|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No,** | **Topic** | **Page No.** | **Remarks** |
| **1.** | *Acknowledgement* | *2* |  |
| **2.** | *Project: Introduction and Tools Used* |  |  |
| **3.** | *Command Line Application* |  |  |
| **4.** | *Data Structure Used* |  |  |
| **5.** | *Execution and Errors* |  |  |
| **6.** | *Source Code* |  |  |
| **7.** | *Graphical User Interface Application* |  |  |
| **8.** | *Execution of GUI Application* |  |  |
| **9.** | *Data Structure Used* |  |  |
| **10.** | *Errors* |  |  |
| **11.** | *References* |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Project:**

Food Delivery System.

**Introduction:**

The project at hand is a comprehensive food delivery system designed to streamline operations and enhance the dining experience for both customers and restaurant staff. Divided into two distinct parts, a Command-Line Interface (CLI) version and a Graphical User Interface (GUI) version, this system offers versatility and usability tailored to different user preferences.

The CLI version provides a straightforward text-based interface suitable for efficient command execution and automation, while the GUI version offers a visually appealing and intuitive interface for users accustomed to graphical interactions. With features ranging from menu item management to customer authentication and order processing, this project aims to deliver a seamless solution for restaurant owners and customers alike, enhancing efficiency, accuracy, and customer satisfaction.

This report delves into the design, implementation, features, and potential extensions of both the CLI and GUI versions of the restaurant management system.

**Tools Used:**

The Command-Line Interface (CLI) version of the application is developed utilizing the C Programming Language, leveraging a variety of fundamental programming concepts such as Dynamic Memory Allocation, Conditionals, Loops, and Functions. However, the crux of the application's logic is formulated around the utilization of advanced data structures including Arrays and Linked Lists to effectively store and process data. These data structures facilitate efficient data management and manipulation, ensuring optimal performance and scalability of the application.

Conversely, the Graphical User Interface (GUI) version of the application is implemented using the JAVA Programming Language, chosen for its inherent capability to facilitate the creation of intuitive and visually appealing GUI pages. Employing libraries such as Swing, AWT, and AWT events, the GUI version harnesses the power of these tools to create an interactive and user-friendly interface. Moreover, the GUI version incorporates fundamental programming concepts like conditionals, loops, and Object-Oriented Programming (OOP) principles, enhancing code modularity and maintainability.

In both versions of the application, data management is facilitated through the utilization of various data structures such as Arrays, Array Lists, and Linked Lists. These data structures are instrumental in storing and organizing data, ensuring efficient retrieval and manipulation operations. By leveraging a combination of advanced programming concepts and versatile data structures, both the CLI and GUI versions of the application deliver a robust and comprehensive solution for restaurant management, catering to the diverse needs of users while prioritizing usability and performance.

**Introduction:**

The Graphical User Interface (GUI) implementation of the food delivery system offers an intuitive and visually engaging platform for users to interact with. Developed using Java programming language, it leverages Swing, AWT, and AWT events libraries to create dynamic and responsive user interfaces. With features such as menu item management, customer authentication, and order processing, the GUI version enhances user experience by providing a seamless and interactive interface. This section of the report explores the design, development, and functionality of the GUI version, emphasizing its usability and potential for further enhancements.

**Project Structure:**

**Classes Used:**

1. AddMenuItemScreen
2. CustomerLoginScreen

Authentication

CustomerLoginScreen

1. HotelBasedOrderingScreen
2. InvoiceScreen
3. MainScreen
4. MenuLinkedList

MenuItem

MenuLinkedList

1. OrderingScreen
2. OrderOptionsScreen
3. OwnerLoginScreen’
   1. OwnerAuthentication
   2. OwnerLoginScreen
4. PaymentGatewayScreen
5. WelcomePage (This is the entry point of the program).

* **WelcomePage class:**

The WelcomePage class serves as the elegant gateway to our innovative demo application. Designed with both functionality and aesthetic appeal in mind, this class encapsulates a welcoming interface that engages users from the moment they launch our application.

**Title and Frame Configuration:**

The class sets the stage with a captivating title, "Welcome to Food Delivery App," creating a professional and inviting atmosphere.

