

## Index

Sr. No,	Topic	Page No.	Remarks
1.	<i>Acknowledgement</i>	2	
2.	<i>Project: Introduction and Tools Used</i>	3	
3.	<i>Command Line Application</i>	4	
4.	<i>Data Structure Used</i>	4	
5.	<i>Execution</i>	5	
7.	<i>Graphical User Interface Application</i>	15	
8.	<i>Execution of GUI Application</i>	16	
9.	<i>Data Structure Used</i>	21	
10.	<i>References</i>	23	

## Acknowledgement

We express our sincere gratitude to Mrs. Santayani Saha for her invaluable support and mentorship throughout the entirety of this group project. Her profound expertise, unwavering guidance, and insightful feedback have significantly contributed to our project's success. Her dedication to fostering our growth and development exemplifies the epitome of professional excellence. We are deeply appreciative of her commitment and invaluable contributions to our collective journey.

## Project: Introduction and Tools Used

### 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.

## Command Line Application

This is a food delivery system implemented in C programming language. It allows users to interact with the system as either an owner or a customer.

### Key features of the system include:

1. **Owner Login:** Owners can log in to manage their menu items and update their information.
2. **Customer Login:** Customers can log in to place orders, providing their personal details such as name, address, phone number, email, and password.
3. **Menu Management:** Owners can add menu items with their prices, and customers can view the available menu items.
4. **Order Placement:** Customers can select items from the menu and place orders. They can specify the quantity of each item they wish to order.
5. **Payment Processing:** Customers can choose between online payment options or cash on delivery.
6. **Invoice Generation:** After placing an order, customers can generate an invoice displaying their details, order summary, and total bill amount, including applicable taxes.

The system provides options for both hotel-based and food-based ordering, catering to different preferences and needs. It aims to streamline the food delivery process, providing convenience and efficiency for both owners and customers.

## Data Structures Used

### Array:

To store customer details and owner details we need to store some data in **string** form, in C programming, there's no built-in **string** data type, so for that purpose we have used **Array of characters**.

### Linked List (for Menu Items):

The system employs a linked list data structure to manage menu items. Each menu item is represented as a node in the linked list, containing information such as serial number, name, and price. This allows for efficient insertion, deletion, and traversal of menu items, enabling dynamic menu management by the restaurant owners.

## Execution

1. Program starts with welcome screen: Code and output is given below –

```
int main() {
    printf("\033[1;31m");
    printf("\t\t\t|-----|\n");
    printf("\t\t\t|          WELCOME TO THE FOOD DELIVERY SYSTEM          |\n");
    printf("\t\t\t|-----|\n\n");
    printf("\nGo to the Owner Login Page ->> ");
}
```

Fig: Code

```

|-----|
|          WELCOME TO THE FOOD DELIVERY SYSTEM          |
|-----|

Go to the Owner Login Page ->>
```

Fig: Execution

2. **Login:** By this function owner and customer both can login to the application. After login of the owner, he can create the menu item, delete the menu item, update the menu item. Here we demonstrate only creation of menu. To login the owner we use an user defined function name OwnerLogin(). After login of the customer, they can order their food as per need. To login the customer we use an user defined function CustomerLogin(). Code and output are given below----

```
// Function for owner login
void OwnerLogin() {
    printf("\033[0;32m");
    char name2[100], email2[25], phone2[15], password2[25];
    printf("\nEnter your name : ");
    fflush(stdin);
    gets(name2);
    printf("\nEnter your phone number : ");
    fflush(stdin);
    gets(phone2);
    printf("\nEnter your email : ");
    fflush(stdin);
    gets(email2);
    printf("\nEnter your password : ");
    fflush(stdin);
    gets(password2);
}
```

Fig: Code

## Execution

```
Enter your name : Owner
Enter your phone number : 123456789
Enter your email : owner@owner.com
Enter your password : ownerPassword
YOU HAVE LOGGED IN TO THE DELIVERY SYSTEM SUCCESSFULLY
```

