

# **FOOD ORDERING SYSTEM**

**A MINI-PROJECT REPORT**

**Submitted by**

**KAAVIYA G                      220701114**

**KEERTHANA S                220701124**

**In partial fulfillment of the award of the degree  
of**

**BACHELOR OF ENGINEERING  
IN  
COMPUTER SCIENCE AND ENGINEERING**



**RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI**

**An Autonomous Institute**

**Thandalam**

**Chennai - 602105**

**2023-2024**

## **BONAFIDE CERTIFICATE**

Certified that this Project report “ **FOOD ORDERING SYSTEM** ” is the bonafide work of “ **KAAVIYA G (220701114)** ” & “ **KEERTHANA S (220701124)** ” who carried out the project work under my supervision.

Submitted for the Practical Examination held on \_\_\_\_\_

**SIGNATURE**

**Dr.R.SABITHA**  
Professor and Academic Head,  
Computer Science and Engineering,  
Rajalakshmi Engineering College  
(Autonomous),  
Thandalam, Chennai - 602 105.

**SIGNATURE**

**Dr.G.Dharani Devi**  
Associate Professor ,  
Computer Science and Engineering,  
Rajalakshmi Engineering College  
(Autonomous),  
Thandalam, Chennai - 602 105.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

# ABSTRACT

This project entails the development of a sophisticated food ordering system using Java Swing for the graphical user interface and MySQL for robust database management, aimed at enhancing the efficiency of restaurant operations and improving customer experience.

The system starts with a secure login page where users authenticate their credentials. Upon successful login, users are navigated to a home page that displays an interactive menu, which is designed to be intuitive and user-friendly. This menu allows users to browse through various categories of food items, select their desired items, and specify quantities. The interface is designed to be dynamic, providing real-time feedback as users add or remove items from their order. The frontend ensures a seamless user experience by maintaining a session-based cart that updates in real time. Additionally, the frontend handles the generation of a detailed bill once the order is placed, displaying all selected items, their quantities, individual prices, and the total cost, including taxes and discounts. The use of Java Swing enables the creation of a responsive and visually appealing interface, enhancing the overall user experience.

The backend of our food ordering system is powered by MySQL, ensuring robust and scalable database management. The backend handles user authentication by verifying login credentials stored securely in the MySQL database. It manages the dynamic menu data, including food items, categories, prices, and availability, allowing for real-time updates and modifications. The backend also processes orders by storing selected items, quantities, and user details, ensuring accurate and efficient data management. It performs the necessary calculations to generate the final bill, including tax and discount computations. The backend uses Java Database Connectivity (JDBC) to facilitate seamless communication between the Java Swing application and the MySQL database, enabling efficient data transactions and real-time updates.

By implementing the Model-View-Controller (MVC) architecture, the backend achieves a clear separation of concerns, enhancing the system's maintainability and scalability. This robust backend infrastructure ensures the reliability and efficiency of the food ordering system, providing a solid foundation for the frontend to deliver a seamless user experience.

# **TABLE OF CONTENTS**

## **1. INTRODUCTION**

1.1. Introduction

1.2. Objectives

1.3. Modules

## **2. SURVEY OF TECHNOLOGIES**

2.1. Software Description

2.2. Languages

## **3. REQUIREMENTS AND ANALYSIS**

3.1. Requirements and Specification

3.2. Hardware and Software Requirements

3.3. Architecture Diagram

3.4. ER Daigram

3.5. Normalization

## **4. PROGRAM CODE**

## **5. RESULTS AND DISCUSSION**

## **6. CONCLUSION**

## **7. REFERENCES**

# CHAPTER 1

## INTRODUCTION

### 1.1 INTRODUCTION

The Food Ordering System project leverages Java Swings for its user interface and MySQL for its database management, providing a seamless and efficient platform for customers to browse, select, and order food. This system aims to enhance the dining experience by streamlining the ordering process, reducing wait times, and minimizing errors associated with manual order-taking. By integrating Java Swings, the application ensures a responsive and user-friendly interface, while MySQL offers robust and scalable data storage to manage customer information, menu items, and order histories. This comprehensive approach ensures a modern, reliable, and efficient solution for the food service industry.

### 1.2. OBJECTIVES

The primary objectives of the student election system are:

- ❖ **Efficiency:** Automate the food ordering process to streamline operations and reduce manual efforts for both users and restaurant staff.
- ❖ **Transparency:** Provide users with clear and detailed information about menu items, including ingredients, prices, and availability.
- ❖ **Accessibility:** Enable users to place orders conveniently from anywhere with internet access, using a user-friendly web interface or mobile application.
- ❖ **Security:** Implement robust security measures to protect user data, including personal information, using encryption and secure authentication methods.
- ❖ **User-Friendliness:** Design an intuitive and visually appealing interface that makes it easy for users to browse the menu, customize orders.
- ❖ **Data Accuracy:** Utilize database management techniques to ensure accurate recording and processing of orders, minimizing errors in order fulfilment.