The frame is thoughtfully sized at 400x300 pixels, offering an optimal balance between visual appeal and user-friendly dimensions.

Its centralized location on the screen ensures a seamless and centred presentation, enhancing the overall user experience.

**Image Section for Logo:**

A visually striking logo is incorporated into the interface, adding a touch of branding and aesthetic charm.

The logo, sourced from the file "Logo.png," is intelligently scaled to a tasteful 200x200 pixels, striking a harmonious balance between prominence and subtlety.

Robust error handling ensures a graceful user experience, with informative messages in case of image loading issues.

**Button for Seamless Navigation:**

A prominent "Start" button beckons users to embark on their journey within the application.

Meticulously styled, the button boasts an elegant font, a custom background colour (#3B59B6), and contrasting white text, creating a visually appealing call-to-action.

Eliminating distractions, the button seamlessly triggers the transition to the main application screen with a single click, enhancing user flow and satisfaction.

**ActionListener for User Interaction:**

The button is equipped with an ActionListener, ensuring a responsive and intuitive interaction.

Upon clicking "Start," the class dynamically initiates the MainScreen – a pivotal component of our application's functionality.

**Effect and User Engagement:**

The WelcomePage class serves as the gateway that not only introduces users to our application but also leaves a lasting impression. Its meticulously crafted interface, blending imagery, branding, and interactive elements, aims to captivate users from the outset. By seamlessly

transitioning to the MainScreen upon user interaction, the class sets the stage for a seamless and enjoyable user experience.

* **MainScreen class:**

The **MainScreen** class serves as the central hub of our Food Delivery System application, offering a visually appealing and user-friendly interface for both owners and customers. The class orchestrates a seamless transition between different login screens, fostering a cohesive user experience.

**Key Components:**

1. **Frame Configuration:**
   * Titled "Food Delivery System," the frame is thoughtfully configured with a size of 600x400 pixels and centred on the screen, ensuring an aesthetically pleasing and accessible layout.
   * The use of a Border Layout facilitates an organized structure for the welcome message and login buttons.
2. **Welcome Message Panel:**
   * A dedicated panel, **welcomePanel**, conveys a warm welcome to users through a visually striking message.
   * The message is styled with a bold font, vibrant colour, and centre alignment, enhancing readability and visual appeal.
3. **Login Buttons:**
   * Two prominent login buttons, "Owner Login" and "Customer Login," are strategically placed in a separate panel (**buttonPanel**) for clarity.
   * Each button is meticulously designed with a bold font, contrasting color scheme, and a polished appearance, promoting a professional and engaging user interface.
4. **Button Actions:**
   * Clicking the "Owner Login" button seamlessly transitions users to the **OwnerLoginScreen**, allowing owners to access administrative functionalities.
   * The "Customer Login" button triggers a smooth transition to the **CustomerLoginScreen**, providing customers with access to their accounts.

* **OwnerLoginScreen class:**

This screen is for the owner to log in to the app.

**Frame Setup:**

Titled "Owner Login," the frame is configured to dispose gracefully when closed, offering a user-friendly experience.

The dimensions (400x250 pixels) strike a balance between visual appeal and practicality, and the window is centrally located for an optimal presentation.

**Input Fields:**

A grid layout with 5 rows and 2 columns neatly organizes the essential input fields.

Username, Password, Email, and Phone Number fields provide a comprehensive set of owner credentials.

Components are labelled appropriately, fostering user clarity and ease of interaction.

**Navigation Buttons:**

A "Back" button seamlessly transitions users to the MainScreen, ensuring intuitive navigation.

The "Login" button triggers the authentication process, validating entered details and facilitating access to the AddMenuItemScreen upon success.

**Authentication Handling:**

User entries are retrieved, and empty field validation ensures completeness.

The OwnerAuthentication class manages the authentication process, providing robust handling for username and password verification.

Error messages promptly alert users to incorrect credentials or missing information, enhancing the overall user experience.

* **AddMenuItemScreen class:**

The **AddMenuItemScreen** class facilitates the seamless addition of new menu items to our Food Delivery System application. With a clean and intuitive interface, users can efficiently input menu details, enhancing the system's versatility and keeping the menu dynamically updated.