Fig: Execution

```
// Function for customer login
void CustomerLogin() {
    printf("\033[0;34m");
    char name[100], address[500], email[25], phone[15], password[25];
    printf("\nEnter your name : ");
    fflush(stdin);
    gets(name);
    printf("\nEnter your full address : ");
    fflush(stdin);
    gets(address);
    printf("\nEnter your phone number : ");
    fflush(stdin);
    gets(phone);
    printf("\nEnter your email : ");
    fflush(stdin);
    gets(email);
    printf("\nEnter your password : ");
    fflush(stdin);
    gets(password);
    strcpy(name1, name);
    strcpy(address1, address);
    strcpy(email1, email);
    strcpy(phone1, phone);
}
```

Fig: Code

```
Go to the Customer Login Page ->>
Enter your name : Customer

Enter your full address : xyz street, city : xyz, xyz, WB, 123456

Enter your phone number : 9876543210

Enter your email : customer@customer.com

Enter your password : customerPassword
```

Fig: Execution

## Execution

3. **Menu Creation:** After login owner can update his menu item name and price by using TakingInput() function and after that this input pass through the Create() as argument. Create() function is used to create the menu dynamically. In this function we use linked list data structure to create the menu. Input code and output is given below—

```
void TakingInput() {
    printf("\033[0;35m");
    char name[50];
    int price;
    printf("\nEnter the item name : ");
    fflush(stdin);
    gets(name);
    printf("\nEnter the price : ");
    scanf("%d", &price);
    Create(name, price);
}

// Function for creating menu items
void Create(char name[50], int price) {
    struct MenuItem *newnode;
    newnode = (struct MenuItem *)malloc(sizeof(struct MenuItem));
    newnode->serial = serial;
    strcpy(newnode->name, name);
    newnode->price = price;
    newnode->next = NULL;
    serial++;
    if (head == NULL) {
        head = temp = newnode;
    } else {
        temp->next = newnode;
        temp = newnode;
    }
    int choice;
    printf("\nWill You Wanna Add More Item (1 for yes 0 for no ) : ");
    scanf("%d", &choice);
    if (choice == 1) {
        TakingInput();
    } else if (choice == 0) {
        return;
    }
}
```

Fig: Code

## Execution

```
Please Insert Your Menu Items -->>
Enter the item name : Food Item - 1
Enter the price : 50
Will You Wanna Add More Item (1 for yes 0 for no ) : 1
Enter the item name : Food - 2
Enter the price : 100
Will You Wanna Add More Item (1 for yes 0 for no ) : 1
Enter the item name : Food - 3
Enter the price : 40
Will You Wanna Add More Item (1 for yes 0 for no ) : 0
You Have Successfully Inserted Your Menu Items
```

Fig: Execution

4. **Food ordering ways:** In our food ordering application customer will get two choices. They can either order their food by choosing the hotel or they can order their food by choosing the food cart. To choose this option we use an user defined function choice(). If customer select by hotel option, then they will see the list of the hotel that available in our application. After selecting the hotel, they will see the menu of that particular hotel. To show the hotel list we use an user defined function by\_hotel(). If customer select by food option, then they will see a menu card. To do this we use a user defined function by\_food. The menu items are defined in an user defined function menu().

by\_hotel(). If customer select by food option, then they will see a menu card. To do this we use an user defined function by\_food. The menu items are defined in an user defined function menu().



## Execution

```
// Function for selecting the way of ordering
void choice() {
    printf("\033[0;36m");
    int n;
    printf("\n\nWe provide two types of orders ---> \n1.By Hotel\n2.By Food\n3.Exit\n");
    printf("\nENTER YOUR CHOICE--> ");
    scanf("%d", &n);
    switch (n) {
        case 1:
            ByHotel();
            break;
        case 2:
            ByFood();
            break;
        case 3:
            exit(1);
            break;
        default:
            printf("\nPlease press the right key ");
            main();
            break;
    }
}
```

Fig: Code

```
We provide two types of orders --->
1.By Hotel
2.By Food
3.Exit

ENTER YOUR CHOICE-->
```

