FRUIT SHOP MANAGEMENT

SYSTEM

**N . RAJAGOKUL (23322048)**

**P . DINESH KUMAR (23322016)**

**C . BARATH(23322045)**

**ADVANCED JAVA PROGRAMMING - LAB**

**COURSE TEACHER: Dr.T.Kalaiselvi, MCA.,Ph.D.,**

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS

THE GANDHIGRAM RURAL INSTITUTE

(DEEMED TO BE UNIVERSITY)

GANDHIGRAM-624 302

**MAY 2024**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NO** |
| 1. | **INTRODUCTION** | 1 |
| 2. | **SOFTWARE DESCRIPTION** | 2 |
| 3. | **SYSTEM DESIGN**   * 1. FLOW DIAGRAM   2. TABLE DESIGN   3. MODULE DESCRIPTION | 7  8  9 |
| 4. | **CODING** | 11 |
| 5. | **SCREENSHOT** | 76 |

#### **INTRODUCTION**

#### The Juicy Basket is a comprehensive Fruit Shop Management System designed to streamline the operations of a fruit shop. This system manages inventory, sales, and supplier information. It aims to enhance efficiency. The system is developed using Java JDBC (Java Database Connectivity) for database connectivity and MS Access for database management.

#### This system allows the user to log in, and on the home page, various fruit categories are listed. After selecting a Fruits, there are various types of fruits available to view the Table. The user can then choose the quantity of the preferred fruits. After selecting the fruits to purchase, they are added to the cart. The bill is generated for the purchase after Store the bill details in Database.

#### The system includes features for billing, billing history, product details, and supplier details

# **SOFTWARE DESCRIPTION :**

#### FRONT-END : Java , Swing.

#### BACK-END : Microsoft Access.

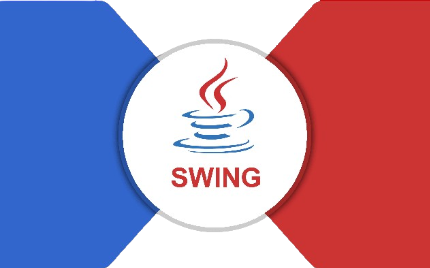
#### CONNECTIVITY : JDBC (Java Database Connectivity).

#### **Java Logo, symbol, meaning, history, PNG, brandJAVA PROGRAMMING LANGUAGE :**

#### Java, introduced by Sun Microsystems in 1995, stands as a testament to the evolution of programming languages, offering a unique blend of simplicity, power, and versatility. Developed by James Gosling and his team, Java was initially aimed at interactive television but quickly found its niche as a language that could transcend platform boundaries. Its core philosophy, encapsulated in the slogan "write once, run anywhere" (WORA), allows Java programs to be executed on any device equipped with a Java Virtual Machine (JVM), making it an indispensable tool for cross-platform development.

#### Java's architecture is designed around the concept of object-oriented programming (OOP), which promotes modular, reusable, and maintainable code. The language supports key OOP principles such as inheritance, encapsulation, polymorphism, and abstraction, which help developers create complex applications with a clear and manageable structure. Java's syntax, borrowing elements from C and C++, is both familiar and straightforward, facilitating a gentle learning curve for beginners while offering advanced features for seasoned developers.

# **JAVA SWING :**



Java Swing, part of the Java Foundation Classes (JFC), is a powerful toolkit for building graphical user interfaces (GUIs) in Java applications. Introduced by Sun Microsystems in 1997 as part of Java 1.2, Swing offers a rich set of components and functionalities that enable developers to create sophisticated, platform-independent interfaces. Unlike its predecessor, the Abstract Window Toolkit (AWT), Swing is entirely written in Java, providing a more flexible and customizable environment for GUI development.

One of the standout features of Swing is its "lightweight" component architecture. This means that Swing components are not reliant on the underlying platform's native GUI controls, allowing for a consistent look and feel across different operating systems. This cross-platform capability is a significant advantage, ensuring that applications behave similarly regardless of where they are run. Swing achieves this through its pluggable look and feel (PLAF) architecture, which allows developers to specify the appearance of their applications either through predefined themes like Metal, Nimbus, or by creating custom ones.

Swing includes a comprehensive set of GUI components, such as buttons, labels, text fields, tables, trees, and more. These components are designed to be highly customizable. For example, developers can alter the behavior and appearance of components by extending existing classes and overriding their methods. The Model-View-Controller (MVC) architecture that Swing employs further enhances this flexibility. By separating the data (Model), the visual representation (View), and the logic that handles the interaction (Controller), Swing components allow for more manageable and scalable code.

# **JDBC DRIVER :**



Java Database Connectivity (JDBC) is a pivotal API introduced by Sun Microsystems that revolutionized how Java applications interact with databases. Since its debut in 1997 as part of JDK 1.1, JDBC has provided developers with a standardized method for connecting to and executing operations on a variety of database management systems (DBMS). The core strength of JDBC lies in its ability to enable database-agnostic development. By adhering to the JDBC API, developers can write Java code that interacts with different databases without needing to tailor their code to specific DBMS features.

JDBC's architecture comprises several key components: the Driver Manager, which loads database drivers dynamically and establishes connections; the Connection interface, representing a session with a specific database; the Statement, PreparedStatement, and CallableStatement interfaces, which are used to execute SQL queries and updates; and the ResultSet interface, which holds data retrieved from the database and allows traversal of the query results.

A primary advantage of JDBC is its support for a wide range of SQL operations, including data querying, updating, and schema manipulation. JDBC also facilitates transaction management, allowing developers to commit or roll back a series of operations, thus ensuring data integrity. Advanced features such as batch updates enhance performance by reducing the number of database round-trips, while connection pooling improves resource management and scalability by reusing database connections.

# **3.SYSTEM DESIGN:**

# **3.1FLOWDIAGM:**

Payment Page

IteamName,

Selling Price

Billing History

Main Frame

Login Page

Billing Page

Supplier Details

Product Details

Index Page

# **3.2 TABLE DESIGN :**

# **DataBaseName:FruitDB.accb**

# **Table Name : BillingHistory**

|  |  |
| --- | --- |
| Fieldname | Data type |
| BillNumber | Number |
| TotalAmount | Currency |
| PaymentDate | Date/Time |
| ReceiveAmount | Currency |
| PaymentMethod | Currency |
| BalanceAmount | Currency |

**Table Name : Dealer**

|  |  |
| --- | --- |
| Fieldname | Data type |
| Name | Long Text |
| ImportItem | Long Text |
| Mobile | Number |
| Email | Long Text |
| Place | Long Text |

**Table Name : Product**

|  |  |
| --- | --- |
| Fieldname | Data type |
| ItemName | Long Text |
| SupplierName | Long Text |
| DOReceiving | Date/Time |
| DOExpiry | Date/Time |
| OrigPrice | Currency |
| SellingPrice | Currency |
| Qty | Number |
| TotalAmount | Currency |
| SellAmt | Currency |

# **3.3 MODULE DESCRIPTION :**

* User Authentication
* Bill
* Payment
* Product Details
* Billing History
* Supplier Details
* **User Authentication**

**Login**: This function allows users to log in using their credentials (username and password). The system uses Java Swing components such as JTextField for the username and JPasswordField for the password. User name & password defuilt in a code and Verified.

**Role-Based Access**: Grants different levels of access (e.g., admin, manager,user) based on user roles defined in the system.

* **Billing**

**Select the Fruit**: Choose the desired fruit from the available options in the system.

**Enter the Quantity**: Specify the quantity of each selected fruit.

**Add to Cart**: Add the selected fruits with the specified quantities to your shopping cart.

**Calculate Total**: The system will automatically calculate the total amount, including any applicable taxes and discounts.

**Click Pay Button**: Once you have reviewed the cart and confirmed the total amount, click the "Pay" button to finalize the purchase and generate the invoice

### Payment

* **Payment Methods**: Supports multiple payment methods such as credit/debit cards, net banking, UPI, and digital wallets.
* **Payment Processing**: Integrates with payment gateways to securely process payments.
* **Transaction History**: Records details of each transaction for future reference and auditing.
* **Refund and Cancellation**: Manages refunds and cancellations, updating billing and inventory records as necessary.

### Billing History

* **Transaction Records**: Stores details of all completed transactions, including dates, products purchased, amounts, and payment methods.
* **User Access**: Enables users to view their own purchase history.
* **Administrative Access**: Allows administrators to view and manage billing records, facilitating tasks like generating reports and conducting audits.

### Supplier Details

* **Supplier Database**: Maintains a database of supplier information, including names, contact details, and the types of products supplied.
* **Order Management**: Tracks orders placed with suppliers, including delivery schedules and received stock.
* **Performance Monitoring**: Monitors supplier performance based on delivery times, product quality, and reliability.
* **Communication Log**: Keeps a log of communications with suppliers for reference in case of disputes or issues.

### Product Details:

* **Product Catalog**: Displays all available fruits categorized by type (e.g., apples, oranges, bananas).
* **Product Information**: Includes detailed descriptions, prices, availability, and nutritional information for each product.
* **Search and Filter**: Allows users to search for specific products and apply filters based on categories, price ranges, and availability.
* **Inventory Management**: Updates stock levels based on sales and restocks, ensuring accurate inventory tracking.

**4.CODING**

(indexpage.java)

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**import java.text.SimpleDateFormat;**

**import java.util.Date;**

**public class Aindexpage extends JFrame {**

**public JFrame f;**

**private JLabel headerLabel, footerLabel;**

**private JButton welcomeButton;**

**public Aindexpage() {**

**setLayout(null);**

**ImageIcon backgroundImage = new ImageIcon("B:\\fruit\\image\\Aindeximg.jfif");**

**JLabel backgroundLabel = new JLabel(backgroundImage);**

**backgroundLabel.setBounds(0, 0, 800, 600);**

**add(backgroundLabel);**

**JPanel headerPanel = new JPanel();**

**headerPanel.setBackground(Color.GRAY);**

**headerPanel.setLayout(new BorderLayout());**

**headerPanel.setBounds(0, 0, 800, 80);**

**backgroundLabel.add(headerPanel);**

**headerLabel = new JLabel("THE JUICY BASKET");**

**headerLabel.setFont(new Font("CAMBRIA", Font.BOLD,24));**

**headerLabel.setForeground(Color.WHITE);**

**headerLabel.setHorizontalAlignment(JLabel.CENTER);**

**headerPanel.add(headerLabel, BorderLayout.CENTER);**

**JLabel headerlJLabel = new JLabel();**

**headerlJLabel.setHorizontalAlignment(JLabel.RIGHT);**

**headerlJLabel.setBounds(500, 0, 280,100);**

**headerlJLabel.setForeground(Color.WHITE);**

**headerPanel.add(headerlJLabel, BorderLayout.LINE\_END);**

**JPanel footerPanel = new JPanel();**

**footerPanel.setBackground(Color.GRAY);**

**footerPanel.setLayout(new BorderLayout());**

**footerPanel.setBounds(0, 550, 800, 50);**

**backgroundLabel.add(footerPanel);**

**footerLabel = new JLabel("Copyright © 2024 DK Software Pvt. Ltd.TM");**

**footerLabel.setForeground(Color.WHITE);**

**footerLabel.setHorizontalAlignment(JLabel.CENTER);**

**footerPanel.add(footerLabel, BorderLayout.CENTER);**

**welcomeButton = new JButton("Welcome");**

**welcomeButton.setBounds(350, 250, 100, 30);**

**welcomeButton.setBackground(new Color(255, 0, 0)); welcomeButton.addMouseListener(new java.awt.event.MouseAdapter() {**

**public void mouseEntered(java.awt.event.MouseEvent evt) {**

**welcomeButton.setBackground(new Color(0, 255, 0)); // Change to green color**

**}**

**public void mouseExited(java.awt.event.MouseEvent evt) {**

**welcomeButton.setBackground(new Color(255, 0, 0));**

**}**

**});**

**backgroundLabel.add(welcomeButton);**

**welcomeButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**LoginFrame indexA = new LoginFrame();**

**indexA.setVisible(true);**

**}**

**});**

**setTitle("Fruit Shop Management");**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setSize(819, 638); // Set preferred size of the window**

**setLocationRelativeTo(null); // Center the window**

**setVisible(true);**

**Timer timer = new Timer(1000, e -> updateTime(headerlJLabel));**

**timer.start();**

**}**

**private void updateTime(JLabel label) {**

**SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");**

**String dateTime = dateFormat.format(new Date());**

**label.setText(dateTime);**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(Aindexpage::new);**

**}**

(LoginFrame.java)

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**import java.text.SimpleDateFormat;**