**Key Components:**

1. **Frame Configuration:**
   * Titled "Add Menu Items," the frame is configured to gracefully dispose when closed, maintaining a user-friendly experience.
   * Sized at 400x200 pixels and centred on the screen, the window provides an optimal layout for menu item input.
2. **Input Fields:**
   * Organized within a Grid Layout of 3 rows and 2 columns, input fields for "Menu Item Name" and "Price" allow for the systematic addition of new items.
   * Clear labelling and consistent spacing contribute to a user-friendly interface.
3. **Action Buttons:**
   * The "Add" button captures and validates user input, adding new menu items to the **MENU\_FILE**.
   * The "Done" button finalizes the addition process, saves the menu items to the file, and seamlessly transitions back to the **MainScreen**.
4. **File Handling:**
   * The **initializeMenuFile** method preloads the **MENU\_FILE** with default menu items for an initial system setup.
   * **addMenuItemToFile** appends user-inputted menu items to the file, maintaining a structured record.
   * **saveMenuItems** serves as a placeholder for future enhancements, allowing for additional operations during the saving process.

* **CustomerLoginScreen class:**

The **CustomerLoginScreen** class provides a secure and user-friendly interface for customer login within our Food Delivery System application. With fields for essential customer details, this class ensures a smooth onboarding experience for customers while maintaining robust authentication.

**Key Components:**

1. **Frame Configuration:**
   * Titled "Customer Login," the frame is configured to dispose gracefully when closed, maintaining a clean and user-friendly experience.
   * Sized at 400x300 pixels and centred on the screen, the window layout accommodates the necessary input fields.
2. **Input Fields:**
   * Organized within a Grid Layout of 6 rows and 2 columns, the input fields capture critical customer information: username, password, email, phone number, and address.
   * Clear labelling and consistent spacing contribute to an intuitive and accessible user interface.
3. **Action Buttons:**
   * The "Back" button seamlessly transitions users back to the **MainScreen** for convenient navigation.
   * The "Login" button triggers the authentication process, validating entered details and allowing access to the **OrderOptionsScreen** upon success.
4. **File Handling:**
   * The **Authentication** class manages the authentication process, ensuring robust username and password verification.
   * The **saveCustomerData** method appends customer details to the **CUSTOMER\_DATA\_FILE**, facilitating a secure and structured record of customer information.

* **OrderOptionsScreen class:**

The **OrderOptionsScreen** class serves as a central hub for customers to choose between hotel-based and food-based ordering within our Food Delivery System application. This class provides a user-friendly interface for initiating different ordering experiences while ensuring easy navigation and access to other functionalities.

**Key Components:**

1. **Frame Configuration:**
   * Titled "Order Options," the frame is configured to exit the application when closed, maintaining a straightforward and user-friendly experience.
   * Sized at 400x200 pixels and centred on the screen, the window layout offers an optimal balance between visual appeal and practicality.
2. **Ordering Options:**
   * Organized within a Grid Layout of 3 rows and 1 column, the ordering options are presented as buttons, facilitating a clean and intuitive interface.
   * The "Hotel-based Ordering" button allows customers to explore hotel-specific ordering functionalities through the **HotelBasedOrderingScreen**.
   * The "Food-based Ordering" button initiates the food-based ordering process through the **OrderingScreen**.
3. **Exit Button:**
   * The "Exit" button provides a convenient way for users to leave the order options screen and return to the **MainScreen**.

* **MenuLinkedList class:**

The **MenuLinkedList** class represents a linked list data structure specifically designed to manage menu items within our Food Delivery System application. This class enables the organization and efficient retrieval of menu items, providing a foundation for various functionalities such as displaying the menu and finding items by their serial number.