Fig: Execution

## Execution

```
// Function for ordering by hotel
void ByHotel() {
    printf("\033[0;32m");
    int n, choice;
    printf("\n1.Hotel Shreyan\n2.Tasty\n3.Tris Planet\n");
    printf("4.Lets Eat\n5.Dhakeshwari Resturant\n6.Go to the choice \n7.Exit");
    printf("\nSelect the hotel name --> ");
    scanf("%d", &choice);
    switch (choice) {
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:
            printf("\n<-----THIS IS OUR MENU----->\n\n");
            Menu();
            Order();
            break;
        case 6:
            Choice();
            break;
        case 7:
            printf("\nTHANK YOU FOR VISITING US \n");
            exit(1);
            break;
        default:
            printf("\nPLEASE ENTER THE CORRECT CHOICE\n");
            ByHotel();
            break;
    }
    printf("\033[1;31m");
    printf("\nGO TO THE PAYMENT GATEWAY(press 1 ) ->> ");
    scanf("%d", &n);
    if (n == 1)
        Payment();
}
```

Fig: Code

```
1.Hotel Shreyan
2.Tasty
3.Tris Planet
4.Lets Eat
5.Dhakeshwari Resturant
6.Go to the choice
7.Exit
Select the hotel name --> |
```

Fig: Execution

## Execution

```
// Function for ordering by food
void ByFood() {
    printf("\033[0;32m");
    int n;
    printf("\n<-----THIS IS OUR MENU----->\n\n");
    Menu();
    Order();
    printf("\033[1;31m");
    printf("\n\nGO TO THE PAYMENT GATEWAY(press 1 ) ->> ");
    scanf("%d", &n);
    if (n == 1)
        Payment();
}
```

Fig: Code

```
<-----THIS IS OUR MENU----->

1          Food Item - 1          50
2          Food - 2              100
3          Food - 3              40
Select the item : |
```

Fig: Execution

```
// Function for displaying menu items
void Menu() {
    printf("\033[0;32m");
    struct MenuItem *temp1;
    temp1 = head;
    if (temp1 == NULL)
        printf("\nMenu Is Empty !!!!");
    while (temp1 != NULL) {
        printf("\n%d\t\t%s\t\t%d", temp1->serial, temp1->name, temp1->price);
        temp1 = temp1->next;
    }
}
```

Fig: Code

```
<-----THIS IS OUR MENU----->

1          Food Item - 1          50
2          Food - 2              100
3          Food - 3              40
Select the item : |
```

Fig: Execution

## Execution

5. **Order Confirmation:** After seeing the menu card customer can do their order confirm. They will select their order and confirm that order. To do that we use an user defined function Order(). Code and outputs are given below ---

```
// Function for ordering items
void Order() {
    printf("\033[0;34m");
    struct MenuItem *temp2;
    int Item, NumItem, Bill, ch;
    printf("\033[1;31m");
    printf("\nSelect the item : ");
    scanf("%d", &Item);
    printf("\033[1;32m");
    temp2 = head;
    while (temp2->serial != Item) {
        temp2 = temp2->next;
    }
    printf("\n How many %s do you want : ", temp2->name);
    scanf("%d", &NumItem);
    Bill = temp2->price * NumItem;
    printf("\nYour Bill For %d no of %s is : %d ", NumItem, temp2->name, Bill);
    total = total + Bill;
    printf("\nDo You Wanna Order More(1 for yes 0 for no) : ");
    scanf("%d", &ch);
    if (ch == 1) {
        Order();
    } else if (ch == 0) {
        return;
    }
}
```

Fig: Code

```
Select the item : 1

How many Food Item - 1 do you want : 2

Your Bill For 2 no of Food Item - 1 is : 100
Do You Wanna Order More(1 for yes 0 for no) : 1

Select the item : 2

How many Food - 2 do you want : 3

Your Bill For 3 no of Food - 2 is : 300
Do You Wanna Order More(1 for yes 0 for no) : 0

GO TO THE PAYMENT GATEWAY(press 1 ) ->> |
```