**import java.util.Date;**

**public class LoginFrame extends JFrame implements ActionListener {**

**private JTextField usernameField;**

**private JPasswordField passwordField;**

**private JButton loginButton;**

**private JLabel timeDateLabel;**

**public LoginFrame() {**

**setTitle("Fruit Shop Management");**

**setSize(400, 250); // Increased the size of the window**

**setDefaultCloseOperation(EXIT\_ON\_CLOSE);**

**setLocationRelativeTo(null);**

**getContentPane().setBackground(Color.LIGHT\_GRAY); // Set background color**

**JPanel headerPanel = new JPanel();**

**headerPanel.setBackground(Color.GRAY);**

**headerPanel.setLayout(new BorderLayout());**

**JLabel headerLabel = new JLabel("Welcome to Login ");**

**headerLabel.setForeground(Color.WHITE); // Set text color**

**headerLabel.setHorizontalAlignment(SwingConstants.CENTER);**

**headerPanel.add(headerLabel, BorderLayout.CENTER);**

**timeDateLabel = new JLabel(getDateTime());**

**timeDateLabel.setForeground(Color.WHITE);**

**headerPanel.add(timeDateLabel, BorderLayout.EAST);**

**JPanel footerPanel = new JPanel();**

**footerPanel.setBackground(Color.GRAY);**

**JLabel footerLabel = new JLabel("Copyright © 2024 DK Software Pvt. Ltd.TM");**

**footerLabel.setForeground(Color.WHITE); // Set text color**

**footerPanel.add(footerLabel);**

**JPanel panel = new JPanel();**

**panel.setLayout(null);**

**panel.setBackground(Color.WHITE); // Set background color**

**JLabel usernameLabel = new JLabel("Username:");**

**JLabel passwordLabel = new JLabel("Password:");**

**usernameField = new JTextField();**

**passwordField = new JPasswordField();**

**loginButton = new JButton("Login");**

**loginButton.addActionListener(this);**

**// Set bounds for each component**

**usernameLabel.setBounds(20, 20, 80, 25);**

**passwordLabel.setBounds(20, 50, 80, 25);**

**usernameField.setBounds(100, 20, 160, 25);**

**passwordField.setBounds(100, 50, 160, 25);**

**loginButton.setBounds(100, 80, 80, 25);**

**panel.add(usernameLabel);**

**panel.add(usernameField);**

**panel.add(passwordLabel);**

**panel.add(passwordField);**

**panel.add(loginButton);**

**add(headerPanel, BorderLayout.NORTH);**

**add(footerPanel, BorderLayout.SOUTH);**

**add(panel, BorderLayout.CENTER);**

**setVisible(true);**

**// Update time and date every second**

**Timer timer = new Timer(1000, e -> timeDateLabel.setText(getDateTime()));**

**timer.start();**

**}**

**public void actionPerformed(ActionEvent e) {**

**if (e.getSource() == loginButton) {**

**String username = usernameField.getText();**

**String password = new String(passwordField.getPassword());**

**// You can perform your authentication logic here**

**if (username.equals("admin") && password.equals("admin")) {**

**// Dispose the current login frame**

**dispose();**

**// Open the new frame (Replace NewFrame with the name of your next frame class)**

**MainFrame newFrame = new MainFrame();**

**newFrame.setVisible(true);**

**} else {**

**JOptionPane.showMessageDialog(this, "Invalid username or password. Please try again.");**

**}**

**}**

**}**

**private String getDateTime() {**

**SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");**

**return dateFormat.format(new Date());**

**}**

**public static void main(String[] args) {**

**new LoginFrame();**

**}**

**}**

(MainFrame.java)

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**public class MainFrame extends JFrame {**

**private JLabel timeLabel;**

**private JLabel dateLabel;**

**private JButton billingButton;**

**private JButton salesReportButton;**

**private JButton productSalesButton;**

**private JButton customerButton;**

**private JButton logoutButtonFooter;**

**private JLabel copyrightLabel;**

**private Image backgroundImage;**

**public MainFrame() {**

**setTitle("Fruit Shop Management");**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setSize(800, 620);**

**backgroundImage = new ImageIcon("B:\\fruit\\image\\main10.jpg").getImage();**

**JLabel backgroundLabel = new JLabel(new ImageIcon(backgroundImage));**

**backgroundLabel.setBounds(0, 0, 880, 600);**

**setContentPane(backgroundLabel);**

**backgroundLabel.setLayout(null);**

**JPanel headerPanel = new JPanel(null);**

**headerPanel.setBackground(Color.GRAY);**

**headerPanel.setBounds(0, 0, 800, 50);**

**backgroundLabel.add(headerPanel);**

**JLabel headerLabel = new JLabel("THE JUICY BASKET");**

**headerLabel.setFont(new Font("CAMBRIA", Font.BOLD, 24));**

**headerLabel.setForeground(Color.WHITE);**

**headerLabel.setHorizontalAlignment(JLabel.CENTER);**

**headerLabel.setBounds(200, 10, 400, 30);**

**headerPanel.add(headerLabel);**

**dateLabel = new JLabel();**

**dateLabel.setBounds(600, 10, 100, 30);**

**dateLabel.setForeground(Color.WHITE);**

**headerPanel.add(dateLabel);**

**timeLabel = new JLabel();**

**timeLabel.setBounds(700, 10, 100, 30);**

**timeLabel.setForeground(Color.WHITE);**

**headerPanel.add(timeLabel);**

**int buttonWidth = 180;**

**int buttonHeight = 80;**

**int buttonMargin = 20;**

**int totalWidth = (buttonWidth + buttonMargin) \* 2 - buttonMargin;**

**int totalHeight = (buttonHeight + buttonMargin) \* 2 - buttonMargin;**

**int xPos = (getWidth() - totalWidth) / 2;**

**int yPos = (getHeight() - totalHeight) / 2;**

**billingButton = createButtonWithIcon("Billing", "B:\\fruit\\image\\billing.png");**

**billingButton.setBounds(xPos, yPos, buttonWidth, buttonHeight);**

**billingButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**openBilling();**

**}**

**});**

**backgroundLabel.add(billingButton);**

**xPos += buttonWidth + buttonMargin;**

**salesReportButton = createButtonWithIcon("Billing History", "B:\\fruit\\image\\sales.png");**

**salesReportButton.setBounds(xPos, yPos, buttonWidth, buttonHeight);**

**salesReportButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**openBillingHis();**

**}**

**});**

**backgroundLabel.add(salesReportButton);**

**xPos -= buttonWidth + buttonMargin;**

**yPos += buttonHeight + buttonMargin;**

**productSalesButton = createButtonWithIcon("Suppliers Details", "B:\\fruit\\image\\supplier.png");**

**productSalesButton.setBounds(xPos, yPos, buttonWidth, buttonHeight);**

**productSalesButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**opensupplierPage();**

**}**

**});**

**backgroundLabel.add(productSalesButton);**

**xPos += buttonWidth + buttonMargin;**

**customerButton = createButtonWithIcon("Products", "B:\\fruit\\image\\prodect.png");**

**customerButton.setBounds(xPos, yPos, buttonWidth, buttonHeight);**

**customerButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**openProduct();**

**}**

**});**

**backgroundLabel.add(customerButton);**

**JPanel footerPanel = new JPanel(null);**

**footerPanel.setBackground(Color.GRAY);**

**footerPanel.setBounds(0, getHeight() - 88, 800, 50);**

**backgroundLabel.add(footerPanel);**

**logoutButtonFooter = new JButton("Logout");**

**logoutButtonFooter.setBounds(10, 10, 80, 30);**

**footerPanel.add(logoutButtonFooter);**

**logoutButtonFooter.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**openindexPage();**

**}**

**});**

**copyrightLabel = new JLabel("Copyright © 2024 DK Software Pvt. Ltd.TM", SwingConstants.CENTER);**

**copyrightLabel.setBounds(200, 10, 400, 30);**

**copyrightLabel.setForeground(Color.WHITE);**

**footerPanel.add(copyrightLabel);**

**updateTimeAndDate();**

**Timer timer = new Timer(1000, e -> updateTimeAndDate());**

**timer.start();**

**}**

**private JButton createButtonWithIcon(String text, String imagePath) {**

**ImageIcon icon = new ImageIcon(imagePath);**

**Image img = icon.getImage();**

**Image resizedImg = img.getScaledInstance(30, 30, Image.SCALE\_SMOOTH); // Resize the image to fit the button**

**icon = new ImageIcon(resizedImg);**

**JButton button = new JButton(text, icon);**

**button.setHorizontalTextPosition(SwingConstants.CENTER);**

**button.setVerticalTextPosition(SwingConstants.BOTTOM);**

**return button;**

**}**

**private void openBilling() {**

**try {**

**Process compileProcess = Runtime.getRuntime().exec("javac Billing.java");**

**int compileResult = compileProcess.waitFor();**

**if (compileResult != 0) {**

**throw new RuntimeException("Compilation failed");**

**}**

**Process runProcess = Runtime.getRuntime().exec("java -cp \"B:\\fruit\\\*;\" Billing");**

**int runResult = runProcess.waitFor();**

**if (runResult != 0) {**

**throw new RuntimeException("Execution failed");**

**}**

**} catch (Exception ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to open Customer Page: " + ex.getMessage());**

**}**

**}**

**private void openBillingHis() {**

**try {**

**Process compileProcess = Runtime.getRuntime().exec("javac BillingHis.java");**

**int compileResult = compileProcess.waitFor();**

**if (compileResult != 0) {**

**throw new RuntimeException("Compilation failed");**

**}**

**Process runProcess = Runtime.getRuntime().exec("java -cp \"B:\\fruit\\\*;\" BillingHis");**

**int runResult = runProcess.waitFor();**

**if (runResult != 0) {**

**throw new RuntimeException("Execution failed");**

**}**

**} catch (Exception ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to open Product Page: " + ex.getMessage());**

**}**

**}**

**private void openProduct() {**

**try {**

**Process compileProcess = Runtime.getRuntime().exec("javac Prodect.java");**

**int compileResult = compileProcess.waitFor();**

**if (compileResult != 0) {**

**throw new RuntimeException("compile failed");**

**}**

**Process runProcess = Runtime.getRuntime().exec("java -cp \"B:\\fruit\\\*;\" Prodect");**

**int runResult = runProcess.waitFor();**

**if (runResult != 0) {**

**throw new RuntimeException("Execution failed");**

**}**

**} catch (Exception ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to open Product Page: " + ex.getMessage());**

**}**

**}**

**private void opensupplierPage() {**

**try {**

**Process compileProcess = Runtime.getRuntime().exec("javac supplierPage.java");**

**int compileResult = compileProcess.waitFor();**

**if (compileResult != 0) {**

**throw new RuntimeException("Compilation failed");**

**}**

**Process runProcess = Runtime.getRuntime().exec("java -cp \"B:\\fruit\\\*;\" supplierPage");**

**int runResult = runProcess.waitFor();**

**if (runResult != 0) {**

**throw new RuntimeException("Execution failed");**

**}**

**} catch (Exception ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to open Customer Page: " + ex.getMessage());**

**}**

**}**

**private void openindexPage() {**

**this.dispose();**

**indexpage indexPage = new indexpage();**

**indexPage.setVisible(true);**

**}**

**private void updateTimeAndDate() {**

**timeLabel.setText(new java.text.SimpleDateFormat("hh:mm:ss a").format(new java.util.Date()));**

**dateLabel.setText(new java.text.SimpleDateFormat("yyyy-MM-dd").format(new java.util.Date()));**

**}**

**public static void main(String[] args) {**

**MainFrame frame = new MainFrame();**

**frame.setVisible(true);**

**}**

(Billing.java)

**import javax.swing.\*;**

**import javax.swing.table.DefaultTableModel;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**import java.text.DecimalFormat;**

