> entry : cartItems.entrySet()) {
            total += entry.getKey().getPrice() * entry.getValue();
        }
        return total;
    }
}
public class SupermarketBillingSwing {
    private JFrame frame;
    private JTable cartTable;
    private DefaultTableModel tableModel;
    private JLabel totalLabel;
    private ShoppingCart cart;
    private Product[] products;
    public SupermarketBillingSwing() {
        cart = new ShoppingCart();
        products = new Product[] {
                new Product("Apple", 0.5),
                new Product("Bread", 1.2),
                new Product("Milk", 1.5)
        };
        frame = new JFrame("Supermarket Billing System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);
        frame.setLayout(new BorderLayout());
        JPanel productPanel = new JPanel();
        productPanel.setLayout(new GridLayout(products.length + 1, 3, 10, 10));
        productPanel.setBorder(BorderFactory.createTitledBorder("Products"));
        JLabel[] productLabels = new JLabel[products.length];
        JButton[] addButtons = new JButton[products.length];
        JTextField[] quantityFields = new JTextField[products.length];
        for (int i = 0; i < products.length; i++) {
            productLabels[i] = new JLabel(products[i].toString());
            quantityFields[i] = new JTextField("1", 5);
            addButtons[i] = new JButton("Add");
            int productIndex = i;
            addButtons[i].addActionListener(e -> {
                try {
                    int quantity = Integer.parseInt(quantityFields[productIndex].getText());
                    if (quantity > 0) {
                        cart.addProduct(products[productIndex], quantity);
                        updateCartTable();
                    } else {
                        JOptionPane.showMessageDialog(frame, "Enter a valid quantity.", "Error",
JOptionPane.ERROR_MESSAGE);
                    }
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(frame, "Invalid quantity.", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            });
             productPanel.add(productLabels[i]);
            productPanel.add(quantityFields[i]);
            productPanel.add(addButtons[i]);
        }
        JPanel cartPanel = new JPanel(new BorderLayout());
        cartPanel.setBorder(BorderFactory.createTitledBorder("Shopping Cart"));
        tableModel = new DefaultTableModel(new String[]{"Product", "Quantity", "Price"}, 0);
        cartTable = new JTable(tableModel);
        JScrollPane scrollPane = new JScrollPane(cartTable);
        JPanel cartButtonPanel = new JPanel();
        JButton removeButton = new JButton("Remove Selected");
        removeButton.addActionListener(e -> {
            int selectedRow = cartTable.getSelectedRow();
            if (selectedRow >= 0) {