Fig: Execution

## Execution

6. **Payment:** After completing the order process customer will go to the payment gateway. To do this process we use and user defined function payment(). Here they will get both options online or offline payment. Here I am choosing online option. After choosing this option some online payment option will come and customer will have to select any one of them and complete their payment. The code and output of the function payment() is given below—

```
// Function for payment
void Payment() {
    printf("\033[0;37m");
    int n, a;
    printf("\n1.ONLINE PAYMENT\n2.CASH ON DELIVERY\n");
    printf("ENTER YOUR CHOICE--> ");
    scanf("%d", &n);
    switch (n) {
        case 1:
            printf("\n1.PAYTM\n2.PHONEPAY\n3.GPAY\n4.CARD PAY\n");
            printf("ENTER YOUR CHOICE--> ");
            scanf("%d", &a);
            switch (a) {
                case 1:
                case 2:
                case 3:
                case 4:
                    printf("\nTHANK YOU !!!!!!! YOU WILL GET YOUR DELICIOUS FOOD JUST AFTER 30-40 MINUTES \n\n");
                    break;
                default:
                    printf("PLEASE SELECT THE CORRECT OPTION\n");
                    Payment();
                    break;
            }
            break;
        case 2:
            printf("\nTHANK YOU !!!!!!! YOU WILL GET YOUR DELICIOUS FOOD JUST AFTER 30-40 MINUTES \n\n");
            break;
        default:
            printf("PLEASE SELECT THE CORRECT OPTION\n");
            Payment();
            break;
    }
    int c;
    printf("\033[1;31m");
    printf("WOULD YOU WANT TO GENERATE YOUR INVOICE ?(press 1 for yes 0 for no)--> ");
    scanf("%d", &c);
    if (c == 1) {
        Invoice();
    } else {
        printf("\n THANK YOU!!! PLEASE COME AGAIN...");
    }
}
```

Fig: Code

```
1.ONLINE PAYMENT
2.CASH ON DELIVERY
ENTER YOUR CHOICE--> 2|
```

```
1.PAYTM
2.PHONEPAY
3.GPAY
4.CARD PAY
ENTER YOUR CHOICE--> |
```

Fig: Execution

## Execution

**7. Generating Invoice:** After completing the payment process customer can generate their invoice. To generate this invoice() function is used. The code and output of invoice() function is given below—

```
// Function for generating invoice
void Invoice() {
    printf("\033[0;35m");
    int total2;
    printf("\n\n\t|-----|\n");
    printf("\t\t\t\t\tYOUR INVOICE\t\t\t\t\t\n");
    printf("\t|-----|\n\n");
    printf("\nName: %s\n", name1);
    printf("Address: %s\n", address1);
    printf("Phone: %s\n", phone1);
    printf("Email: %s\n", email1);
    int gst;
    gst = total + total * .18;
    total2 = gst;
    printf("\nYOUR TOTAL BILL IS--> %d (including 18 percent GST)\n", total2);
    printf("\n THANK YOU!!! PLEASE COME AGAIN...");
}
```

Fig: Code

Fig: Execution



## Graphical User Interface Application

### 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.

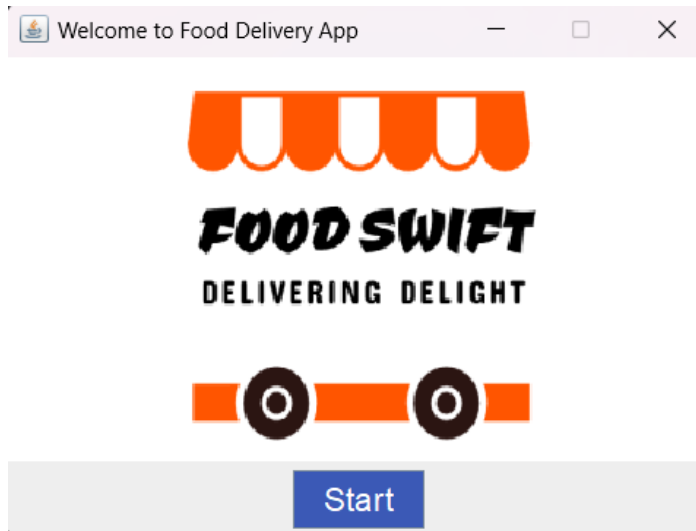
### Project Structure:

#### Classes Used:

- i) AddMenuItemScreen
- ii) CustomerLoginScreen
  - Authentication
  - CustomerLoginScreen
- iii) HotelBasedOrderingScreen
- iv) InvoiceScreen
- v) MainScreen
- vi) MenuLinkedList
  - MenuItem
  - MenuLinkedList
- vii) OrderingScreen
- viii) OrderOptionsScreen
- ix) OwnerLoginScreen'
  - a. OwnerAuthentication
  - b. OwnerLoginScreen
- x) PaymentGatewayScreen
- xi) WelcomePage (This is the entry point of the program).

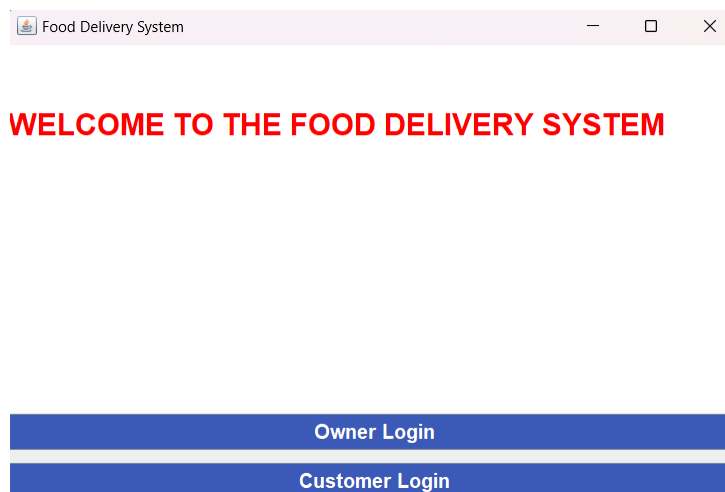
## Execution of GUI Program

### 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:

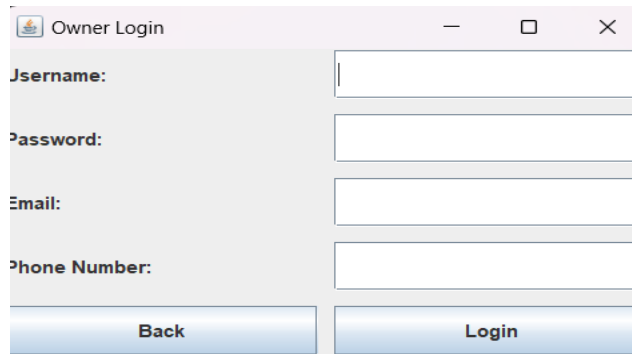


- This page show the welcome message and gives two options one for owner login and other for Customer login.
- Owner Login button starts Owner Login activity and disposes this one.
- Customer Login button starts the Customer Login activity and disposes this one.



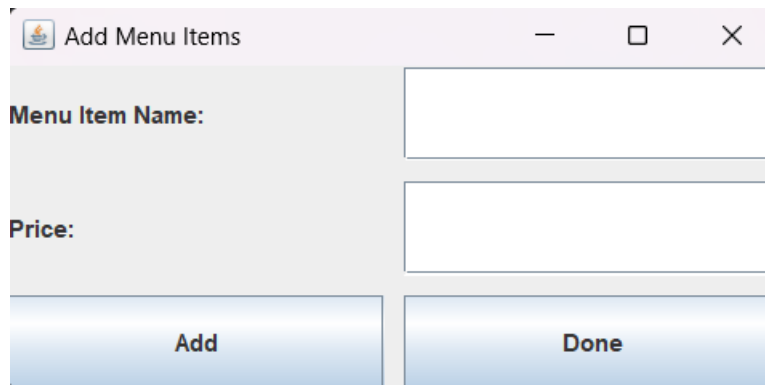
## Execution of GUI Program

### Owner Login:

A screenshot of a GUI window titled "Owner Login". The window has a light gray background and a title bar with standard Windows window controls (minimize, maximize, close). On the left side, there are four labels: "Username:", "Password:", "Email:", and "Phone Number:". To the right of each label is a white text input field. At the bottom of the window, there are two blue buttons with white text: "Back" on the left and "Login" on the right.