**import java.time.LocalDateTime;**

**import java.time.format.DateTimeFormatter;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.sql.\*;**

**public class Billing extends JFrame implements ActionListener {**

**private JTextField quantityField, priceField;**

**private JButton addButton, calculateButton, paymentButton, deleteButton, logoutButton;**

**private JTextArea billTextArea;**

**private double totalCost = 0;**

**private JTable fruitTable;**

**private DefaultTableModel fruitTableModel;**

**private JLabel currentDateTimeLabel;**

**private JComboBox<String> quantityUnitComboBox;**

**private String[] fruitsFromDatabase;**

**public Billing() {**

**setTitle("Fruit Shop Billing");**

**setSize(550, 600);**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setLayout(new BorderLayout());**

**// Initialize fruits data from database**

**initializeFruitsFromDatabase();**

**// Header Panel with Current DateTime and Logout Button**

**JPanel headerPanel = new JPanel(new BorderLayout());**

**headerPanel.setBackground(Color.GRAY);**

**JLabel headerLabel = new JLabel("Billing", JLabel.CENTER);**

**headerLabel.setFont(new Font("CAMBRIA", Font.BOLD, 24));**

**headerLabel.setForeground(Color.WHITE);**

**headerPanel.add(headerLabel, BorderLayout.CENTER);**

**// Time & Date**

**currentDateTimeLabel = new JLabel();**

**headerPanel.add(currentDateTimeLabel, BorderLayout.EAST);**

**updateDateTime();**

**Timer timer = new Timer(1000, e -> updateDateTime());**

**timer.start();**

**// Footer Panel with Logout Button**

**JPanel footerPanel = new JPanel(new BorderLayout());**

**footerPanel.setBackground(Color.GRAY);**

**logoutButton = new JButton("Logout");**

**logoutButton.addActionListener(this);**

**footerPanel.add(logoutButton, BorderLayout.EAST);**

**JLabel copyrightLabel = new JLabel("2024 © DK Software Pvt. Ltd.TM", JLabel.CENTER);**

**copyrightLabel.setForeground(Color.WHITE);**

**footerPanel.add(copyrightLabel, BorderLayout.CENTER);**

**// Main Panel**

**JPanel mainPanel = new JPanel();**

**mainPanel.setLayout(null);**

**JLabel quantityLabel = new JLabel("Quantity:");**

**quantityLabel.setBounds(20, 180, 80, 20);**

**quantityField = new JTextField();**

**quantityField.setBounds(120, 180, 100, 20);**

**JLabel quantityUnitLabel = new JLabel("Unit:");**

**quantityUnitLabel.setBounds(240, 180, 80, 20);**

**quantityUnitComboBox = new JComboBox<>(new String[]{"kilograms", "grams"});**

**quantityUnitComboBox.setBounds(280, 180, 100, 20);**

**JLabel priceLabel = new JLabel("Price per unit:");**

**priceLabel.setBounds(20, 200, 100, 20);**

**priceField = new JTextField();**

**priceField.setBounds(120, 200, 100, 20);**

**priceField.setEditable(false);**

**addButton = new JButton("Add to Cart");**

**addButton.setBounds(20, 250, 120, 25);**

**addButton.addActionListener(this);**

**calculateButton = new JButton("Calculate Total");**

**calculateButton.setBounds(150, 250, 150, 25);**

**calculateButton.addActionListener(this);**

**paymentButton = new JButton("Payment");**

**paymentButton.setBounds(320, 250, 100, 25);**

**paymentButton.addActionListener(this);**

**deleteButton = new JButton("Delete All");**

**deleteButton.setBounds(20, 450, 120, 25);**

**deleteButton.addActionListener(this);**

**billTextArea = new JTextArea();**

**JScrollPane billingScrollPane = new JScrollPane(billTextArea);**

**billingScrollPane.setBounds(20, 300, 300, 150);**

**// Fruit List**

**JLabel fruitHeaderLabel = new JLabel("Fruits");**

**fruitHeaderLabel.setBounds(20, 20, 100, 20);**

**fruitHeaderLabel.setFont(new Font("Arial", Font.BOLD, 14));**

**// Initialize fruitTableModel**

**fruitTableModel = new DefaultTableModel();**

**fruitTableModel.addColumn("Fruit Name");**

**fruitTableModel.addColumn("Price");**

**// Populate fruitTableModel with data from the database**

**for (String fruit : fruitsFromDatabase) {**

**String[] parts = fruit.split(",");**

**fruitTableModel.addRow(parts);**

**}**

**// Create fruitTable using fruitTableModel**

**fruitTable = new JTable(fruitTableModel);**

**fruitTable.getSelectionModel().addListSelectionListener(e -> updatePriceField()); // Update priceField when a fruit is selected**

**// Add fruitTable to a JScrollPane**

**JScrollPane fruitScrollPane = new JScrollPane(fruitTable);**

**fruitScrollPane.setBounds(20, 40, 200, 100);**

**// Adding components to main panel**

**mainPanel.add(quantityLabel);**

**mainPanel.add(quantityField);**

**mainPanel.add(quantityUnitLabel);**

**mainPanel.add(quantityUnitComboBox);**

**mainPanel.add(priceLabel);**

**mainPanel.add(priceField);**

**mainPanel.add(addButton);**

**mainPanel.add(calculateButton);**

**mainPanel.add(paymentButton);**

**mainPanel.add(deleteButton);**

**mainPanel.add(billingScrollPane);**

**mainPanel.add(fruitHeaderLabel);**

**mainPanel.add(fruitScrollPane);**

**// Add panels to the frame**

**add(headerPanel, BorderLayout.NORTH);**

**add(footerPanel, BorderLayout.SOUTH);**

**add(mainPanel, BorderLayout.CENTER);**

**}**

**private void initializeFruitsFromDatabase() {**

**fruitsFromDatabase = new String[0]; // Initialize the array**

**try {**

**// Database connection URL**

**String DATABASE\_URL = "jdbc:ucanaccess://B://fruit//FruitDB.accdb";**

**// Establishing the database connection**

**Connection connection = DriverManager.getConnection(DATABASE\_URL);**

**// Creating a statement for executing SQL queries**

**Statement statement = connection.createStatement();**

**// Executing the SQL query to retrieve fruit names and selling prices**

**ResultSet resultSet = statement.executeQuery("SELECT ItemName, SellingPrice FROM Product");**

**// Creating a list to hold the fruit names**

**List<String> fruitsList = new ArrayList<>();**

**// Iterating through the result set and adding fruit names to the list**

**while (resultSet.next()) {**

**String fruitName = resultSet.getString("ItemName");**

**double sellingPrice = resultSet.getDouble("SellingPrice");**

**fruitsList.add(fruitName + "," + sellingPrice); // Changed the format to comma-separated**

**}**

**// Closing the result set, statement, and connection**

**resultSet.close();**

**statement.close();**

**connection.close();**

**// Convert the list to an array**

**fruitsFromDatabase = fruitsList.toArray(new String[0]);**

**} catch (SQLException e) {**

**e.printStackTrace(); // Printing the stack trace of the exception**

**JOptionPane.showMessageDialog(this, "Error accessing database.", "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void updatePriceField() {**

**int selectedRow = fruitTable.getSelectedRow();**

**if (selectedRow >= 0) {**

**String price = (String) fruitTableModel.getValueAt(selectedRow, 1);**

**priceField.setText(price);**

**}**

**}**

**public void actionPerformed(ActionEvent e) {**

**if (e.getSource() == addButton) {**

**addFruit();**

**} else if (e.getSource() == calculateButton) {**

**calculateTotal();**

**} else if (e.getSource() == paymentButton) {**

**printBill();**

**} else if (e.getSource() == deleteButton) {**

**deleteAll();**

**} else if (e.getSource() == logoutButton) {**

**dispose(); // Close the current window**

**new MainFrame().setVisible(true); // Assuming LoginFrame is your login window class**

**}**

**}**

**private void updateDateTime() {**

**LocalDateTime now = LocalDateTime.now();**

**DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss a");**

**String formattedDateTime = now.format(formatter);**

**currentDateTimeLabel.setText(formattedDateTime);**

**currentDateTimeLabel.setForeground(Color.WHITE);**

**}**

**private void addFruit() {**

**int selectedRow = fruitTable.getSelectedRow();**

**if (selectedRow == -1) {**

**JOptionPane.showMessageDialog(this, "Please select a fruit.", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**String fruitName = (String) fruitTableModel.getValueAt(selectedRow, 0);**

**double quantity = Double.parseDouble(quantityField.getText());**

**double price = Double.parseDouble(priceField.getText());**

**String quantityUnit = (String) quantityUnitComboBox.getSelectedItem();**

**double cost = (quantityUnit.equals("kilograms")) ? quantity \* price : (quantity / 1000.0) \* price;**

**double amountPerUnit = price;**

**// Format the output in a table-like structure**

**String tableRow = String.format("%-15s %-8.2f %-15s $%-10.2f $%-10.2f\n", fruitName, quantity, quantityUnit, amountPerUnit, cost);**

**billTextArea.append(tableRow);**

**// Update the total cost**

**totalCost += cost;**

**// Clear fields after adding fruit**

**quantityField.setText("");**

**priceField.setText("");**

**}**

**private void calculateTotal() {**

**DecimalFormat df = new DecimalFormat("#.##");**

**billTextArea.append("\n-----------------------------------------------------------------------\n");**

**billTextArea.append("Total Cost: $" + df.format(totalCost) + "\n");**

**}**

**private void printBill() {**

**JOptionPane.showMessageDialog(this, billTextArea.getText(), "Bill", JOptionPane.PLAIN\_MESSAGE);**

**double totalAmount = totalCost;**

**PaymentPage paymentPage = new PaymentPage(totalAmount);**

**paymentPage.setVisible(true);**

**}**

**private void deleteAll() {**

**billTextArea.setText(""); // Clear the JTextArea**

**totalCost = 0; // Reset the total cost**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(() -> new Billing().setVisible(true));**

**}**

**}**

(PaymentPage.java)

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.sql.SQLException;**

**import java.text.SimpleDateFormat;**

**import java.util.Date;**

**public class PaymentPage extends JFrame {**

**private JTextField amountField, receivedAmountField, balanceField;**

**private JComboBox<String> paymentMethodComboBox;**

**private JButton payButton, cancelButton;**

**private JLabel headerLabel, footerLabel;**

**private int billNumber;**

**public PaymentPage(double totalAmount) {**

**setTitle("Payment Page");**

**setSize(600, 400);**

**setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);**

**setLayout(null); // Using null layout to use setBounds**

**// Generate random bill number**

**billNumber = (int) (Math.random() \* 1000000);**

**// Header Panel**

**JPanel headerPanel = new JPanel(null);**

**headerPanel.setBounds(0, 0, getWidth(), 50);**

**headerPanel.setBackground(Color.GRAY); // Set header background color**

**JButton previousPageButton = new JButton("Previous Page");**

**previousPageButton.setBounds(10, 10, 120, 30);**

**headerPanel.add(previousPageButton);**

**previousPageButton.addActionListener(e -> {**

**Billing billingPage = new Billing();**

**billingPage.setVisible(true);**

**dispose(); // Close current frame**

**});**

**headerLabel = new JLabel("Payment Page", SwingConstants.CENTER);**

**headerLabel.setBounds(0, 0, getWidth(), 50);**

**headerLabel.setFont(new Font("Arial", Font.BOLD, 18));**

**headerLabel.setForeground(Color.WHITE);**

**headerPanel.add(headerLabel);**

**SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");**

**String dateTime = dateFormat.format(new Date());**

**JLabel dateTimeLabel = new JLabel(dateTime);**

**dateTimeLabel.setBounds(-30, 0, getWidth(), 50);**

**dateTimeLabel.setHorizontalAlignment(SwingConstants.RIGHT);**

**headerPanel.add(dateTimeLabel);**

**dateTimeLabel.setForeground(Color.WHITE);**

**// Input Panel**

**JPanel inputPanel = new JPanel(null);**

**inputPanel.setBounds(50, 80, getWidth() - 100, 160);**

**inputPanel.setBorder(BorderFactory.createTitledBorder("Payment Details"));**

**JLabel totalLabel = new JLabel("Total Amount:");**

**totalLabel.setBounds(20, 30, 100, 20);**

**inputPanel.add(totalLabel);**

**amountField = new JTextField();**

**amountField.setEditable(false);**

**amountField.setText(Double.toString(totalAmount));**

**amountField.setBounds(120, 30, 120, 20);**

**inputPanel.add(amountField);**

**JLabel receivedAmountLabel = new JLabel("Received Amount:");**

**receivedAmountLabel.setBounds(20, 60, 120, 20);**

**inputPanel.add(receivedAmountLabel);**

**receivedAmountField = new JTextField();**

**receivedAmountField.setBounds(150, 60, 120, 20);**

**receivedAmountField.addActionListener(e -> calculateBalance());**

**inputPanel.add(receivedAmountField);**

**JLabel paymentMethodLabel = new JLabel("Payment Method:");**

**paymentMethodLabel.setBounds(20, 90, 120, 20);**

**inputPanel.add(paymentMethodLabel);**

**String[] paymentMethods = {"Cash", "Card", "QR"};**

**paymentMethodComboBox = new JComboBox<>(paymentMethods);**

**paymentMethodComboBox.setBounds(150, 90, 120, 20);**

**paymentMethodComboBox.addActionListener(e -> calculateBalance());**

**inputPanel.add(paymentMethodComboBox);**

**JLabel balanceTextLabel = new JLabel("Balance Amount:");**

**balanceTextLabel.setBounds(20, 120, 120, 20);**

**inputPanel.add(balanceTextLabel);**

**balanceField = new JTextField();**

**balanceField.setEditable(false);**

**balanceField.setBounds(150, 120, 120, 20);**

**inputPanel.add(balanceField);**

**// Button Panel**

**JPanel buttonPanel = new JPanel(null);**

**buttonPanel.setBounds(200, 250, getWidth() - 400, 50);**

**payButton = new JButton("Pay");**

**payButton.setBounds(0, 0, 100, 30);**

**payButton.addActionListener(e -> {**

**if (pay()) {**

**JOptionPane.showMessageDialog(PaymentPage.this, "Payment successful. Please visit again.", "Payment Confirmation", JOptionPane.INFORMATION\_MESSAGE);**