                String productName = (String) tableModel.getValueAt(selectedRow, 0);
                Product productToRemove = null;
                for (Product product : products) {
                    if (product.getName().equals(productName)) {
                        productToRemove = product;
                        break;
                    }
                }
                if (productToRemove != null) {
                    cart.removeProduct(productToRemove);
     updateCartTable();
                }
            } else {
                JOptionPane.showMessageDialog(frame, "Select a product to remove.", "Error",
JOptionPane.ERROR_MESSAGE)
}
        });
        JButton generateBillButton = new JButton("Generate Bill");
        generateBillButton.addActionListener(e -> generateBill());
        cartButtonPanel.add(removeButton);
        cartButtonPanel.add(generateBillButton);

        cartPanel.add(scrollPane, BorderLayout.CENTER);
        cartPanel.add(cartButtonPanel, BorderLayout.SOUTH);
        JPanel totalPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        totalLabel = new JLabel("Total: $0.00");
        totalPanel.add(totalLabel);
        frame.add(productPanel, BorderLayout.NORTH);
        frame.add(cartPanel, BorderLayout.CENTER);
        frame.add(totalPanel, BorderLayout.SOUTH);
        frame.setVisible(true);
      }
      private void updateCartTable() {
        tableModel.setRowCount(0); // Clear the table
        for (Map.Entry<Product, Integer> entry : cart.getCartItems().entrySet()) {
            Product product = entry.getKey();
            int quantity = entry.getValue();
            tableModel.addRow(new Object[]{product.getName(), quantity, product.getPrice() *
quantity});
        }
        totalLabel.setText(String.format("Total: $%.2f", cart.calculateTotal()));
      }

      private void generateBill() {
        double total = cart.calculateTotal();
        if (total == 0) {
            JOptionPane.showMessageDialog(frame, "Your cart is empty.", "Bill",
JOptionPane.INFORMATION_MESSAGE);
        } else {
            double tax = total * 0.10; // 10% tax
            double finalTotal = total + tax;
            JOptionPane.showMessageDialog(frame,
                    String.format("Subtotal: $%.2f\nTax (10%%): $%.2f\nTotal: $%.2f", total, tax,
finalTotal),
                    "Bill", JOptionPane.INFORMATION_MESSAGE);
        }
    }
    public static void main(String[] args) {
        new SupermarketBillingSwing();
    }
}
```

APPENDIX B

(SCREENSHOTS)



1

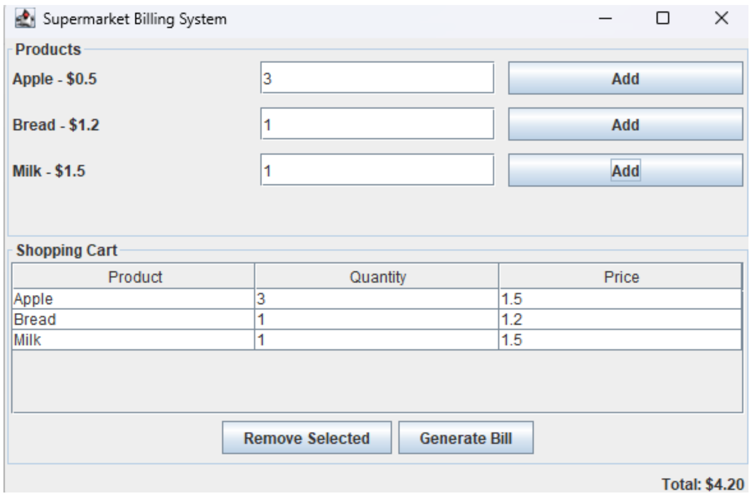
1

1

Product	Quantity	Price

Total: $0.00

ADD PRODUCT



3

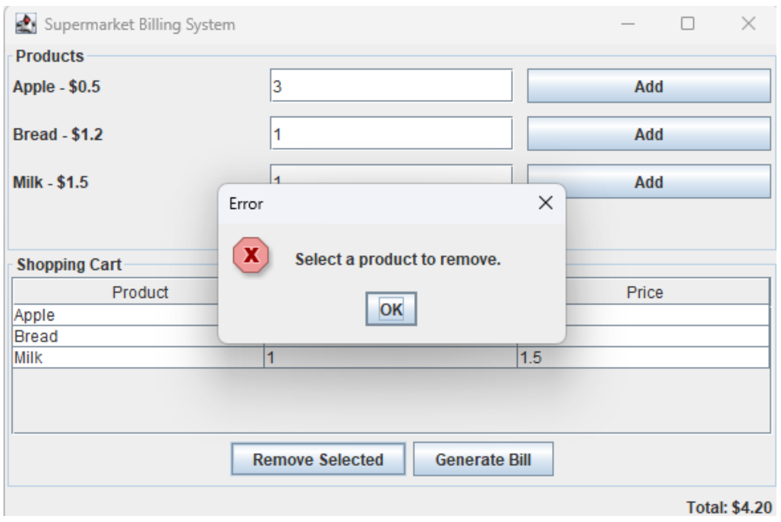
1

1

Product	Quantity	Price
Apple	3	1.5
Bread	1	1.2
Milk	1	1.5

Total: $4.20

REMOVE PRODUCT



3

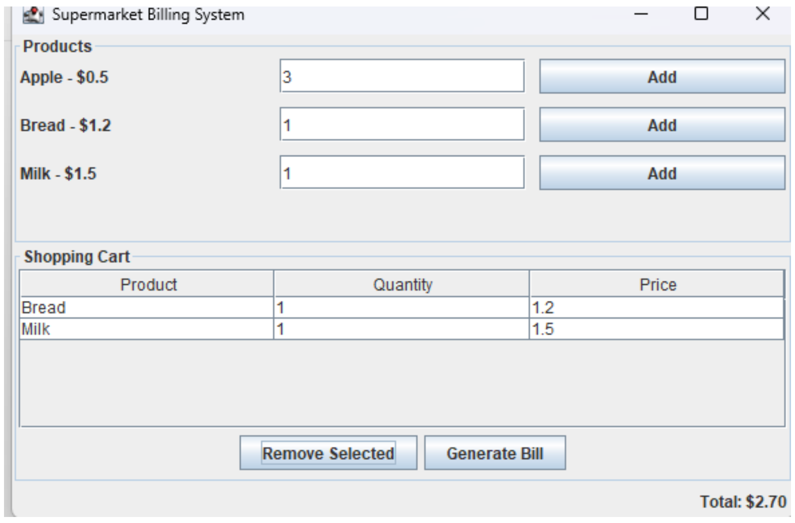
1

Select a product to remove.

