DineDash

Quick, efficient ordering

Project Motivation:

The primary motivation behind this project is to develop an interactive and user-friendly Food Ordering System that streamlines the ordering process for customers. With the increasing trend of online food ordering, creating a system that enhances user experience while being efficient and visually appealing is essential. The project aims to replicate a real-world scenario of a food ordering system, providing a platform for practical learning and implementation of Java Swing components.

Project Definition:

This project is a desktop application named **"Foodie's Hub - Premium Ordering System"** built using Java Swing. It allows users to browse through a menu of food items categorized as Food, Drinks, and Desserts. Users can select items, adjust quantities, and add them to their cart. The application calculates the total price and offers two payment methods: Cash on Counter and Online Payment via QR code. Additionally, it features a feedback system where users can rate the service and provide textual feedback.

Design and Idea:

- **Idea Origin:** The concept originated from the need to create a hands-on project demonstrating practical Java Swing skills. The idea was to simulate a real-world food ordering system with interactive features like order summary, payment methods, and feedback mechanisms.

- **Design Approach:** The interface is designed using Java Swing components with a blend of modern UI practices like gradient backgrounds, tabbed menus, grid-based layouts, and structured panels. The design focuses on simplicity, ensuring a smooth and user-friendly experience.

Detailed Explanation of Java Swing Components

1. JFrame

- **Purpose:** Acts as the main window for the application, where all UI components are added.
- **Usage in Project:**
    - The **FoodOrderingSystem** class extends `JFrame`, making it the core container.
    - Used to set window properties like **title**, **size**, and **layout**.
- **Why Used:** To create a top-level window where all components like panels, buttons, and text fields are placed.

---

2. JPanel

- **Purpose:** A container that organizes components. It can be nested to create complex layouts.
- **Usage in Project:**
    - Panels are used to create the header, user details, menu sections, order panel, and footer.
    - It helps in organizing the components logically.
- **Why Used:** Provides a structured layout and a way to group related components visually.

---

3. JLabel

- **Purpose:** Displays text, images, or both. Used for non-editable display.
- **Usage in Project:**
    - Displays titles like **"Foodie's Hub - Premium Ordering System"**.

    - Used for item names, prices, and placeholders when images fail to load.
- **Why Used:** Provides information labels, enhancing the readability and aesthetics of the interface.

---

4. JTextField

- **Purpose:** Allows users to input a single line of text.
- **Usage in Project:**
    - To input customer details like **name** and **phone number**.
- **Why Used:** For user input, crucial for order tracking and confirmation.

---

5. JTextArea

- **Purpose:** Used for multi-line text display and input.
- **Usage in Project:**
    - Displays the **order summary**.
    - The text area is non-editable, ensuring it acts as a display area.
- **Why Used:** To show dynamic information about the order, including item details and the total amount.

---

6. JTabbedPane

- **Purpose:** Provides a tabbed navigation mechanism to switch between different views.
- **Usage in Project:**
    - Categorizes menu items into **Food**, **Drinks**, and **Desserts**.
- **Why Used:** Makes navigation user-friendly, improving organization and accessibility.

---

7. JScrollPane

- **Purpose:** Adds scrolling capability to components that might exceed visible space.
- **Usage in Project:**
    - Used within tabbed panes to handle long lists of items.
    - Wraps the order summary text area.
- **Why Used:** Ensures usability when the content overflows, especially in the menu panel.

---

8. JButton

- **Purpose:** A clickable button that triggers actions.
- **Usage in Project:**
    - Adds items to the cart (`"Add to Cart"`).
    - Place orders (`"Place Order"`).
    - Choose payment methods (`"Pay in Cash"` and `"Pay Online"`).
- **Why Used:** For interaction, making the application dynamic and responsive.

---

9. JSpinner

- **Purpose:** Allows selecting a numeric value from a range.
- **Usage in Project:**
    - To adjust the quantity of each food item.
- **Why Used:** Ensures controlled and valid input for item quantity.

---

10. JOptionPane

- **Purpose:** Provides simple pop-up dialog boxes for messages, confirmations, and input.
- **Usage in Project:**
    - Alerts for empty cart.
    - Prompts for missing customer details.
    - **Why Used:** Enhances user experience by providing feedback and handling errors.