- 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.

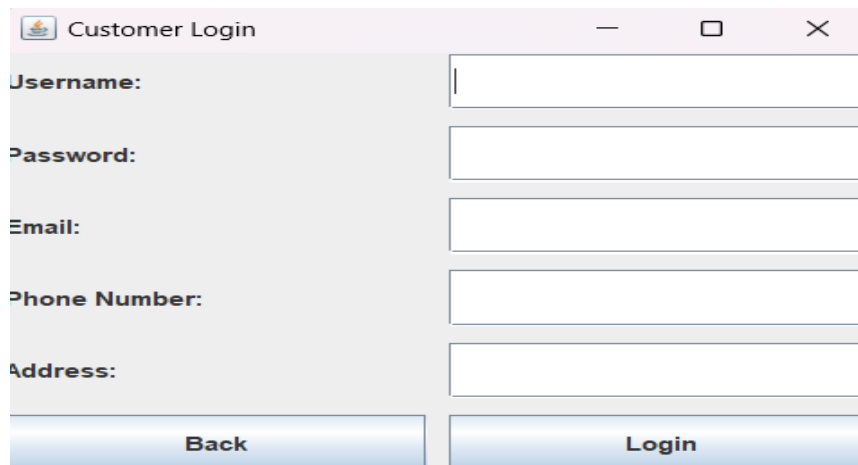
### Add Menu Items:

A screenshot of a GUI window titled "Add Menu Items". The window has a light gray background and a title bar with standard Windows window controls (minimize, maximize, close). On the left side, there are two labels: "Menu Item Name:" and "Price:". To the right of each label is a white text input field. At the bottom of the window, there are two blue buttons with white text: "Add" on the left and "Done" on the right.

- 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.

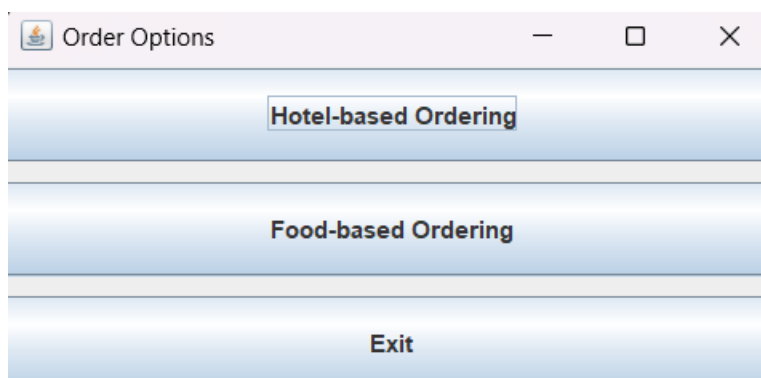
## Execution of GUI Program

### Customer Login:

A screenshot of a Java Swing window titled "Customer Login". The window has a light gray background and a title bar with standard Windows controls (minimize, maximize, close). It contains five text input fields stacked vertically, each preceded by a label: "Username:", "Password:", "Email:", "Phone Number:", and "Address:". Below the input fields are two buttons: "Back" on the left and "Login" on the right. The buttons have a blue gradient and a 3D effect.

- 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:

A screenshot of a Java Swing window titled "Order Options". The window has a light gray background and a title bar with standard Windows controls. It contains three large, rectangular buttons stacked vertically. The top button is labeled "Hotel-based Ordering", the middle button is labeled "Food-based Ordering", and the bottom button is labeled "Exit". All buttons have a blue gradient and a 3D effect.

- This screen shows all three options, 1<sup>st</sup> one starts the Hotel-Based Ordering activity , 2<sup>nd</sup> 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.