**Billing billingPage = new Billing();**

**billingPage.setVisible(true);**

**dispose(); // Close the payment page**

**}**

**});**

**buttonPanel.add(payButton);**

**cancelButton = new JButton("Cancel");**

**cancelButton.setBounds(100, 0, 100, 30);**

**cancelButton.addActionListener(e -> dispose());**

**buttonPanel.add(cancelButton);**

**// Footer Panel**

**JPanel footerPanel = new JPanel(null);**

**footerPanel.setBounds(0, getHeight() - 88, getWidth(), 50);**

**footerPanel.setBackground(Color.GRAY); // Set footer background color**

**footerLabel = new JLabel(" 2024 © DK Software Pvt. Ltd.TM", SwingConstants.CENTER);**

**footerLabel.setBounds(0, 0, getWidth(), 50);**

**footerPanel.add(footerLabel);**

**footerLabel.setForeground(Color.WHITE);**

**// Add components to the frame**

**add(headerPanel);**

**add(inputPanel);**

**add(buttonPanel);**

**add(footerPanel);**

**}**

**private void calculateBalance() {**

**try {**

**double receivedAmount = Double.parseDouble(receivedAmountField.getText());**

**double totalAmount = Double.parseDouble(amountField.getText());**

**double balance = receivedAmount - totalAmount;**

**balanceField.setText(Double.toString(balance));**

**} catch (NumberFormatException e) {**

**balanceField.setText("0.0");**

**}**

**}**

**private boolean pay() {**

**try {**

**double receivedAmount = Double.parseDouble(receivedAmountField.getText());**

**double totalAmount = Double.parseDouble(amountField.getText());**

**String selectedMethod = (String) paymentMethodComboBox.getSelectedItem();**

**double balance = Double.parseDouble(balanceField.getText());**

**if (receivedAmount < totalAmount) {**

**JOptionPane.showMessageDialog(this, "Received amount is less than total amount.", "Payment Error", JOptionPane.ERROR\_MESSAGE);**

**return false;**

**} else if (receivedAmount == 0) {**

**JOptionPane.showMessageDialog(this, "Received amount cannot be zero.", "Payment Error", JOptionPane.ERROR\_MESSAGE);**

**return false;**

**} else if (selectedMethod == null || selectedMethod.isEmpty()) {**

**JOptionPane.showMessageDialog(this, "Please select a payment method.", "Payment Error", JOptionPane.ERROR\_MESSAGE);**

**return false;**

**} else {**

**storeBillingDetails(totalAmount, receivedAmount, selectedMethod, balance);**

**return true;**

**}**

**} catch (NumberFormatException ex) {**

**JOptionPane.showMessageDialog(this, "Invalid input for received amount.", "Payment Error", JOptionPane.ERROR\_MESSAGE);**

**return false;**

**}**

**}**

**private void storeBillingDetails(double totalAmount, double receivedAmount, String paymentMethod, double balance) {**

**try {**

**String DATABASE\_URL = "jdbc:ucanaccess://B://fruit//FruitDB.accdb";**

**Connection connection = DriverManager.getConnection(DATABASE\_URL);**

**String sql = "INSERT INTO BillingHistory (BillNumber, TotalAmount, PaymentDate, ReceivedAmount, PaymentMethod, BalanceAmount) VALUES (?, ?, ?, ?, ?, ?)";**

**PreparedStatement preparedStatement = connection.prepareStatement(sql);**

**preparedStatement.setInt(1, billNumber);**

**preparedStatement.setDouble(2, totalAmount);**

**preparedStatement.setString(3, new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date()));**

**preparedStatement.setDouble(4, receivedAmount);**

**preparedStatement.setString(5, paymentMethod);**

**preparedStatement.setDouble(6, balance);**

**preparedStatement.executeUpdate();**

**preparedStatement.close();**

**connection.close();**

**JOptionPane.showMessageDialog(this, "Billing details stored successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);**

**} catch (SQLException ex) {**

**JOptionPane.showMessageDialog(this, "Error storing billing details: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**ex.printStackTrace();**

**}**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(() -> new PaymentPage(100).setVisible(true));**

**}**

**}**

(BillingHis.java)

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import javax.swing.table.DefaultTableModel;**

**public class BillingHis extends JFrame {**

**private DefaultTableModel tableModel;**

**public BillingHis() {**

**setTitle("Fruit Shop Management");**

**setSize(800, 600);**

**setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);**

**setLayout(new BorderLayout());**

**// Header Label**

**JLabel headerLabel = new JLabel("Billing History", SwingConstants.CENTER);**

**headerLabel.setFont(new Font("Arial", Font.BOLD, 24));**

**headerLabel.setOpaque(true);**

**headerLabel.setBackground(Color.GRAY);**

**headerLabel.setForeground(Color.WHITE);**

**add(headerLabel, BorderLayout.NORTH);**

**// Table to display data**

**JTable table = new JTable();**

**tableModel = new DefaultTableModel();**

**tableModel.addColumn("Bill Number");**

**tableModel.addColumn("Total Amount");**

**tableModel.addColumn("Payment Date");**

**tableModel.addColumn("Received Amount");**

**tableModel.addColumn("Payment Method");**

**tableModel.addColumn("Balance Amount");**

**table.setModel(tableModel);**

**// Scroll Pane for the table**

**JScrollPane scrollPane = new JScrollPane(table);**

**add(scrollPane, BorderLayout.CENTER);**

**// Panel for buttons**

**JPanel buttonPanel = new JPanel();**

**buttonPanel.setLayout(new BorderLayout());**

**JButton clearButton = new JButton("Clear Data");**

**clearButton.addActionListener(e -> clearTableData());**

**buttonPanel.add(clearButton, BorderLayout.NORTH);**

**add(buttonPanel, BorderLayout.SOUTH);**

**// Footer Label**

**JLabel footerLabel = new JLabel("© 2024 DK Software Pvt. Ltd.TM", SwingConstants.CENTER);**

**footerLabel.setFont(new Font("Arial", Font.ITALIC, 12));**

**footerLabel.setOpaque(true);**

**footerLabel.setBackground(Color.GRAY);**

**footerLabel.setForeground(Color.WHITE);**

**buttonPanel.add(footerLabel, BorderLayout.SOUTH);**

**// Fetch data from the database and populate the table**

**fetchData();**

**}**

**private void fetchData() {**

**Connection connection = null;**

**PreparedStatement statement = null;**

**ResultSet resultSet = null;**

**try {**

**// Database connection URL**

**String DATABASE\_URL = "jdbc:ucanaccess://B://fruit//FruitDB.accdb";**

**// Establishing the database connection**

**connection = DriverManager.getConnection(DATABASE\_URL);**

**// SQL query to fetch billing details**

**String sql = "SELECT \* FROM BillingHistory";**

**// Creating a prepared statement**

**statement = connection.prepareStatement(sql);**

**// Executing the query**

**resultSet = statement.executeQuery();**

**// Adding data to the table model**

**while (resultSet.next()) {**

**int billNumber = resultSet.getInt("BillNumber");**

**double totalAmount = resultSet.getDouble("TotalAmount");**

**String paymentDate = resultSet.getString("PaymentDate");**

**double receivedAmount = resultSet.getDouble("ReceivedAmount");**

**String paymentMethod = resultSet.getString("PaymentMethod");**

**double balanceAmount = resultSet.getDouble("BalanceAmount");**

**Object[] row = {billNumber, totalAmount, paymentDate, receivedAmount, paymentMethod, balanceAmount};**

**tableModel.addRow(row);**

**}**

**} catch (SQLException ex) {**

**// Inform the user about any errors that occur during database operation**

**JOptionPane.showMessageDialog(this, "Error fetching data: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**ex.printStackTrace(); // Print the stack trace for debugging**

**} finally {**

**// Closing resources**

**try {**

**if (resultSet != null) resultSet.close();**

**if (statement != null) statement.close();**

**if (connection != null) connection.close();**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**}**

**}**

**}**

**private void clearTableData() {**

**Connection connection = null;**

**PreparedStatement statement = null;**

**try {**

**// Database connection URL**

**String DATABASE\_URL = "jdbc:ucanaccess://B://fruit//FruitDB.accdb";**

**// Establishing the database connection**

**connection = DriverManager.getConnection(DATABASE\_URL);**

**// SQL query to delete all records from BillingHistory**

**String sql = "DELETE FROM BillingHistory";**

**// Creating a prepared statement**

**statement = connection.prepareStatement(sql);**

**// Executing the update**

**statement.executeUpdate();**

**// Clear the table model**

**tableModel.setRowCount(0); // Clears all rows in the table**

**JOptionPane.showMessageDialog(this, "All billing records have been cleared from the database.", "Success", JOptionPane.INFORMATION\_MESSAGE);**

**} catch (SQLException ex) {**

**// Inform the user about any errors that occur during database operation**

**JOptionPane.showMessageDialog(this, "Error clearing data: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**ex.printStackTrace(); // Print the stack trace for debugging**

**} finally {**

**// Closing resources**

**try {**

**if (statement != null) statement.close();**

**if (connection != null) connection.close();**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**}**

**}**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(() -> new BillingHis().setVisible(true));**

**}**

(supplierPage.java)