11. JDialog

- **Purpose:** A pop-up window to capture user input or show notifications.
- **Usage in Project:**
    - Used for payment options and order confirmation.
    - The feedback and rating system uses dialogs.
    -
- **Why Used:** For focused, modal interactions, preventing user actions outside the dialog until it's closed.

---

12. ImageIO and BufferedImage

- **Purpose:** Loads and manipulates images for the application.
- **Usage in Project:**
    - Loads images of menu items.
    - Displays a QR code for online payments.
- **Why Used: Enhances the visual aspect, making the UI appealing.**

---

13. BorderFactory

- **Purpose: Creates standard and customized borders.**
- **Usage in Project:**
    - **Adds decorative borders to panels for a structured and aesthetic look.**
- **Why Used: To organize UI elements with clear boundaries.**

---

14. FlowLayout, BorderLayout, GridLayout, BoxLayout

- **Purpose: Manage component arrangement and layout.**
- **Usage in Project:**
    - `BorderLayout` **for the main window.**
    - `FlowLayout` **for customer details and quantity controls.**
    - `GridLayout` **for arranging menu items.**
    - `BoxLayout` **for feedback and rating dialogs.**
- **Why Used: Ensures a neat, organized, and user-friendly interface.**

---

15. SwingUtilities

- **Purpose: Facilitates safe, concurrent updates to the GUI.**
- **Usage in Project:**
    - **Initiates the program with the event dispatch thread.**

Why Used: Ensures the application runs smoothly without UI freezing issues.

Detailed Code Block Analysis

1. Main Application Window Initialization

```
public class FoodOrderingSystem extends JFrame {
    private JTextField nameField, phoneField;
    private JTextArea orderSummary;
    private ArrayList<FoodItem> cart = new ArrayList<>();
    private double total = 0.0;
    private int userRating = 0;

    public FoodOrderingSystem() {
        setTitle("Foodie's Hub - Premium Ordering System");
        setSize(1400, 900);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout(15, 15));
        getContentPane().setBackground(new Color(245, 245, 245));

        add(createHeader(), BorderLayout.NORTH);

        JPanel mainContent = new JPanel(new BorderLayout());
        mainContent.add(createUserDetailsPanel(), BorderLayout.NORTH);
        mainContent.add(createMenuPanel(), BorderLayout.CENTER);

        add(mainContent, BorderLayout.CENTER);
        add(createOrderPanel(), BorderLayout.EAST);
        add(createFooter(), BorderLayout.SOUTH);

        setVisible(true);
    }
```

Purpose and Breakdown

- **Window Configuration**
  - Sets a large window size (1400x900 pixels)
  - Defines application title
  - Ensures application closes when the window is closed
- **Layout Management**
  - Uses `BorderLayout` with 15-pixel gaps between components
  - Divides the window into five main regions:
    1. North: Header
    2. North of main content: User Details
    3. Center of main content: Menu
    4. East: Order Panel
    5. South: Footer
- **Visual Styling**

- Sets a light gray background color
- Makes the window visible

2. Header Panel Creation

```java
private JPanel createHeader() {
    JPanel headerPanel = new JPanel() {
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            Graphics2D g2d = (Graphics2D) g;
            Color gradientStart = new Color(255, 165, 0);
            Color gradientEnd = new Color(255, 140, 0);
            GradientPaint gradientPaint = new GradientPaint(0, 0, gradientStart, getWidth(), getHeight(), gradientEnd);
            g2d.setPaint(gradientPaint);
            g2d.fillRect(0, 0, getWidth(), getHeight());
        }
    };
    headerPanel.setLayout(new BorderLayout());
    headerPanel.setPreferredSize(new Dimension(getWidth(), 100));

    JLabel titleLabel = new JLabel("Foodie's Hub - Premium Ordering System", JLabel.CENTER);
    titleLabel.setFont(new Font("Verdana", Font.BOLD, 28));
    titleLabel.setForeground(Color.WHITE);

    headerPanel.add(titleLabel, BorderLayout.CENTER);

    return headerPanel;
}
```