**Key Components:**

1. **Linked List Structure:**
   * Each menu item is represented by the **MenuItem** class, containing attributes such as serial ID, name, price, and a reference to the next item in the list.
   * The **head** attribute points to the first item in the linked list.
2. **Insertion Method:**
   * The **insert** method adds a new menu item to the end of the linked list, ensuring that the order of items is maintained.
3. **Display Method:**
   * The **display** method traverses the linked list and prints the details of each menu item, including serial ID, name, and price.
4. **Size and Head Retrieval:**
   * The **size** method calculates the number of items in the linked list.
   * The **getHead** method returns the head of the linked list.
5. **Item Retrieval by Serial Number:**
   * The **findMenuItemBySerialNo** method searches for a menu item based on its serial number, returning the corresponding **MenuItem** object or **null** if not found.

* **HotelBasedOrderingScreen class:**

The **HotelBasedOrderingScreen** class represents the screen where users can select a hotel for placing their food orders within our Food Delivery System application. This class provides an interface with buttons for each hotel, allowing users to choose their preferred hotel for ordering.

**Key Components:**

1. **Hotel Buttons:**
   * The class includes buttons for each hotel (e.g., Hotel 1, Hotel 2), providing a user-friendly interface for selecting a specific hotel for ordering.
   * Each hotel button is associated with an **ActionListener** that triggers the opening of the **OrderingScreen** for the selected hotel.
2. **ActionListener for Hotel Buttons:**
   * Each hotel button has an **ActionListener** that, upon activation, calls the **openOrderingScreen** method, passing the corresponding hotel name as an argument.
3. **Back Button:**
   * The "Back" button allows users to return to the **OrderOptionsScreen** for choosing between hotel-based and food-based ordering.
4. **openOrderingScreen Method:**
   * The private method **openOrderingScreen** disposes of the current **HotelBasedOrderingScreen** and opens the **OrderingScreen** for the selected hotel

* **OrderingScreen class:**

The **OrderingScreen** class represents the screen where users can view and select menu items for placing orders within our Food Delivery System application. This class provides an interface that displays the available menu items, allows users to add items to their orders, and proceed to the payment gateway.

**Key Components:**

1. **Menu Display:**
   * The class dynamically loads menu items from the **menu\_items.txt** file into a **MenuLinkedList**. It then displays the menu items with their serial number, name, and price in a structured grid layout.
2. **Order Panel:**
   * Users can interact with the order panel to add items to their orders. The order panel includes text fields for entering the serial number and quantity of the desired menu item, along with an "Add" button to add the item to the order.
3. **Orders Panel:**
   * The orders panel displays the selected items along with their quantities and total prices. Each time an item is added, a corresponding label is dynamically created and added to this panel.
4. **Payment Button:**
   * The "Proceed to Payment" button allows users to proceed to the payment gateway screen for completing the order. Clicking this button disposes of the current **OrderingScreen** instance and opens a new **PaymentGatewayScreen**.
5. **ActionListener for Add Button:**
   * The "Add" button has an **ActionListener** that validates user input, calculates the total price for the selected item, writes the order details to the **orders.txt** file, and updates the orders panel.
6. **Load Menu Items Method:**
   * The **loadMenuItems** method reads menu items from the **menu\_items.txt** file and inserts them into a **MenuLinkedList**. If the file doesn't exist, it creates a default one with some entries.
7. **Create Default Menu Items File Method:**
   * The **createDefaultMenuItemsFile** method creates a default **menu\_items.txt** file with some sample menu entries in case the file is missing.

**User Interaction:**

Users can interact with the **OrderingScreen** by viewing the menu, entering the serial number and quantity of desired items, clicking the "Add" button to add items to their orders, and proceeding to payment using the "Proceed to Payment" button.

* **PaymentGatewayScreen class:**

The **PaymentGatewayScreen** class represents the screen where users can choose their payment method for completing the order in our Food Delivery System application. This class provides options for online payment and cash on delivery, allowing users to make a secure transaction.