**import javax.swing.\*;**

**import javax.swing.table.DefaultTableModel;**

**import java.awt.\*;**

**import java.awt.event.\*;**

**import java.sql.\*;**

**public class supplierPage extends JFrame {**

**private JLabel nameLabel, importItemLabel, mobileLabel, emailLabel, placeLabel, dateLabel, timeLabel;**

**private JTextField nameField, importItemField, mobileField, emailField, placeField;**

**private JButton saveButton, clearButton, deleteButton;**

**private JTable table;**

**private DefaultTableModel tableModel;**

**private Connection conn;**

**public supplierPage() {**

**setTitle("Suppliers Details");**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setSize(800, 600);**

**setLocationRelativeTo(null);**

**getContentPane().setBackground(Color.WHITE);**

**initComponents();**

**connectToDatabase();**

**}**

**private void initComponents() {**

**setLayout(null);**

**nameLabel = new JLabel("Customer Name:");**

**nameLabel.setBounds(20, 20, 120, 25);**

**nameField = new JTextField();**

**nameField.setBounds(150, 20, 200, 25);**

**importItemLabel = new JLabel("Import Item:");**

**importItemLabel.setBounds(20, 50, 120, 25);**

**importItemField = new JTextField();**

**importItemField.setBounds(150, 50, 200, 25);**

**mobileLabel = new JLabel("Mobile:");**

**mobileLabel.setBounds(20, 80, 120, 25);**

**mobileField = new JTextField();**

**mobileField.setBounds(150, 80, 200, 25);**

**emailLabel = new JLabel("Email:");**

**emailLabel.setBounds(20, 110, 120, 25);**

**emailField = new JTextField();**

**emailField.setBounds(150, 110, 200, 25);**

**placeLabel = new JLabel("Place:");**

**placeLabel.setBounds(20, 140, 120, 25);**

**placeField = new JTextField();**

**placeField.setBounds(150, 140, 200, 25);**

**saveButton = new JButton("Save");**

**saveButton.setBounds(150, 170, 80, 25);**

**saveButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**saveCustomerDetails();**

**}**

**});**

**clearButton = new JButton("Clear");**

**clearButton.setBounds(250, 170, 80, 25);**

**clearButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**clearFields();**

**}**

**});**

**deleteButton = new JButton("Delete");**

**deleteButton.setBounds(350, 170, 80, 25);**

**deleteButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**deleteCustomerDetails();**

**}**

**});**

**add(nameLabel);**

**add(nameField);**

**add(importItemLabel);**

**add(importItemField);**

**add(mobileLabel);**

**add(mobileField);**

**add(emailLabel);**

**add(emailField);**

**add(placeLabel);**

**add(placeField);**

**add(saveButton);**

**add(clearButton);**

**add(deleteButton);**

**dateLabel = new JLabel();**

**dateLabel.setBounds(600, 20, 150, 25);**

**dateLabel.setForeground(Color.WHITE);**

**add(dateLabel);**

**timeLabel = new JLabel();**

**timeLabel.setBounds(600, 50, 150, 25);**

**timeLabel.setForeground(Color.WHITE);**

**add(timeLabel);**

**tableModel = new DefaultTableModel();**

**tableModel.setColumnIdentifiers(new String[]{"Name", "Import Item", "Mobile", "Email", "Place"});**

**table = new JTable(tableModel);**

**JScrollPane scrollPane = new JScrollPane(table);**

**scrollPane.setBounds(20, 200, 740, 150);**

**add(scrollPane);**

**JPanel footerPanel = new JPanel(new BorderLayout());**

**footerPanel.setBackground(Color.GRAY);**

**footerPanel.setBounds(0, getHeight() - 88, 800, 50);**

**add(footerPanel);**

**JLabel copyrightLabel = new JLabel("Copyright © 2024 DK Software Pvt. Ltd.TM", SwingConstants.CENTER);**

**copyrightLabel.setForeground(Color.WHITE);**

**footerPanel.add(copyrightLabel, BorderLayout.CENTER);**

**}**

**private void connectToDatabase() {**

**try {**

**Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");**

**String url = "jdbc:ucanaccess://B://fruit//FruitDB.accdb";**

**conn = DriverManager.getConnection(url);**

**fetchAndDisplayDetails();**

**} catch (Exception ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to connect to the database.");**

**System.exit(1);**

**}**

**}**

**private void fetchAndDisplayDetails() {**

**tableModel.setRowCount(0); // Clear existing rows in the table model**

**try {**

**Statement statement = conn.createStatement();**

**ResultSet resultSet = statement.executeQuery("SELECT \* FROM Dealer");**

**while (resultSet.next()) {**

**String name = resultSet.getString("Name");**

**String importItem = resultSet.getString("ImportItem");**

**String mobile = resultSet.getString("Mobile");**

**String email = resultSet.getString("Email");**

**String place = resultSet.getString("Place");**

**tableModel.addRow(new Object[]{name, importItem, mobile, email, place});**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to fetch customer details from the database.");**

**}**

**}**

**private void saveCustomerDetails() {**

**saveButton.setEnabled(false); // Disable the save button**

**String name = nameField.getText();**

**String importItem = importItemField.getText();**

**String mobile = mobileField.getText();**

**String email = emailField.getText();**

**String place = placeField.getText();**

**if (isDuplicateRecord(mobile, email)) {**

**JOptionPane.showMessageDialog(this, "Customer details already exist.");**

**saveButton.setEnabled(true); // Re-enable the save button**

**return;**

**}**

**try {**

**PreparedStatement ps = conn.prepareStatement("INSERT INTO Dealer (Name, ImportItem, Mobile, Email, Place) VALUES (?, ?, ?, ?, ?)");**

**ps.setString(1, name);**

**ps.setString(2, importItem);**

**ps.setString(3, mobile);**

**ps.setString(4, email);**

**ps.setString(5, place);**

**ps.executeUpdate();**

**JOptionPane.showMessageDialog(this, "Customer details saved successfully.");**

**clearFields();**

**fetchAndDisplayDetails();**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to save customer details: " + ex.getMessage());**

**} finally {**

**saveButton.setEnabled(true); // Re-enable the save button**

**}**

**}**

**private void deleteCustomerDetails() {**

**int selectedRow = table.getSelectedRow();**

**if (selectedRow == -1) {**

**JOptionPane.showMessageDialog(this, "Please select a record to delete.");**

**return;**

**}**

**String mobile = (String) tableModel.getValueAt(selectedRow, 2); // Assuming mobile is unique**

**try {**

**PreparedStatement ps = conn.prepareStatement("DELETE FROM Dealer WHERE Mobile = ?");**

**ps.setString(1, mobile);**

**int result = ps.executeUpdate();**

**if (result > 0) {**

**JOptionPane.showMessageDialog(this, "Customer details deleted successfully.");**

**tableModel.removeRow(selectedRow); // Remove row from table model**

**} else {**

**JOptionPane.showMessageDialog(this, "Failed to delete customer details.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to delete customer details: " + ex.getMessage());**

**}**

**}**

**private boolean isDuplicateRecord(String mobile, String email) {**

**try {**

**PreparedStatement ps = conn.prepareStatement("SELECT COUNT(\*) FROM Dealer WHERE Mobile = ? OR Email = ?");**

**ps.setString(1, mobile);**

**ps.setString(2, email);**

**ResultSet rs = ps.executeQuery();**

**if (rs.next()) {**

**return rs.getInt(1) > 0;**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to check for duplicate records: " + ex.getMessage());**

**}**

**return false;**

**}**

**private void clearFields() {**

**nameField.setText("");**

**importItemField.setText("");**

**mobileField.setText("");**

**emailField.setText("");**

**placeField.setText("");**

**}**

**public static void main(String[] args) {**

**supplierPage frame = new supplierPage();**

**frame.setVisible(true);**

**}**

**}**

(Product.java)

**import javax.swing.\*;**

**import javax.swing.table.DefaultTableModel;**

**import java.awt.\*;**

**import java.awt.event.\*;**

**import java.sql.\*;**

**import java.text.ParseException;**

**import java.text.SimpleDateFormat;**

**import java.util.Date;**

**public class Prodect extends JFrame {**

**private JLabel itemNameLabel, supplierNameLabel, doReceivingLabel, doExpiryLabel, origPriceLabel, sellingPriceLabel, qtyLabel, totalAmountLabel;**

**private JTextField itemNameField, supplierNameField, doReceivingField, doExpiryField, origPriceField, sellingPriceField, qtyField;**

**private JLabel totalAmountField;**

**private JButton saveButton, clearButton, deleteButton;**

**private JTable table;**

**private DefaultTableModel tableModel;**

**private Connection conn;**

**public Prodect() {**

**setTitle("Prodect Details");**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setSize(817, 620);**

**setLocationRelativeTo(null);**

**initComponents();**

**connectToDatabase();**

**}**

**private void initComponents() {**

**setLayout(null);**

**// Header Panel**

**JPanel headerPanel = new JPanel(new BorderLayout());**

**headerPanel.setBackground(Color.GRAY);**

**headerPanel.setBounds(0, 0, 800, 50);**

**// Title Label**

**JLabel titleLabel = new JLabel("Product Details", JLabel.CENTER);**

**titleLabel.setForeground(Color.WHITE);**

**titleLabel.setFont(new Font("CAMBRIA", Font.BOLD, 16));**

**headerPanel.add(titleLabel, BorderLayout.CENTER);**

**// Date-Time Label**

**JLabel dateTimeLabel = new JLabel();**

**updateDateTimeLabel(dateTimeLabel);**

**headerPanel.add(dateTimeLabel, BorderLayout.EAST);**

**dateTimeLabel.setForeground(Color.WHITE);**

**add(headerPanel);**

**// Footer Panel**

**JPanel footerPanel = new JPanel(null);**

**footerPanel.setBackground(Color.GRAY);**

**footerPanel.setBounds(0, 540, 800, 50);**

**JLabel titleLabel1 = new JLabel("Copyright © 2024 DK Software Pvt. Ltd.TM", JLabel.CENTER);**

**footerPanel.setForeground(Color.WHITE);**

**footerPanel.setFont(new Font("Arial", Font.BOLD, 16));**

**titleLabel1.setBounds(200, 10, 400, 30);**

**footerPanel.add(titleLabel1, BorderLayout.CENTER);**

**add(footerPanel);**

**itemNameLabel = new JLabel("Item Name:");**

**itemNameLabel.setBounds(20, 60, 120, 25);**

**itemNameField = new JTextField();**

**itemNameField.setBounds(150, 60, 200, 25);**

**supplierNameLabel = new JLabel("Supplier Name:");**

**supplierNameLabel.setBounds(20, 90, 120, 25);**

**supplierNameField = new JTextField();**

**supplierNameField.setBounds(150, 90, 200, 25);**

**doReceivingLabel = new JLabel("D.O. Receiving:");**

**doReceivingLabel.setBounds(20, 120, 120, 25);**

**doReceivingField = new JTextField();**

**doReceivingField.setBounds(150, 120, 200, 25);**

**doExpiryLabel = new JLabel("D.O. Expiry:");**

**doExpiryLabel.setBounds(20, 150, 120, 25);**

**doExpiryField = new JTextField();**

**doExpiryField.setBounds(150, 150, 200, 25);**

**origPriceLabel = new JLabel("Orig Price:");**

**origPriceLabel.setBounds(20, 180, 120, 25);**

**origPriceField = new JTextField();**

**origPriceField.setBounds(150, 180, 200, 25);**

**sellingPriceLabel = new JLabel("Selling Price per Kg/Gms:");**

**sellingPriceLabel.setBounds(20, 210, 170, 25);**

**sellingPriceField = new JTextField();**

**sellingPriceField.setBounds(200, 210, 150, 25);**

**sellingPriceField.addKeyListener(new KeyAdapter() {**

**public void keyReleased(KeyEvent e) {**

**updateTotalAmount();**

**}**

**});**

**qtyLabel = new JLabel("Qty (Kg/Gms):");**

**qtyLabel.setBounds(20, 240, 120, 25);**

**qtyField = new JTextField();**

**qtyField.setBounds(150, 240, 200, 25);**

**qtyField.addKeyListener(new KeyAdapter() {**

**public void keyReleased(KeyEvent e) {**

**updateTotalAmount();**

**}**

**});**

**totalAmountLabel = new JLabel("Total Amount:");**

**totalAmountLabel.setBounds(20, 270, 120, 25);**

**totalAmountField = new JLabel();**

**totalAmountField.setBounds(150, 270, 200, 25);**

**totalAmountField.setOpaque(true);**

**totalAmountField.setBackground(Color.LIGHT\_GRAY);**

**saveButton = new JButton("Save");**

**saveButton.setBounds(150, 300, 80, 25);**

**saveButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**saveInventoryDetails();**

**}**

**});**

**clearButton = new JButton("Clear");**

**clearButton.setBounds(250, 300, 80, 25);**

**clearButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**clearFields();**

**}**

**});**

**deleteButton = new JButton("Delete");**

**deleteButton.setBounds(350, 300, 80, 25);**

**deleteButton.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**deleteInventoryDetails();**

**}**

**});**

**add(itemNameLabel);**

**add(itemNameField);**

**add(supplierNameLabel);**

**add(supplierNameField);**

**add(doReceivingLabel);**

**add(doReceivingField);**

**add(doExpiryLabel);**

**add(doExpiryField);**

**add(origPriceLabel);**

**add(origPriceField);**

**add(sellingPriceLabel);**

**add(sellingPriceField);**

**add(qtyLabel);**

**add(qtyField);**

**add(totalAmountLabel);**

**add(totalAmountField);**

**add(saveButton);**

**add(clearButton);**

**add(deleteButton);**

**tableModel = new DefaultTableModel();**

**tableModel.setColumnIdentifiers(new String[]{"Item Name", "Supplier Name", "D.O. Receiving", "D.O. Expiry", "Orig Price", "Selling Price", "Qty", "Total Amount"});**

**table = new JTable(tableModel);**

**JScrollPane scrollPane = new JScrollPane(table);**

**scrollPane.setBounds(20, 340, 740, 200);**

**add(scrollPane);**

**}**

**private void connectToDatabase() {**

**try {**

**Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");**

**String url = "jdbc:ucanaccess://B://fruit//FruitDB.accdb";**

**conn = DriverManager.getConnection(url);**

**fetchAndDisplayDetails();**

**} catch (Exception ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to connect to the database.");**

**System.exit(1);**

**}**

**}**

**private void fetchAndDisplayDetails() {**

**tableModel.setRowCount(0); // Clear existing rows in the table model**

**try (Statement statement = conn.createStatement(); ResultSet resultSet = statement.executeQuery("SELECT \* FROM Product")) {**

**while (resultSet.next()) {**

**String itemName = resultSet.getString("ItemName");**

**String supplierName = resultSet.getString("SupplierName");**

**Date doReceiving = resultSet.getDate("DOReceiving");**

**Date doExpiry = resultSet.getDate("DOExpiry");**

**double origPrice = resultSet.getDouble("OrigPrice");**

**double sellingPrice = resultSet.getDouble("SellingPrice");**

**int qty = resultSet.getInt("Qty");**

**double totalAmount = resultSet.getDouble("TotalAmount");**

**tableModel.addRow(new Object[]{itemName, supplierName, doReceiving, doExpiry, origPrice, sellingPrice, qty, totalAmount});**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to fetch Product details from the database.");**