Purpose and Breakdown

- **Custom Gradient Background**
  - Overrides `paintComponent()` to create an orange gradient
  - Uses `GradientPaint` for smooth color transition
- **Title Label**
  - Centers the application title
  - Uses Verdana bold font, size 28
  - Sets text color to white for contrast
- **Panel Configuration**
  - Sets a fixed height of 100 pixels
  - Uses `BorderLayout` to center the title

3. User Details Panel

```java
private JPanel createUserDetailsPanel() {
    JPanel userPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 20, 10));
    userPanel.setBackground(Color.WHITE);
    userPanel.setBorder(BorderFactory.createTitledBorder("Customer Details"));

    JLabel nameLabel = new JLabel("Name:");
    nameField = new JTextField(20);

    JLabel phoneLabel = new JLabel("Phone:");
    phoneField = new JTextField(15);

    userPanel.add(nameLabel);
    userPanel.add(nameField);
    userPanel.add(Box.createHorizontalStrut(20));
    userPanel.add(phoneLabel);
    userPanel.add(phoneField);

    return userPanel;
}
```

Purpose and Breakdown

- **Layout and Styling**
  - Uses `FlowLayout` with left alignment
  - White background
  - Adds a titled border for "Customer Details"
- **Input Fields**
  - Name and phone number text fields
  - `Box.createHorizontalStrut()` adds spacing between fields
- **User Information Collection**
  - Allows customers to enter their contact details before ordering

4. Menu Panel with Tabbed Interface

```
private JComponent createMenuPanel() {
    JTabbedPane menuTabs = new JTabbedPane();
    menuTabs.setFont(new Font("Arial", Font.BOLD, 14));

    menuTabs.addTab("Food", new JScrollPane(createCategoryPanel(
            new String[]{"Burger", "Pizza", "Pasta", "Fries", "Sandwich", "Tacos"},
            new int[]{150, 250, 180, 100, 120, 150},
            new String[]{
                    "https://images.unsplash.com/photo-1550547660-d9450f859349",
                    "https://images.unsplash.com/photo-1613564834361-9436948817d1",
                    "https://images.unsplash.com/photo-1551183053-bf91a1d81141",
                    "https://plus.unsplash.com/premium_photo-1672774750509-bc9ff226f3e8",
                    "https://images.unsplash.com/photo-1592415486689-125cbbfcbee2",
                    "https://plus.unsplash.com/premium_photo-1661730329741-b3bf77019b39"
            }
    )));

    menuTabs.addTab("Drinks", new JScrollPane(createCategoryPanel(
            new String[]{"Soda", "Coffee", "Coke", "Juice", "Tea", "Milkshake"},
            new int[]{50, 90, 50, 80, 60, 100},
            new String[]{
                    "https://images.unsplash.com/photo-1603968070333-58761fa00853",
                    "https://plus.unsplash.com/premium_photo-1674327105074-46dd8319164b",
                    "https://plus.unsplash.com/premium_photo-1725075086083-89117890371d",
                    "https://images.unsplash.com/photo-1600271886742-f049cd451bba",
                    "https://images.unsplash.com/photo-1597481499666-130f8eb2c9cd",
                    "https://images.unsplash.com/photo-1619158401201-8fa932695178"
            }
    )));

    menuTabs.addTab("Desserts", new JScrollPane(createCategoryPanel(
            new String[]{"Ice Cream", "Cake", "Brownies"},
            new int[]{80, 120, 90},
            new String[]{
                    "https://plus.unsplash.com/premium_photo-1678198786424-c2cc6593f59c",
                    "https://images.unsplash.com/photo-1599785209707-a456fc1337bb",
                    "https://images.unsplash.com/photo-1636743715220-d8f8dd900b87"
            }
    )));

    return menuTabs;
}
```

Purpose and Breakdown

- **Tabbed Navigation**
  - Creates a `JTabbedPane` for menu categories
  - Bold Arial font for tab labels
- **Dynamic Menu Generation**
  - Uses `createCategoryPanel()` to generate menu items
  - Supports multiple categories (Food, Drinks, Desserts)