**Key Components:**

1. **Online Payment Options:**
   * The screen presents buttons for different online payment methods, such as PAYTM, PHONEPAY, GPAY, and CARD PAY.
   * Users can select their preferred online payment method by clicking the respective button.
2. **Cash on Delivery (COD) Option:**
   * Users can choose the "Cash on Delivery" option, indicating that they will pay in cash when the food is delivered.
   * A confirmation message is displayed, informing users that they will receive their food after 30-40 minutes.
3. **Cancel Button:**
   * Users can cancel the payment process and return to the main screen by clicking the "Cancel" button.
4. **Open Online Payment Options Method:**
   * This method opens a dialog for users to choose from various online payment methods.
   * After selecting an online payment method, a confirmation message is displayed, and the invoice is opened.
5. **Open Invoice Method:**
   * This method prompts users to decide whether they want to generate an invoice.
   * If users choose to generate an invoice, the **InvoiceScreen** is opened, providing details about the order.

**User Interaction:**

Users interact with the **PaymentGatewayScreen** by selecting their preferred payment method (online or cash on delivery). The screen offers a choice of online payment options and provides a confirmation message for each selection. Users can also decide whether to generate an invoice for their order.

* **InvoiceScreen class:**

The **InvoiceScreen** class is responsible for displaying the invoice containing customer information, order details, and the total bill (including 18% GST) in our Food Delivery System application. This class provides a summary of the customer's order and serves as a confirmation of the transaction.

**Key Components:**

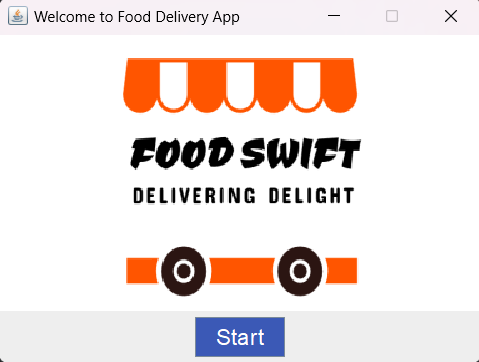
1. **Customer Information:**
   * The invoice includes the customer's name, email, phone, and address information.
2. **Order Details:**
   * The order details are loaded from the "orders.txt" file and displayed in a scrollable text area.
3. **Total Bill Calculation:**
   * The total bill is calculated by summing up the individual order prices and adding 18% GST (Goods and Services Tax).
4. **Thank You Message:**
   * A "Thank you! Please come again..." message is displayed at the end of the invoice.
5. **Load Customer Information Method:**
   * This method reads the customer information (name, email, phone, address) from the "customer\_data.txt" file.
6. **Load Orders Method:**
   * This method reads and loads the order details from the "orders.txt" file.
7. **Calculate Total Bill Method:**
   * This method calculates the total bill amount by summing up individual order prices and adding GST.

**User Interaction:**

Users do not directly interact with the **InvoiceScreen** class as it is automatically displayed after the user completes the payment process. The screen serves as a confirmation of the order and provides customers with a summary of their transaction.