**}**

**}**

**private void saveInventoryDetails() {**

**saveButton.setEnabled(false); // Disable the save button**

**String itemName = itemNameField.getText();**

**String supplierName = supplierNameField.getText();**

**String doReceiving = doReceivingField.getText();**

**String doExpiry = doExpiryField.getText();**

**String origPrice = origPriceField.getText();**

**String sellingPrice = sellingPriceField.getText();**

**String qty = qtyField.getText();**

**double totalAmount;**

**try {**

**double sellingPriceDouble = Double.parseDouble(sellingPrice);**

**double qtyDouble = Double.parseDouble(qty);**

**totalAmount = sellingPriceDouble \* qtyDouble;**

**} catch (NumberFormatException e) {**

**JOptionPane.showMessageDialog(this, "Please enter valid numbers for selling price and quantity.");**

**saveButton.setEnabled(true);**

**return;**

**}**

**if (isDuplicateRecord(itemName, supplierName)) {**

**JOptionPane.showMessageDialog(this, "Product details already exist.");**

**saveButton.setEnabled(true); // Re-enable the save button**

**return;**

**}**

**try {**

**PreparedStatement ps = conn.prepareStatement("INSERT INTO Product (ItemName, SupplierName, DOReceiving, DOExpiry, OrigPrice, SellingPrice, Qty, TotalAmount, SellAmt) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");**

**ps.setString(1, itemName);**

**ps.setString(2, supplierName);**

**ps.setDate(3, parseDate(doReceiving));**

**ps.setDate(4, parseDate(doExpiry));**

**ps.setDouble(5, Double.parseDouble(origPrice));**

**ps.setDouble(6, Double.parseDouble(sellingPrice));**

**ps.setInt(7, Integer.parseInt(qty));**

**ps.setDouble(8, totalAmount);**

**ps.setDouble(9, totalAmount); // Assuming SellAmt is the same as Total Amount on insertion**

**ps.executeUpdate();**

**JOptionPane.showMessageDialog(this, "Product details saved successfully.");**

**clearFields();**

**fetchAndDisplayDetails();**

**} catch (SQLException | ParseException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to save Product details: " + ex.getMessage());**

**} finally {**

**saveButton.setEnabled(true); // Re-enable the save button**

**}**

**}**

**private void deleteInventoryDetails() {**

**int selectedRow = table.getSelectedRow();**

**if (selectedRow == -1) {**

**JOptionPane.showMessageDialog(this, "Please select a record to delete.");**

**return;**

**}**

**String itemName = (String) tableModel.getValueAt(selectedRow, 0); // Assuming item name is unique**

**try {**

**PreparedStatement ps = conn.prepareStatement("DELETE FROM Product WHERE ItemName = ?");**

**ps.setString(1, itemName);**

**int result = ps.executeUpdate();**

**if (result > 0) {**

**JOptionPane.showMessageDialog(this, "Product details deleted successfully.");**

**tableModel.removeRow(selectedRow); // Remove row from table model**

**} else {**

**JOptionPane.showMessageDialog(this, "Failed to delete Product details.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to delete Product details: " + ex.getMessage());**

**}**

**}**

**private boolean isDuplicateRecord(String itemName, String supplierName) {**

**try {**

**PreparedStatement ps = conn.prepareStatement("SELECT COUNT(\*) FROM Product WHERE ItemName = ? AND SupplierName = ?");**

**ps.setString(1, itemName);**

**ps.setString(2, supplierName);**

**ResultSet rs = ps.executeQuery();**

**if (rs.next()) {**

**return rs.getInt(1) > 0;**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to check for duplicate records: " + ex.getMessage());**

**}**

**return false;**

**}**

**private void clearFields() {**

**itemNameField.setText("");**

**supplierNameField.setText("");**

**doReceivingField.setText("");**

**doExpiryField.setText("");**

**origPriceField.setText("");**

**sellingPriceField.setText("");**

**qtyField.setText("");**

**totalAmountField.setText("");**

**}**

**private java.sql.Date parseDate(String date) throws ParseException {**

**SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");**

**java.util.Date utilDate = sdf.parse(date);**

**return new java.sql.Date(utilDate.getTime());**

**}**

**private void updateDateTimeLabel(JLabel label) {**

**SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss a");**

**Timer timer = new Timer(1000, new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**label.setText(sdf.format(new Date()));**

**}**

**});**

**timer.start();**

**}**

**private void updateTotalAmount() {**

**try {**

**double sellingPrice = Double.parseDouble(sellingPriceField.getText());**

**double qty = Double.parseDouble(qtyField.getText());**

**double totalAmount = sellingPrice \* qty;**

**totalAmountField.setText(String.valueOf(totalAmount));**

**} catch (NumberFormatException e) {**

**totalAmountField.setText("");**

**}**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(() -> {**

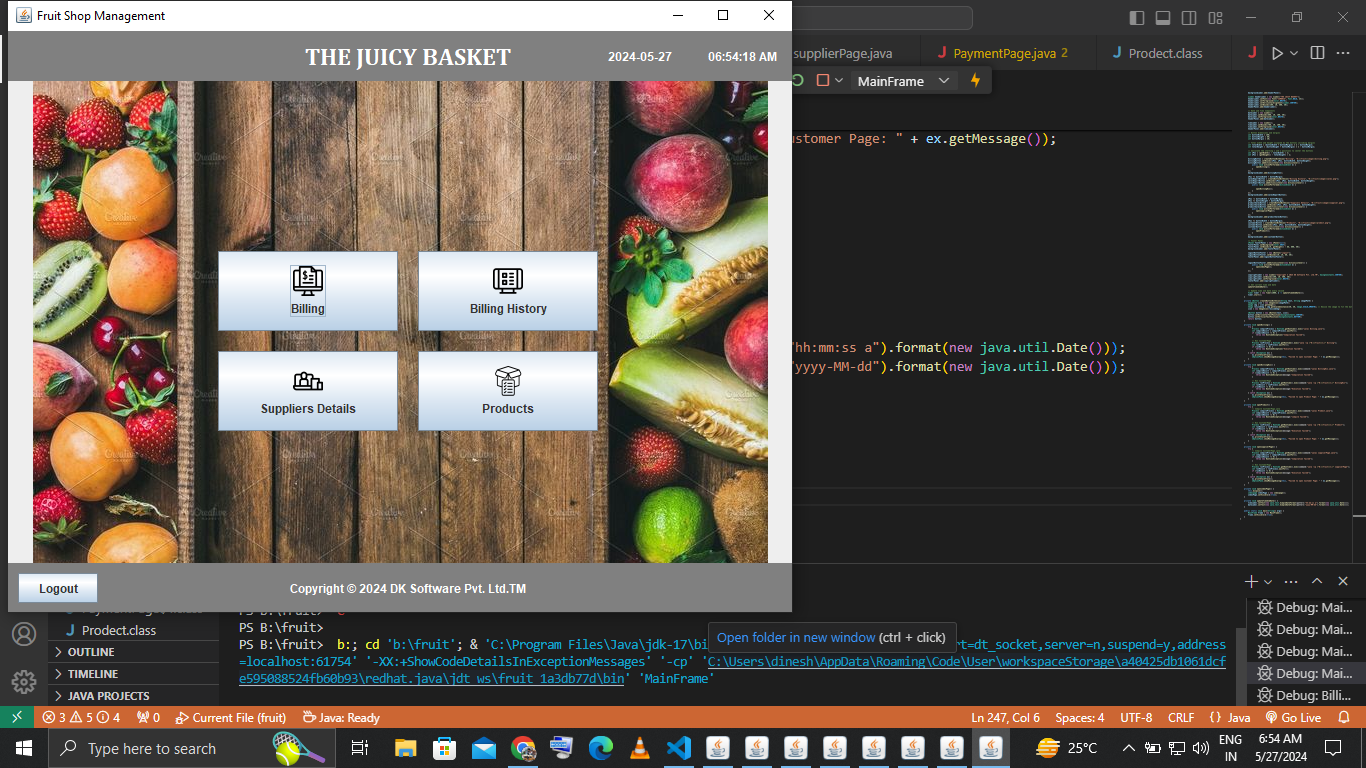
**new Prodect().setVisible(true);**

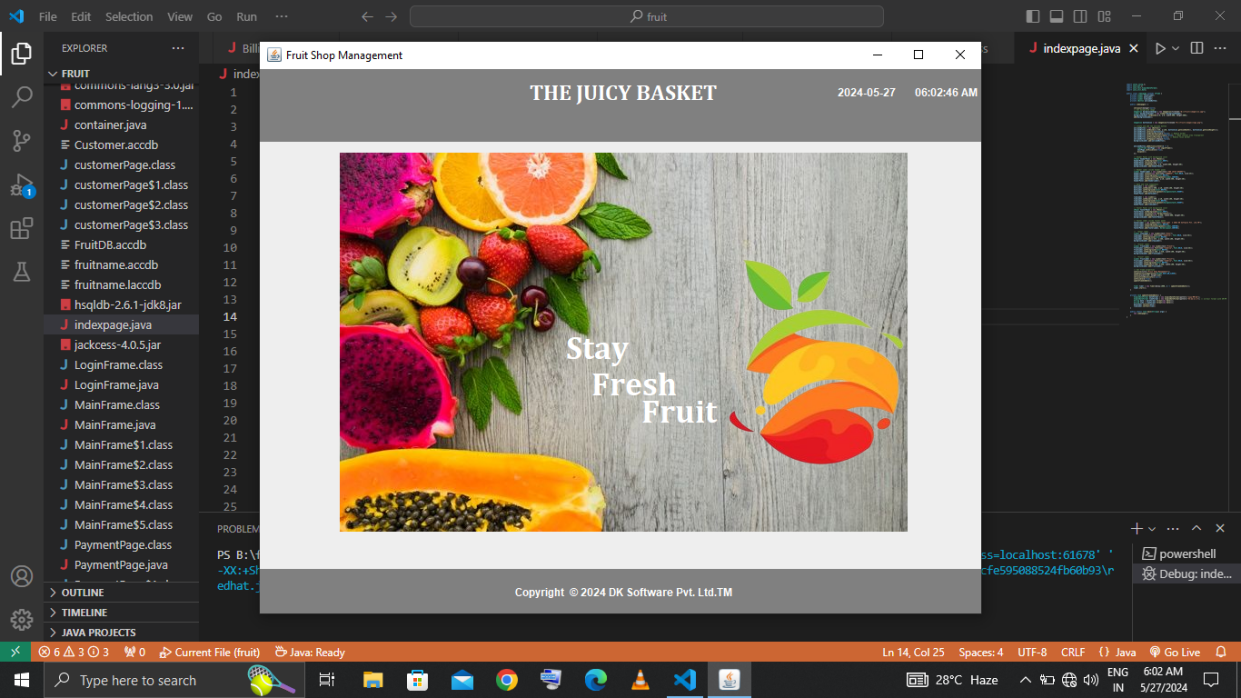
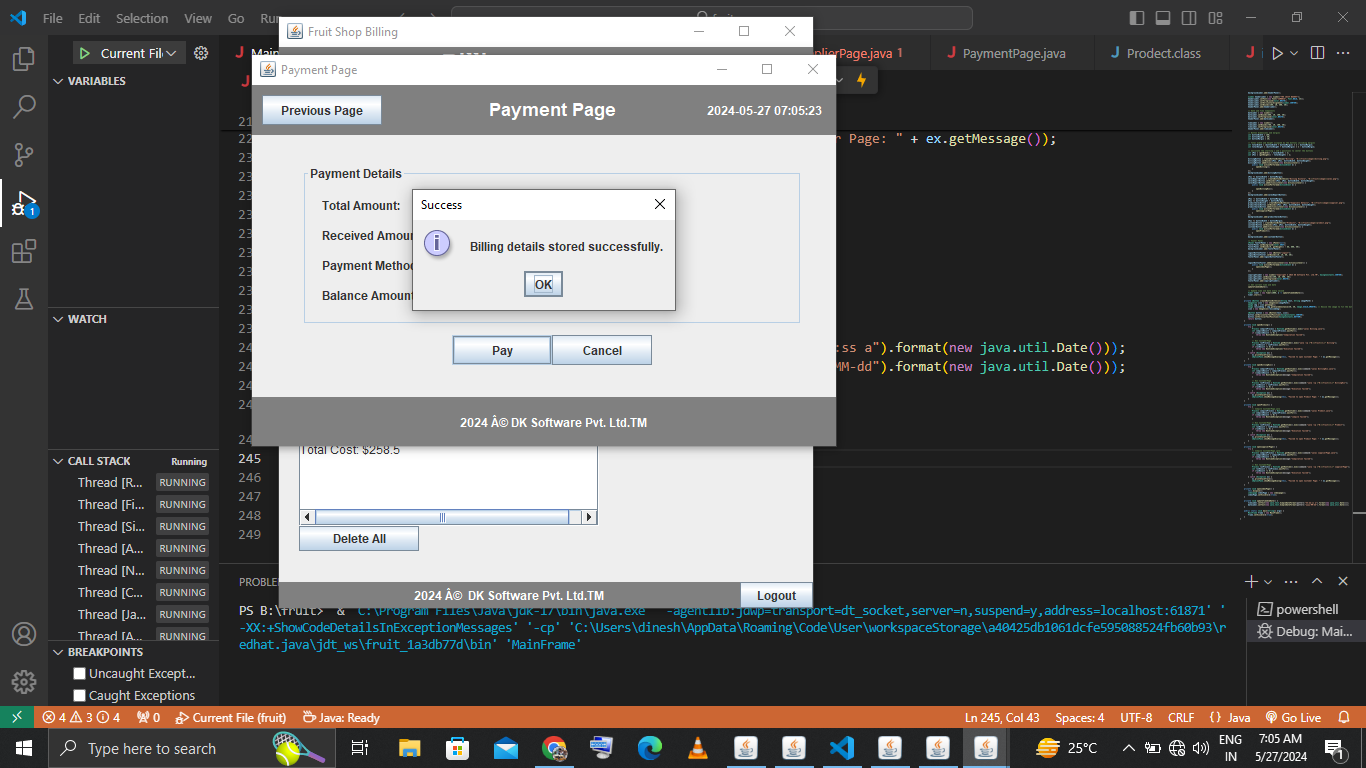
**});**