- **Scrollable Panels**
    - Wraps category panels in `JScrollPane` for scrollability

5. Category Panel (Menu Grid)

```
private JPanel createCategoryPanel(String[] items, int[] prices, String[] imageUrls) {
    JPanel panel = new JPanel(new GridLayout(0, 3, 15, 15));
    panel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));
    panel.setBackground(new Color(245, 245, 245));

    for (int i = 0; i < items.length; i++) {
        panel.add(createMenuCard(items[i], prices[i], imageUrls[i]));
    }
    return panel;
}
```

Purpose and Breakdown

- **Grid Layout**
    - Arranges menu items in 3 columns
    - Adds 15-pixel gaps between items
- **Dynamic Item Generation**
    - Iterates through items, prices, and image URLs
    - Creates a menu card for each item using `createMenuCard()`
- **Styling**
    - Light gray background
    - 15-pixel padding around the panel

6. Menu Card Creation

```java
private JPanel createMenuCard(String itemName, int price, String imageUrl) {
    JPanel card = new JPanel(new GridBagLayout());
    card.setBackground(Color.WHITE);
    card.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createLineBorder(new Color(200, 200, 200)),
            BorderFactory.createEmptyBorder(15, 15, 15, 15)
    ));
    card.setPreferredSize(new Dimension(280, 400));

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.weightx = 1.0;
    gbc.weighty = 1.0;
    gbc.fill = GridBagConstraints.BOTH;

    // Image Panel
    JPanel imagePanel = new JPanel(new BorderLayout());
    imagePanel.setPreferredSize(new Dimension(250, 250));
    try {
        URL url = new URL(imageUrl + "?w=250&h=250&fit=crop");
        BufferedImage originalImage = ImageIO.read(url);
        Image scaledImage = originalImage.getScaledInstance(250, 250, Image.SCALE_SMOOTH);
        JLabel imgLabel = new JLabel(new ImageIcon(scaledImage));
        imagePanel.add(imgLabel, BorderLayout.CENTER);
    } catch (IOException e) {
        JLabel placeholder = new JLabel("📷 Image not available", JLabel.CENTER);
        placeholder.setForeground(Color.GRAY);
        imagePanel.add(placeholder);
    }
    card.add(imagePanel, gbc);

    // Item Details
    gbc.gridy++;
    gbc.weighty = 0.2;
    gbc.fill = GridBagConstraints.HORIZONTAL;

    JPanel detailsPanel = new JPanel();
    detailsPanel.setLayout(new BoxLayout(detailsPanel, BoxLayout.Y_AXIS));
    detailsPanel.setBackground(Color.WHITE);

    JLabel nameLabel = new JLabel(itemName, JLabel.CENTER);
    nameLabel.setFont(new Font("Arial", Font.BOLD, 18));
```

Purpose and Breakdown

- **Comprehensive Item Representation**
    - Loads item image from URL
    - Displays item name and price
    - Provides quantity selection
    - "Add to Cart" functionality
- **Flexible Layout**

    - Uses `GridBagLayout` for precise component positioning
- **Error Handling**
    - Provides placeholder if image cannot be loaded

7. Order Panel

```java
private JPanel createOrderPanel() {
    JPanel orderPanel = new JPanel(new BorderLayout(10, 10));
    orderPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
    orderPanel.setPreferredSize(new Dimension(350, 0));

    orderSummary = new JTextArea();
    orderSummary.setEditable(false);
    orderSummary.setFont(new Font("Monospaced", Font.PLAIN, 14));
    orderSummary.setBackground(new Color(240, 255, 240));

    JButton placeOrderBtn = new JButton("Place Order");
    placeOrderBtn.setBackground(new Color(255, 69, 0));
    placeOrderBtn.setForeground(Color.WHITE);
    placeOrderBtn.addActionListener(e -> showPaymentOptions());

    orderPanel.add(new JScrollPane(orderSummary), BorderLayout.CENTER);
    orderPanel.add(placeOrderBtn, BorderLayout.SOUTH);
    return orderPanel;
}
```