Product	Quantity	Price
Apple		
Bread		
Milk	1	1.5

Total: $4.20

VIEW PRODUCT



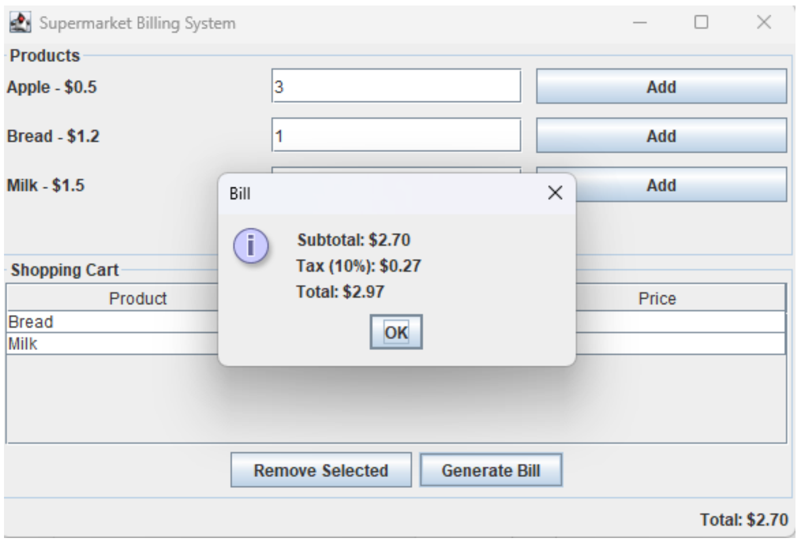
Apple - $0.5	3
Bread - $1.2	1
Milk - $1.5	1

Shopping Cart

Product	Quantity	Price
Bread	1	1.2
Milk	1	1.5

Total: $2.70

GENERATE TOTAL BILL



Apple - $0.5	3
Bread - $1.2	1
Milk - $1.5	

Subtotal: $2.70
Tax (10%): $0.27
Total: $2.97

Shopping Cart

Product	Price
Bread	
Milk	

Total: $2.70

REFERENCES

1. Deitel, P.J., & Deitel, H.M. (2006) **"Java: How to Program"**, 6th Edition, Prentice Hall, Chapter 14: Swing Components for GUI Applications, pp. 507–550. (Covers GUI design using Java Swing relevant to billing systems.)

2. Schildt, H. (2014) **"Java: The Complete Reference"**, 9th Edition, McGraw Hill, Chapter 22: Java Swing, pp. 1065–1125. (Detailed explanation of GUI components and event handling in Java.)

3.Mahajan, M. (2020) **"Building Real-World Applications with Java"**, Packt Publishing, Chapter 5: Designing Billing Systems, pp. 145–175. (Step-by-step development of a billing system in Java, including cart management and billing logic.)

4.Cornell, G., & Horstmann, C. (2005) **"Core Java Volume I - Fundamentals"**, 8th Edition, Prentice Hall, Chapter 11: AWT and Swing, pp. 469–512. (Covers the basics of building user interfaces, essential for supermarket billing systems.)