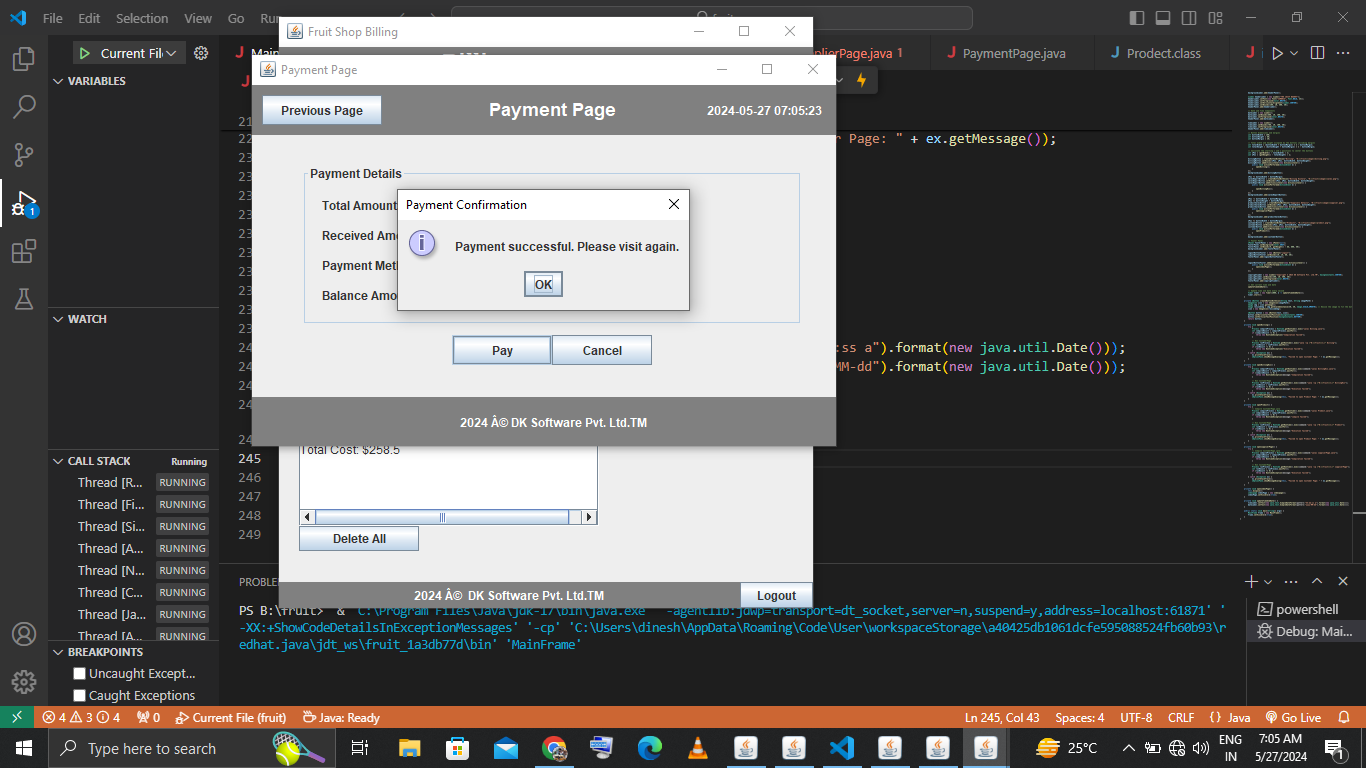
**}**

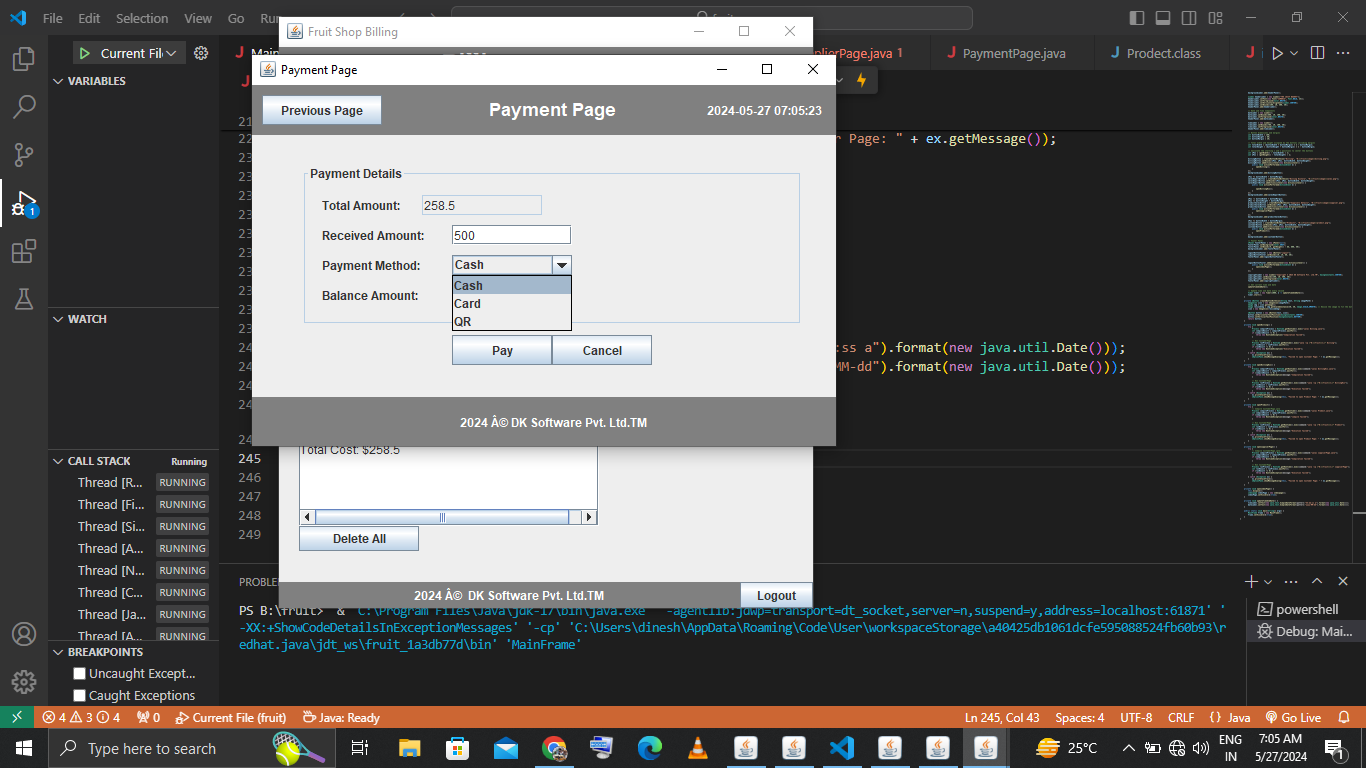
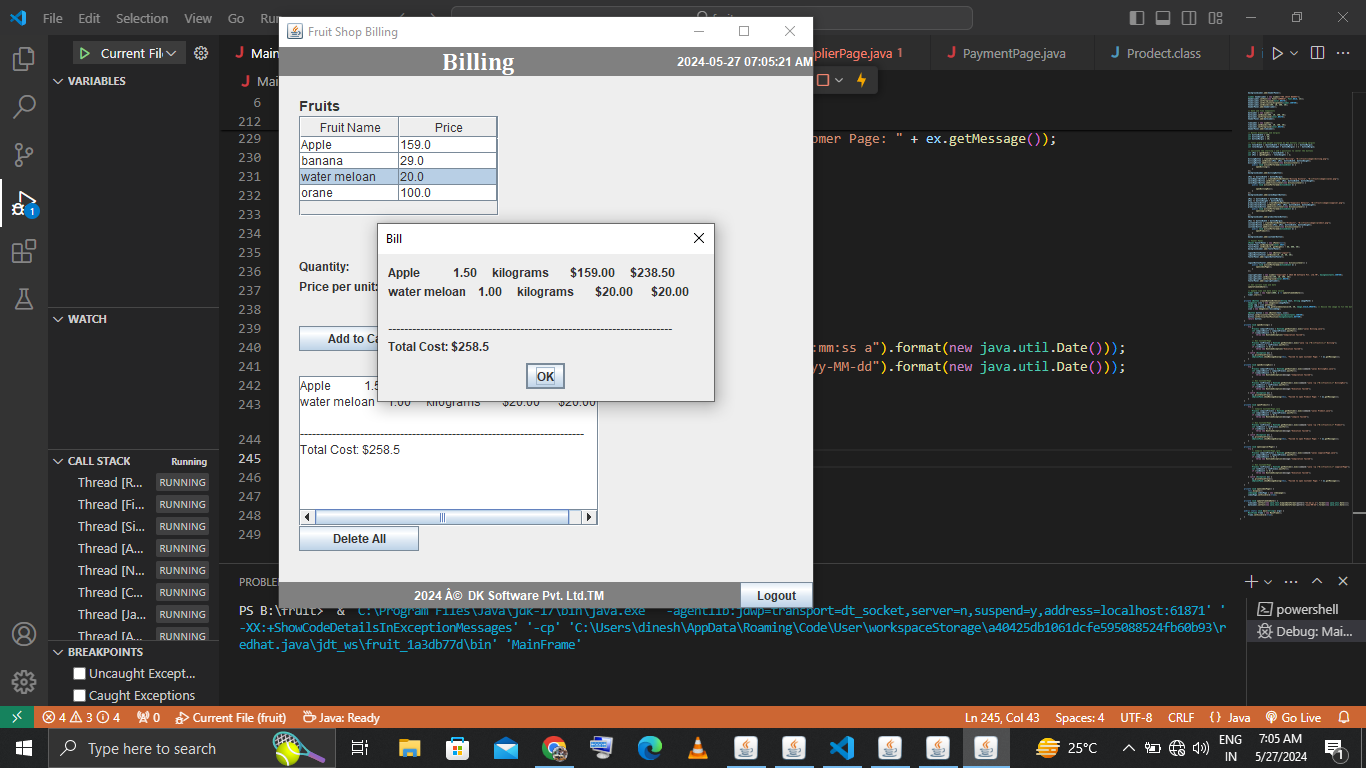
**}**