Purpose and Breakdown

- **Order Summary Display**
    - Non-editable `JTextArea` shows selected items
    - Monospaced font for alignment
    - Light green background
- **Order Placement**
    - "Place Order" button triggers payment options
    - Styled with orange-red color
- **Layout**
    - Order summary in the center
    - Order button at the bottom

8. Payment Options Method (`showPaymentOptions()`)

```
private void showPaymentOptions() {
    if (cart.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Your cart is empty!", "Order Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    String name = nameField.getText().trim();
    String phone = phoneField.getText().trim();

    if (name.isEmpty() || phone.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Please enter your name and phone number!", "Missing Information", JOptionPane.WARNING_MESSAGE);
        return;
    }

    JDialog paymentDialog = new JDialog(this, "Payment Options", true);
    paymentDialog.setSize(400, 300);
    paymentDialog.setLayout(new BorderLayout(10, 10));

    JPanel optionsPanel = new JPanel();
    optionsPanel.setLayout(new BoxLayout(optionsPanel, BoxLayout.Y_AXIS));
    optionsPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));

    JButton cashButton = new JButton("Pay in Cash");
    cashButton.setPreferredSize(new Dimension(250, 60));
    cashButton.setMaximumSize(new Dimension(250, 60));
    cashButton.setBackground(new Color(100, 149, 237));
    cashButton.setForeground(Color.WHITE);
    cashButton.setFont(new Font("Arial", Font.BOLD, 14));
    cashButton.setAlignmentX(Component.CENTER_ALIGNMENT);
    cashButton.addActionListener(e -> {
        paymentDialog.dispose();
        confirmOrder(name, phone, "Pay On Counter");
    });
```

Purpose: Validates order details, creates a payment method selection dialog with cash and online payment options, and handles user selection.

2. QR Code Payment Method (`showQRCodePayment()`)

```java
private void showQRCodePayment(String name, String phone) {
    JDialog qrDialog = new JDialog(this, "QR Code Payment", true);
    qrDialog.setSize(500, 600);
    qrDialog.setLayout(new BorderLayout(10, 10));

    try {
        URL qrUrl = new URL("https://api.qrserver.com/v1/create-qr-code/?size=300x300&data=FoodieHubPayment:" + total);
        BufferedImage qrImage = ImageIO.read(qrUrl);
        JLabel qrLabel = new JLabel(new ImageIcon(qrImage));
        qrLabel.setHorizontalAlignment(JLabel.CENTER);
        qrDialog.add(qrLabel, BorderLayout.CENTER);
    } catch (IOException e) {
        JLabel errorLabel = new JLabel("Could not load QR code", JLabel.CENTER);
        errorLabel.setForeground(Color.RED);
        qrDialog.add(errorLabel, BorderLayout.CENTER);
    }

    JPanel infoPanel = new JPanel(new GridLayout(0, 1, 5, 5));
    infoPanel.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));

    infoPanel.add(new JLabel("Scan this QR code to pay ₹" + String.format("%.2f", total), JLabel.CENTER));
    infoPanel.add(new JLabel("Supported apps: PayTM, PhonePe, Google Pay", JLabel.CENTER));
    infoPanel.add(new JLabel("Order will be processed after payment confirmation", JLabel.CENTER));

    JButton confirmButton = new JButton("Payment Done");
    confirmButton.setBackground(new Color(50, 205, 50));
    confirmButton.setForeground(Color.WHITE);
    confirmButton.addActionListener(e -> {
        qrDialog.dispose();
        confirmOrder(name, phone, "Online");
    });

    JButton backButton = new JButton("Back to Payment Options");
    backButton.addActionListener(e -> {
        qrDialog.dispose();
        showPaymentOptions();
    });
```

Purpose: Generates a dynamic QR code for online payment, displays payment instructions, and provides navigation options.