## Execution of GUI Program



### Ordering Screen:

The screenshot shows a window titled "Ordering Screen". At the top, there are two input fields: "Enter Serial No.:" and "Enter Quantity:", followed by an "Add" button. Below these fields is a list of menu items with three columns: "Serial No.:", "Name:", and "Price:". The items listed are:

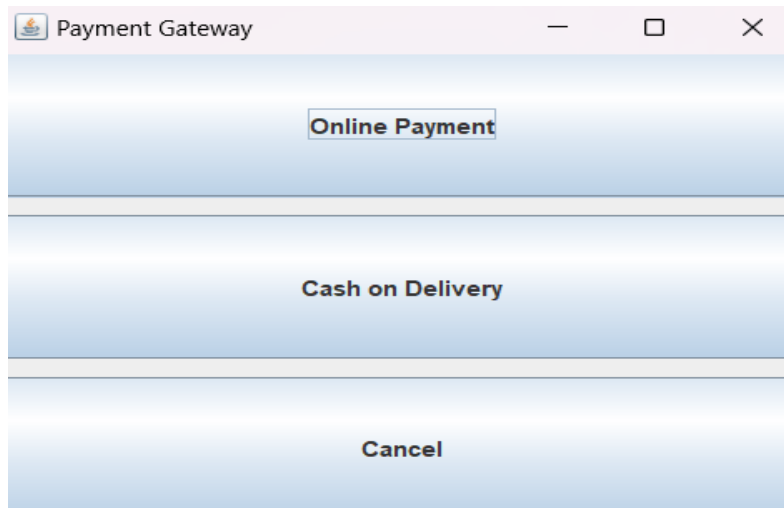
Serial No.:	Name:	Price:
1	Burger	40.0Rs.
2	Cake	400.0Rs.
3	Fried Rice	80.0Rs.

At the bottom of the window, there is a "Proceed to Payment" button.

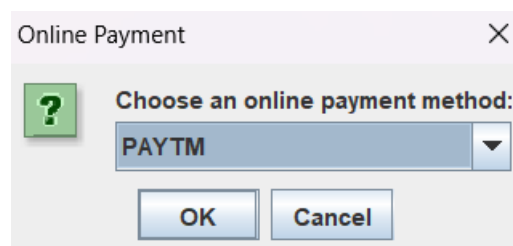
- On this screen user can enter serial no. of menu item and it's quantity to add order, if the owner has added a menu item file then it's content will be visible on the screen, else the default menu item file.
- Proceed to payment buttons starts the payment gateway screen and disposes this one.

## Execution of GUI Program

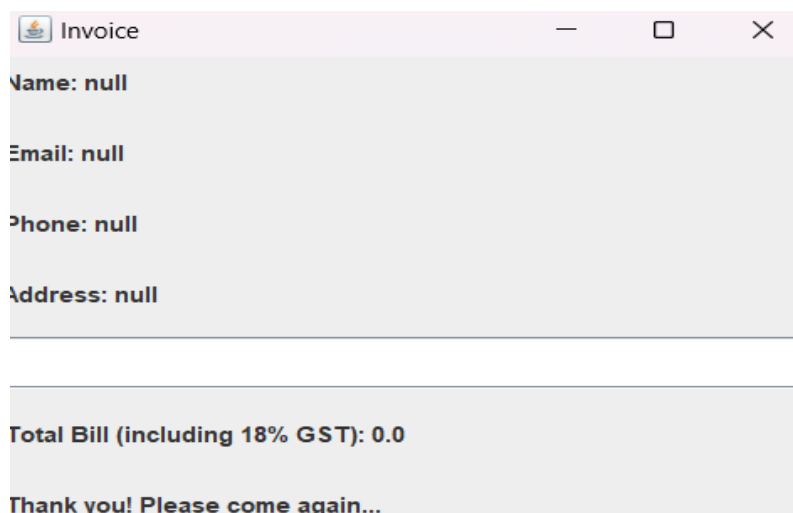
### Payment Gateway:



- 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.

## Data Structure Used

```
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;
        }
    }
}
```

## Data Structure Used

```
    }
    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.

### **Data Structure Used**

- **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.

## References

### References

Source Code: [https://github.com/anish-2903/Food\\_Delivery\\_App](https://github.com/anish-2903/Food_Delivery_App)