## **1.3. MODULES**

The online food ordering system is composed of several key modules, each serving a specific function to ensure the smooth operation of the ordering process:

### **1. User Authentication Module**

- ❖ **Customer Registration:**
  - ❖ Allows users to create accounts by providing necessary details like name, email, and password.
- ❖ **Customer Login:**
  - ❖ Enables registered customers to securely log in to the system using their credentials.
- ❖ **Admin Login:**
  - ❖ Provides a login interface for restaurant owners to manage menu items and orders.

### **2. Menu Management Module**

- ❖ **Restaurant Menu Management:**
  - ❖ Enables restaurant owners to add, edit, and remove menu items along with their descriptions, prices, and availability.
- ❖ **Menu Display:**
  - ❖ Displays the restaurant menus to customers in a user-friendly format, allowing for easy browsing and selection.

### **3. Order Management Module**

- ❖ Allows customers to add items to their cart, adjust quantities, and remove items before checkout.

### **4. Admin Dashboard**

- ❖ Displays order history like Order Id, Order name, Quantity, Price according to the selected month for administrators to monitor.

# CHAPTER 2

## SURVEY OF TECHNOLOGIES

### 2.1 SOFTWARE DESCRIPTION

#### ARCHITECTURE DESCRIPTION

##### **Presentation Layer:**

Responsible for the graphical user interface (GUI) and user interaction. Implements Java Swing components for the desktop application interface. Utilizes Java for logic handling and event-driven programming. Provides a visually appealing and responsive interface for users to interact with. Handles user input, displays menu options, and facilitates order placement.

##### **Application Layer:**

Manages the business logic and flow of the food ordering system. Processes user actions, coordinates data flow, and interacts with the database. Implements logic for user authentication, menu management, order processing, and billing. Consists of Java classes and methods responsible for executing various system functionalities. Ensures smooth operation and seamless user experience throughout the application.

##### **Data Access Layer:**

Handles interactions with the MySQL database for data retrieval, storage, and manipulation. Executes SQL queries to fetch menu items, user information, and order details. Ensures data integrity and security by enforcing proper access controls and validation checks. Utilizes JDBC (Java Database Connectivity) for establishing connections and executing database operations. Manages the persistence and retrieval of data between the application layer and the database.

##### **Database Layer:**

Stores and organizes data related to users, menu items, orders, and billing information. Utilizes MySQL as the relational database management system for efficient data storage and retrieval. Maintains tables for storing user profiles, menu items, order details, and transaction records. Ensures data consistency and reliability by enforcing constraints and relationships between entities. Handles concurrent access and transaction management to maintain data integrity and consistency.

## **MODEL-VIEW-CONTROLLER (MVC) DESCRIPTION**

### **Model**

Represents the data entities and business logic of the food ordering system. Defines data structures for menu items, user profiles, orders, and billing information. Handles database interactions using JDBC (Java Database Connectivity) to retrieve, store, and manipulate data in the MySQL database. Validates user input, enforces business rules such as item availability and pricing, and maintains data integrity throughout the application. Includes classes and methods for managing user sessions, processing orders, and generating bills.

### **View**

Renders the graphical user interface (GUI) and presents data to users in the food ordering system. Implements Java Swing components for creating interactive desktop application interfaces. Utilizes layout managers and components such as buttons, labels, and text fields to design the user interface. Provides visual presentation and user experience enhancements through custom styling and design. Includes event listeners and handlers to capture user interactions and trigger appropriate actions in the controller.

### **Controller**

Acts as an intermediary between the model and the view in the food ordering system. Handles user interactions and translates them into actions on the model and view components. Implements routing logic to map user requests to specific functionalities within the application. Invokes methods in the model layer to process user actions, such as adding items to the cart or placing orders. Updates the view layer with changes in data or application state, ensuring a responsive and dynamic user interface. Coordinates the flow of data and control between the model and view, enforcing separation of concerns and promoting code maintainability.



## **COMPONENT DESCRIPTION**

### **User Authentication Component**

Manages user authentication and access control for the food ordering system. Validates user credentials provided during login against the database. Establishes and maintains secure user sessions to ensure authenticated access to system resources. Enforces access control policies to restrict unauthorized users from accessing sensitive functionalities.

### **Menu Management Component**

Handles the management of restaurant menus and menu items. Allows restaurant owners to add, edit, and remove menu items along with their descriptions, prices, and availability. Provides administrative tools for managing menu categories and organizing menu items for easy navigation. Ensures consistency and accuracy of menu information displayed to customers.