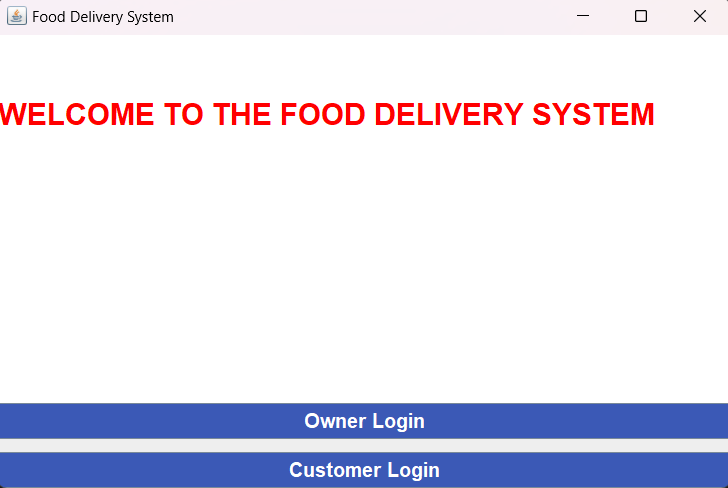
* **Flow of Program (Customer Side):**
* **Main Screen:**
  + The application starts with the Main Screen.
  + Users are presented with options like "Customer Login" and "Owner Login."
* **Customer Login:**
  + If the user selects "Customer Login," the Customer Login Screen is displayed.
  + Users enter their details, including username, password, email, phone, and address.
  + Clicking the "Login" button triggers authentication.
* **Authentication:**
  + The Authentication class validates the user's credentials.
  + If the customer is a new user (no existing customer\_data.txt file), a new file is created to store their data.
  + If the customer already exists, the entered credentials are compared with the stored data.
* **Order Options:**
  + Upon successful login, users are directed to the Order Options Screen.
  + Options include "Hotel-based Ordering," "Food-based Ordering," and "Exit."
* **Hotel-based Ordering:**
  + Choosing "Hotel-based Ordering" opens the HotelBasedOrderingScreen.
  + Users see a list of hotels (Hotel 1 to Hotel 6) and a "Back" button.
  + Selection of a hotel triggers the Opening of the Ordering Screen for the chosen hotel.
* **Food-based Ordering:**
  + Choosing "Food-based Ordering" opens the OrderingScreen.
  + The screen displays a menu loaded from the menu\_items.txt file.
  + Users can view items with serial numbers, names, and prices.
  + They can add items to their order by entering the serial number and quantity.
* **Order Confirmation:**
  + The user can add multiple items to their order.
  + The "Proceed to Payment" button takes them to the PaymentGatewayScreen.
* **Payment Gateway:**
  + Users can choose between "Online Payment" and "Cash on Delivery."
  + Online payment methods include PAYTM, PHONEPAY, GPAY, and CARD PAY.
  + Upon selection, a confirmation message is shown, and an invoice can be generated.
* **Invoice Generation:**
  + If the user chooses "Cash on Delivery," a thank you message is displayed.
  + If the user chooses online payment, they receive a thank you message and are prompted to generate an invoice.
  + An invoice screen (InvoiceScreen) is displayed, showing customer details, order summary, and total bill with GST.
* **Exit and Return:**
  + At various stages, users have the option to "Exit" the application.
  + Upon completion of the process, users can choose to generate an invoice or exit.
  + In any screen, users can go back to the Main Screen using "Back" buttons.
* **Flow of Program (Owner Side):**
* **Main Screen:**
  + Similar to the customer, the owner starts at the **MainScreen**.
  + Options include "Hotel Login," "Customer Login," and "Exit."
* **Add Menu Item:**
  + This adds menu items, a name of menu item and its price.
  + The owner can add, remove, or modify menu items, updating the **menu\_items.txt** file.
* **Exit:**
  + At any point, the owner can choose to "Exit" the application.

**Start Screen:**



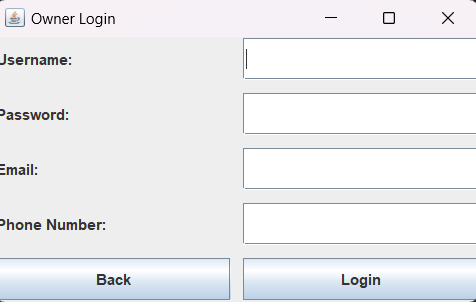
* This page shows the logo of the app and a start button.
* When the start button is pressed the main screen activity starts and this one gets disposed.

**Main Screen:**



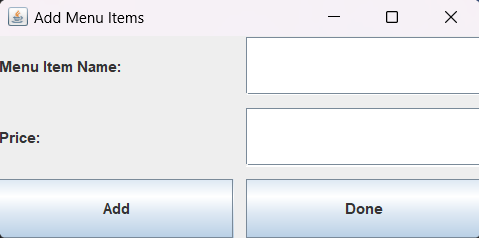
* This page show the welcome message and gives two options one for owner login and other for Customer login.
* Owner Login button starts Owner Login activity and disposes this one.
* Customer Login button starts the Customer Login activity and disposes this one.

**Owner Login:**



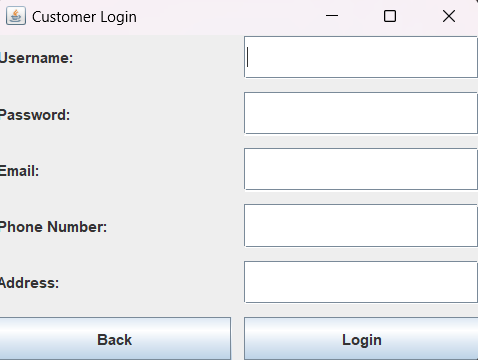
* This page is for the Owner to log in with his credentials, if the owner is logging in for the first time the data entered will get stored in a file called “owner\_data” and starts the Adding Menu Items Activity and if he’s trying to log in for the second time, The Authentication method will confirm the username and password and only start the next activity if it matches with the details in the owner data file ,else it shows a wrong password or username.
* To login use the Login button and to go back to the main screen press the back button.