3. Order Confirmation Method (`confirmOrder()`)

```java
private void confirmOrder(String name, String phone, String paymentMethod) {
    JDialog confirmDialog = new JDialog(this, "Order Confirmation", true);
    confirmDialog.setSize(500, 400);
    confirmDialog.setLayout(new BorderLayout(10, 10));

    try {
        URL url = new URL("https://img.freepik.com/free-vector/order-confirmed-concept-illustration_114360-1486.jpg");
        BufferedImage img = ImageIO.read(url);
        Image scaledImg = img.getScaledInstance(200, 200, Image.SCALE_SMOOTH);
        JLabel imgLabel = new JLabel(new ImageIcon(scaledImg));
        imgLabel.setHorizontalAlignment(JLabel.CENTER);
        confirmDialog.add(imgLabel, BorderLayout.CENTER);
    } catch (IOException e) {
        JLabel errorLabel = new JLabel("Order Confirmed!", JLabel.CENTER);
        errorLabel.setFont(new Font("Arial", Font.BOLD, 16));
        confirmDialog.add(errorLabel, BorderLayout.CENTER);
    }

    JPanel infoPanel = new JPanel(new GridLayout(0, 1, 5, 5));
    infoPanel.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));

    infoPanel.add(new JLabel("Thank you for your order!", JLabel.CENTER));
    infoPanel.add(new JLabel("Name: " + name, JLabel.CENTER));
    infoPanel.add(new JLabel("Phone: " + phone, JLabel.CENTER));
    infoPanel.add(new JLabel("Payment Method: " + paymentMethod, JLabel.CENTER));

    JButton closeButton = new JButton("Close");
    closeButton.setBackground(Color.RED);
    closeButton.setForeground(Color.WHITE);

    closeButton.addActionListener(e -> {
        confirmDialog.dispose();
        showStarRatingDialog(); // Show Star Rating Dialog after closing confirmation dialog
    });

    JPanel buttonPanel = new JPanel(new FlowLayout());
    buttonPanel.add(closeButton);

    confirmDialog.add(infoPanel, BorderLayout.NORTH);
    confirmDialog.add(buttonPanel, BorderLayout.SOUTH);
```

Purpose: Displays order confirmation with customer details, confirmation image, and triggers the star rating dialog.

4. Star Rating Dialog Method (`showStarRatingDialog()`)

```java
/-------------------------------------------------------------------------------------------------------------/
private void showStarRatingDialog() {
        JDialog ratingDialog = new JDialog(this, "Rate Our Service", true);
        ratingDialog.setSize(500, 200);
        ratingDialog.setLayout(new BorderLayout(10, 10));

        // Instruction Label
        JLabel instructionLabel = new JLabel("How would you rate our service?");
        instructionLabel.setFont(new Font("Arial", Font.BOLD, 16));
        instructionLabel.setHorizontalAlignment(JLabel.CENTER);
        ratingDialog.add(instructionLabel, BorderLayout.NORTH);

        // Star Rating Panel
        JPanel starPanel = new JPanel(new FlowLayout());
        starPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        // Create star labels for rating
        JLabel[] starLabels = new JLabel[5];
        try {
            // Star image URL - replace with your actual URL
            URL starUrl = new URL("https://t3.ftcdn.net/jpg/01/09/84/42/240_F_109844239_A7MdQSDf4y1H80cfvHZuSa0zKBkZ68S7.jpg");

            for (int i = 0; i < starLabels.length; i++) {
                // Load the star image
                BufferedImage originalImage = ImageIO.read(starUrl);
                Image scaledImage = originalImage.getScaledInstance(50, 50, Image.SCALE_SMOOTH);
                ImageIcon starIcon = new ImageIcon(scaledImage);

                // Create label with star icon
                starLabels[i] = new JLabel(starIcon);
                starLabels[i].setPreferredSize(new Dimension(50, 50));

                int rating = i + 1; // Star number (1 to 5)
                final JLabel currentLabel = starLabels[i];

                currentLabel.addMouseListener(new MouseAdapter() {
                    @Override
                    public void mouseClicked(MouseEvent e) {
                        // Highlight selected stars
                        try {
                            for (int j = 0; j < starLabels.length; j++) {
                                BufferedImage img = ImageIO.read(starUrl);
                                if (j < rating) {
                                    // Darken the star to show selection
                                    Graphics2D g2d = img.createGraphics();
                                    g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.7f));
                                    g2d.setColor(new Color(218, 165, 32)); // Gold color
                                    g2d.fillRect(0, 0, img.getWidth(), img.getHeight());
                                    g2d.dispose();
                                }
```

Purpose: Creates an interactive star rating dialog with visual feedback, allowing users to rate their experience with hover and click effects.