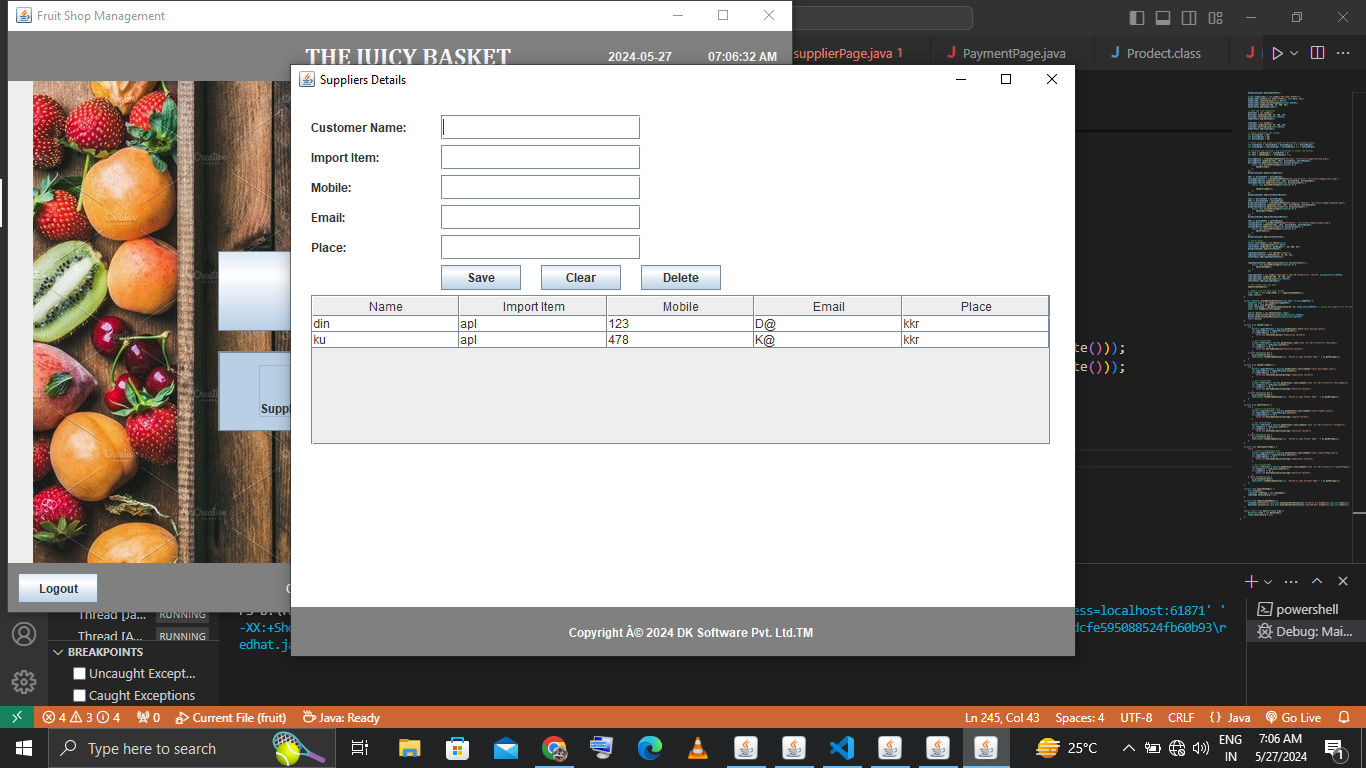
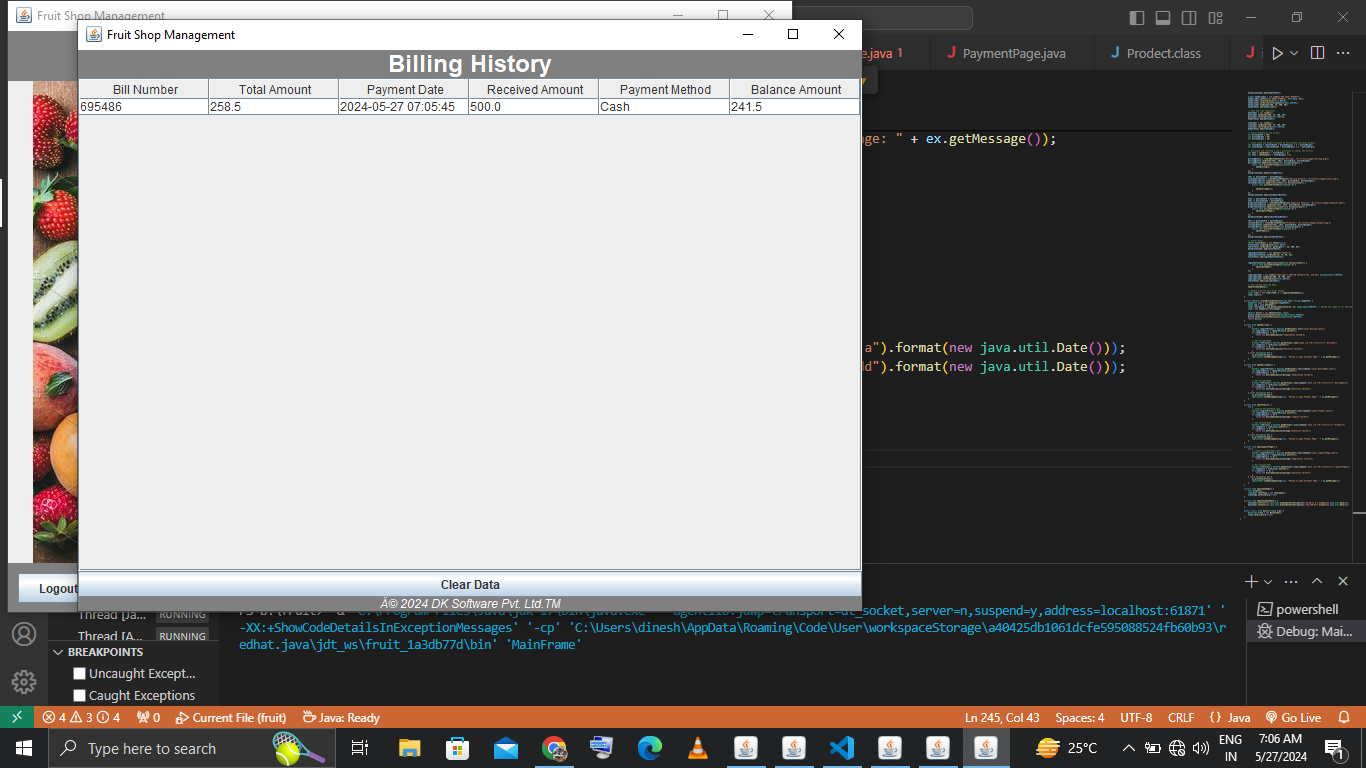
**5.SCREEN SHORT**

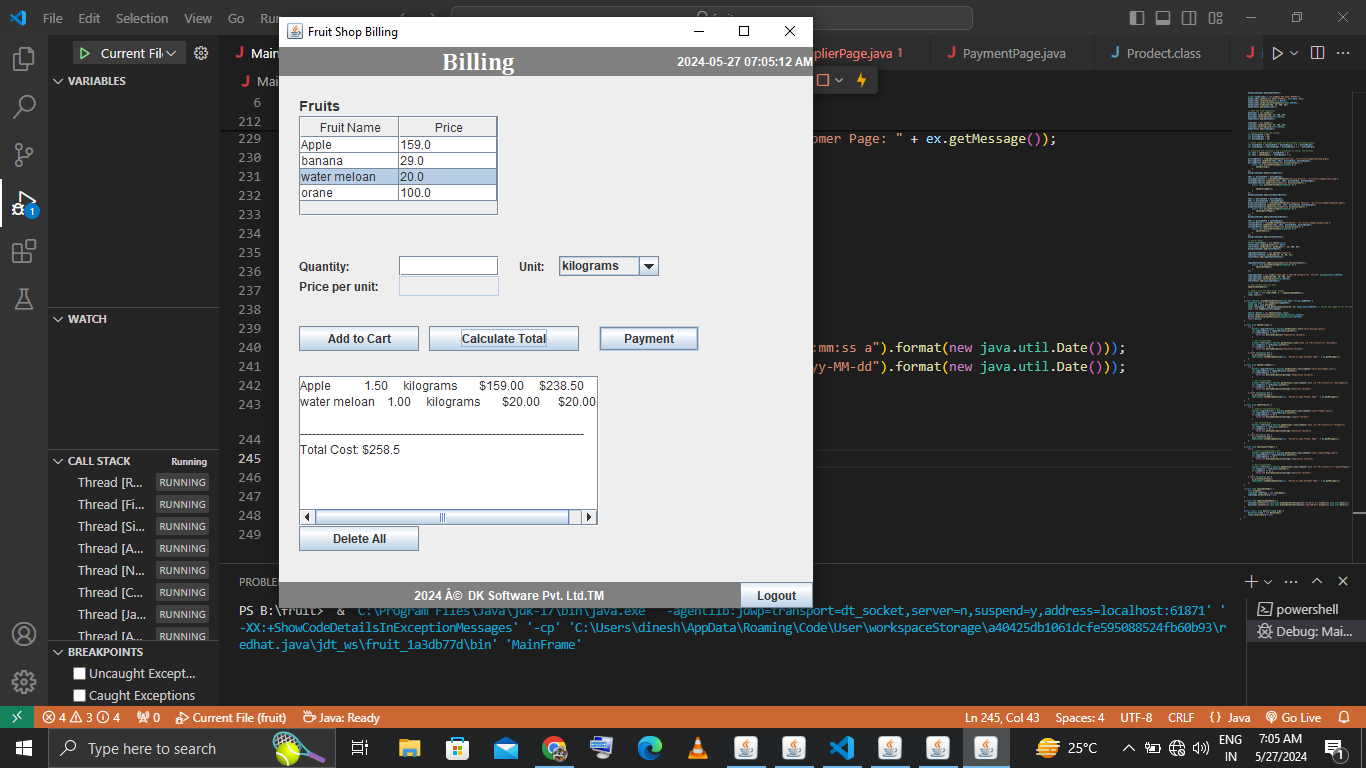
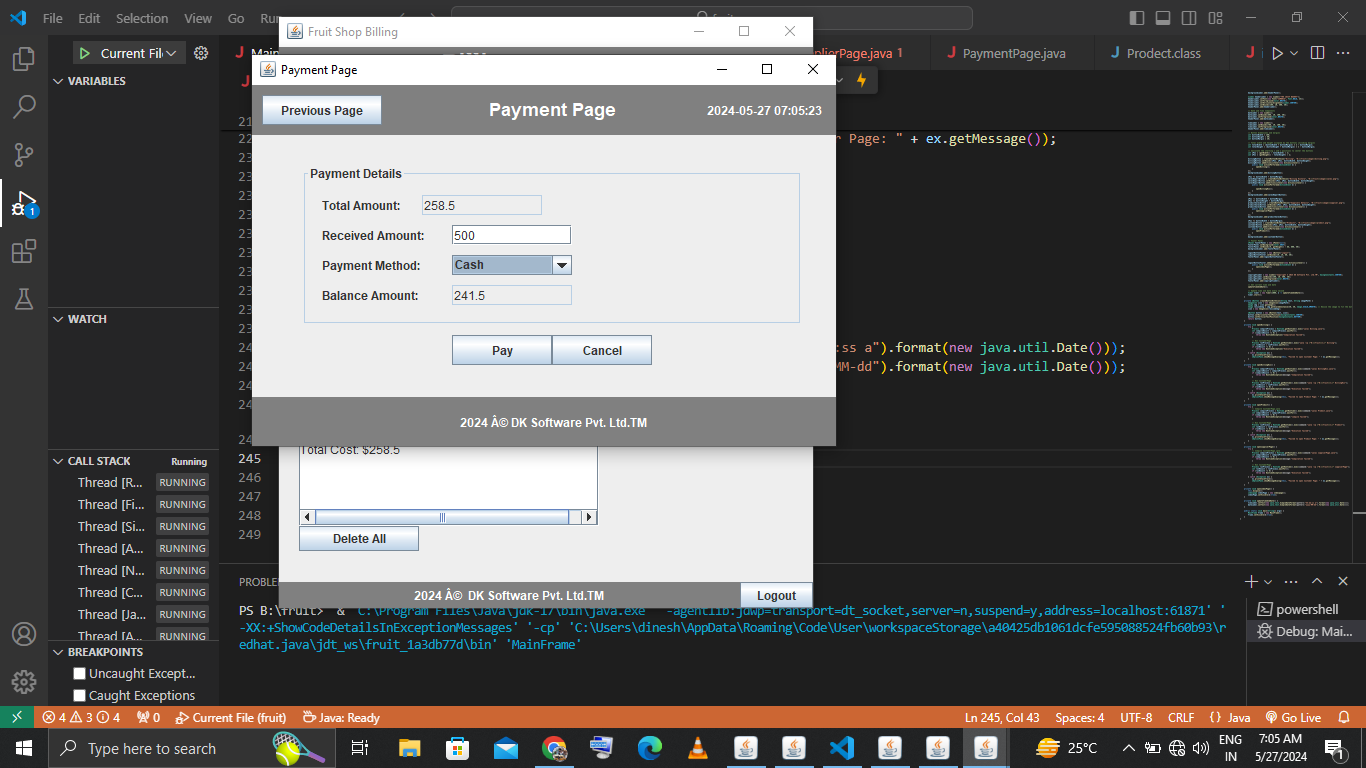
****

****

****

****

****

****