**Add Menu Items:**



* This screen offers two text fields to add the name of menu item and it’s price, when Add button is pressed the menu item along with it’s price will get stored in a file called menu\_items with a unique serial ID, and the prompt will ask if the owner wants to add more menu items or not.
* Done button will dispose this screen and start the main screen.

**Customer Login:**



* On this screen the customer can add his details and all the data will get stored in a file called customer\_data, this page implements the same Authentication methods and button features.

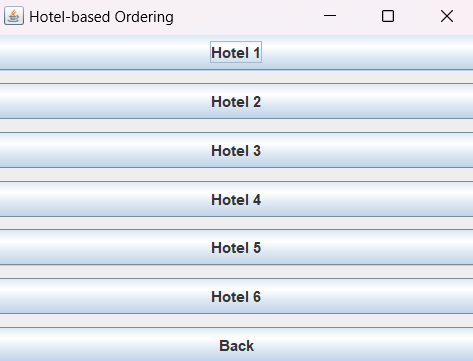
**Order Options:**



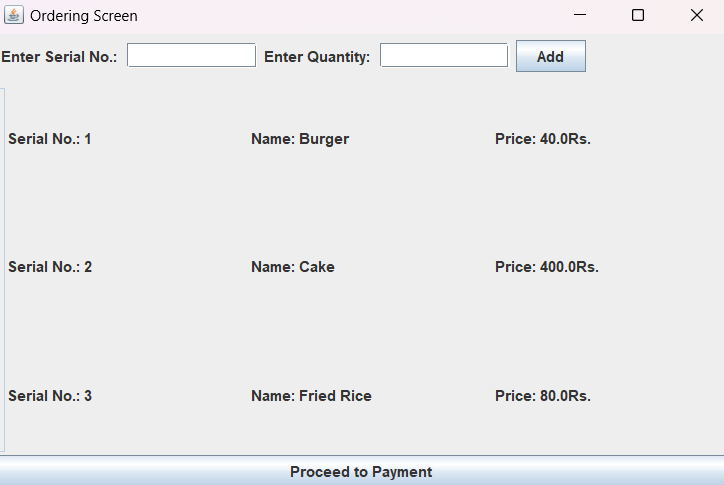
* This screen shows all three options, 1st one starts the Hotel-Based Ordering activity , 2nd starts the Ordering Activity directly and the Exit button quits the program.

**Hotel Based Ordering:**

* This screen shows different Hotel Options and a Back Button to get back to the Ordering Options activity.
* Pressing the hotel option starts the Ordering Screen activity.

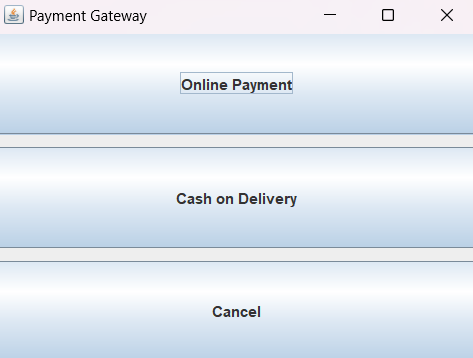


**Ordering Screen:**

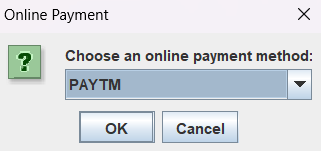


* On this screen user can enter serial no. of menu item and it’s quantity to add order, if the owner has added a menu item file then it’s content will be visible on the screen, else the default menu item file.
* Proceed to payment buttons starts the payment gateway screen and disposes this one.