5. Feedback Dialog Method (`showFeedbackDialog()`)

```
private void showFeedbackDialog() {
    JDialog feedbackDialog = new JDialog(this, "Feedback", true);
    feedbackDialog.setSize(400, 200);
    feedbackDialog.setLayout(new BorderLayout(10, 10));

    JLabel instructionLabel = new JLabel("Any additional feedback?");
    instructionLabel.setFont(new Font("Arial", Font.BOLD, 16));
    instructionLabel.setHorizontalAlignment(JLabel.CENTER);

    JTextArea feedbackArea = new JTextArea();
    JScrollPane scrollPane = new JScrollPane(feedbackArea);

    JButton submitButton = new JButton("Submit Feedback");
    submitButton.addActionListener(e -> {
        String feedback = feedbackArea.getText();
        JOptionPane.showMessageDialog(feedbackDialog, "Thank you for your feedback!");
        feedbackDialog.dispose();
    });

    feedbackDialog.add(instructionLabel, BorderLayout.NORTH);
    feedbackDialog.add(scrollPane, BorderLayout.CENTER);
    feedbackDialog.add(submitButton, BorderLayout.SOUTH);

    feedbackDialog.setLocationRelativeTo(this);
    feedbackDialog.setVisible(true);
}

private JPanel createFooter() {
    JPanel footerPanel = new JPanel();
    footerPanel.setPreferredSize(new Dimension(getWidth(), 50));
    footerPanel.setBackground(new Color(51, 51, 51));

    JLabel copyrightLabel = new JLabel("© 2024 Foodie's Hub. All rights reserved.");
    copyrightLabel.setForeground(Color.WHITE);
    copyrightLabel.setFont(new Font("Arial", Font.PLAIN, 12));
    footerPanel.add(copyrightLabel);

    return footerPanel;
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(FoodOrderingSystem::new);
 }
```

Purpose: Provides a dialog for users to submit additional textual feedback after rating their experience.

📌 Future Aspects of the Project

The **Foodie's Hub - Premium Ordering System** can be expanded in various ways to add advanced features:

1. **Database Integration:** Use MySQL or SQLite for storing menu items, order history, and customer data.
2. **User Authentication:** Create login/signup features and an admin dashboard for better management.
3. **Online Payment Integration:** Link to real-time payment gateways like **PayPal** or **Stripe** for secure transactions.
4. **Dynamic Menu Management:** Allow admins to add, update, or delete items.
5. **Feedback Analysis:** Use analytics to study customer feedback and improve the menu.
6. **Mobile and Web Versions:** Develop Android or web apps for broader accessibility.

📌 What We Have Learned from the Project

Technical Skills:

- **Java Swing:** Building GUIs using components like **JFrame**, **JPanel**, **JLabel**, etc.
- **Event Handling:** Managing user interactions through **ActionListeners** and **MouseAdapter**.
- **Layout Management:** Using **BorderLayout**, **GridLayout**, and **FlowLayout** for structured UIs.
- **Error Handling:** Handling exceptions like `IOException` and managing user inputs via **JOptionPane**.

Project Management:

- **Modular Design:** Dividing code into clear, manageable sections for easier maintenance.
- **Problem-Solving:** Debugging errors during image loading and event handling.

- **Teamwork:** Collaborating using version control (Git) and documenting the code effectively.

Real-World Applications:

- Understanding the practical aspects of food ordering systems.
- Gaining exposure to **payment methods**, **feedback systems**, and user-centered design.
- Learning the basics of project planning, implementation, and testing.

📌 Challenges Faced:

1. **Handling User Input Errors:** Validating customer details and preventing incorrect inputs.
2. **Image Loading Failures:** Managing exceptions when loading images from URLs.
3. **Ensuring User-Friendly UI:** Adjusting layout issues to maintain a responsive, appealing design.
4. **Scalability:** Adapting the code for future expansions like database integration and online payments.