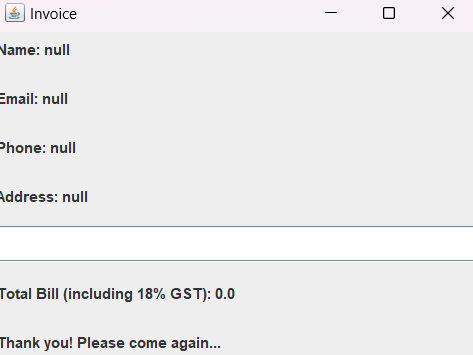
**Payment Gateway:**



* This screen offers option for online payment and cash on delivery option cancel button cancels the whole program execution.
* Cash on delivery option takes us to invoice activity and online payment screen offeres different options for online payment.



**Invoice:**



* The invoice will show all the customer info from that file also show the total bill.
* Incase there’s any error all the data will become null.

package main.java.com.swift.app;  
  
import java.io.\*;  
import java.util.\*;  
  
class MenuItem {  
 int serialId;  
 String name;  
 double price;  
 MenuItem next;  
  
 public MenuItem(int serialId, String name, double price) {  
 this.serialId = serialId;  
 this.name = name;  
 this.price = price;  
 this.next = null;  
 }  
}  
  
public class MenuLinkedList {  
 private MenuItem head;  
  
 public MenuLinkedList() {  
 head = null;  
 }  
  
 public void insert(int serialId, String name, double price) {  
 MenuItem newItem = new MenuItem(serialId, name, price);  
 if (head == null) {  
 head = newItem;  
 } else {  
 MenuItem current = head;  
 while (current.next != null) {  
 current = current.next;  
 }  
 current.next = newItem;  
 }  
 }  
  
 public void display() {  
 MenuItem current = head;  
 while (current != null) {  
 System.*out*.println("Serial ID: " + current.serialId);  
 System.*out*.println("Name: " + current.name);  
 System.*out*.println("Price: " + current.price);  
 System.*out*.println();  
 current = current.next;  
 }  
 }  
  
 // Method to get the head of the linked list  
 public MenuItem getHead() {  
 return head;  
 }  
  
 // Method to get the size of the linked list  
 public int size() {  
 int count = 0;  
 MenuItem current = head;  
 while (current != null) {  
 count++;  
 current = current.next;

}  
 return count;  
 }  
  
 // Method to find a menu item by serial number  
 public MenuItem findMenuItemBySerialNo(int serialNo) {  
 MenuItem current = head;  
 while (current != null) {  
 if (current.serialId == serialNo) {  
 return current;  
 }  
 current = current.next;  
 }  
 return null;  
 }  
}

**Class Structure:**

* **Class Name:** MenuLinkedList
* **Fields:**
  + private MenuItem head: Represents the head of the linked list, initially set to null.

**2. Constructors:**

* **public MenuLinkedList()**
  + Initializes an empty linked list.

**3. Methods:**

* **public void insert(int serialId, String name, double price)**
  + Inserts a new MenuItem at the end of the linked list.
  + If the list is empty, the new item becomes the head; otherwise, it's added at the end.
* **public void display()**
  + Displays the contents of the linked list.
  + Iterates through the list and prints the serial ID, name, and price of each menu item.
* **public MenuItem getHead()**
  + Returns the head of the linked list.
* **public int size()**
  + Returns the size (number of elements) in the linked list.
  + Iterates through the list, counting the elements.
* **public MenuItem findMenuItemBySerialNo(int serialNo)**
  + Finds and returns a menu item based on its serial number.
  + Iterates through the list until it finds an item with the specified serial number.
  + Returns null if no item is found.

**4. File Handling:**

* **Data Storage in File:**
  + The class has a commented-out section in the main method (which can be uncommented) to read data from a file (menu\_items.txt).
  + The file is expected to have a specific format where each menu item is represented by three lines: serial ID, name, and price.
* **How Data is Stored:**
  + The insert method is used to add menu items to the linked list.
  + The commented-out code in the main method demonstrates reading from a file and inserting items into the linked list.
* **File Reading:**
  + Uses Scanner to read from a file.
  + Parses each line to extract serial ID, name, and price.
  + Calls the insert method to add items to the linked list.