### **Order Management Component**

Facilitates the processing and management of customer orders. Manages the user's order, allowing customers to add, remove, or modify items before checkout. Generates detailed bills for customer orders, including itemized lists, quantities, prices, and total costs.

### **Admin Dashboard Component**

Offers a centralized interface for administrators to monitor and manage the order history.

Displays order id, item name, quantity and price for selected month.

## 2.2. LANGUAGES

**Java:** Java is the primary language used for developing the user interface and backend logic of the food ordering application. Java's platform independence and robust features make it suitable for creating a reliable and efficient application.

**SQL:** Structured Query Language (SQL) is used for managing the database of the food ordering application. SQL is essential for creating, updating, and querying the database to store and retrieve information about orders, customers, and menu items.

**JavaScript:** JavaScript is used for implementing interactive features and functionalities in the food ordering application. It is used for form validation, handling user inputs, and creating dynamic content on the web pages.

**NetBeans IDE:** NetBeans Integrated Development Environment (IDE) is used for developing the food ordering application. NetBeans provides a user-friendly interface and powerful tools for writing, compiling, and debugging Java code, making the development process more efficient.

**MySQL:** MySQL is used as the relational database management system (RDBMS) for storing and managing the data of the food ordering application. MySQL is known for its reliability, scalability, and performance, making it suitable for handling large volumes of data in the application.

# **CHAPTER 3**

## **REQUIREMENTS AND ANALYSIS**

### **3.1 REQUIREMENT SPECIFICATION**

#### **USER STORIES**

- As a customer, I want to browse the menu easily, so I can quickly find the items I want to order.
- As a customer, I want to create an account easily, so I can save my preferences and track my order history.
- As a customer, I want to view detailed descriptions and images of menu items, so I can make informed decisions before placing an order.
- As an administrator, I want to manage menu items easily (e.g., add, edit, delete), so I can keep the menu up-to-date with current offerings.
- As a customer, I want to review my selections before checkout, so I can ensure accuracy and completeness.
- As an administrator, I want to generate reports on order trends, sales performance, and customer feedback, so I can make data-driven decisions to improve the business.
- As a customer, I want to have the ability to rate and review menu items and overall service, so I can provide feedback to improve the system.
- As a customer, I want to have the option to reorder previous orders with one-click, so I can quickly purchase my favourite items without browsing the entire menu.

## **3.2 HARDWARE AND SOFTWARE REQUIREMENTS**

### **HARDWARRE REQUIREMENTS :**

#### **1. Server**

A dedicated server or cloud-based hosting capable of handling the expected traffic and data processing requirements of the system.

Suggested specifications:

- CPU: Multi-core processor (e.g., Intel Xeon, AMD Ryzen)
- RAM: At least 8GB (16GB recommended)
- Storage: SSD storage for faster data access
- Network: Stable internet connection with sufficient bandwidth

#### **2. Database Server**

A separate database server to store user accounts, menu information, order details, and other relevant data securely.

Suggested specifications:

- CPU: Multi-core processor
- RAM: At least 8GB (16GB recommended)
- Storage: SSD storage for optimal database performance
- Database Management System (DBMS): MySQL, PostgreSQL, or other suitable database software

#### **3. Network Infrastructure**

Reliable networking equipment including switches, routers, and firewalls to ensure secure communication between clients and servers.

Network security measures such as firewalls, intrusion detection/prevention systems, and regular security updates to protect against unauthorized access and data breaches.

#### **4. Client Devices**

Devices for users to access the system, including desktop computers, laptops, tablets, and smartphones.

Compatible web browsers for accessing the system's web interface, such as Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge.

# **SOFTWARE REQUIREMENTS :**

## **1. Operating System**

For the server: Linux distribution (e.g., Ubuntu Server, CentOS) or Windows Server, depending on the system administrator's preference and expertise.

For client devices: Compatibility with popular operating systems like Windows, macOS, iOS, Android, and Linux.

## **2. Web Server**

Apache, Nginx, or Microsoft Internet Information Services (IIS) to serve web pages and handle HTTP requests.

## **3. Programming Languages**

Server-side scripting language such as PHP, Python, or Node.js for dynamic web page generation and server-side logic.

Client-side scripting languages like JavaScript for interactive features.

## **4. Database Management System (DBMS)**

MySQL, PostgreSQL, MongoDB, or other relational or NoSQL databases for storing and managing data.

## **5. Security Software**

SSL/TLS certificates for encrypting data transmitted over the network.

Security protocols and best practices for user authentication, authorization, and data protection.

Regular security updates and patches to address potential vulnerabilities.

## **6. Other Dependencies**

Additional software libraries, frameworks, or modules required by the chosen programming languages and development tools.

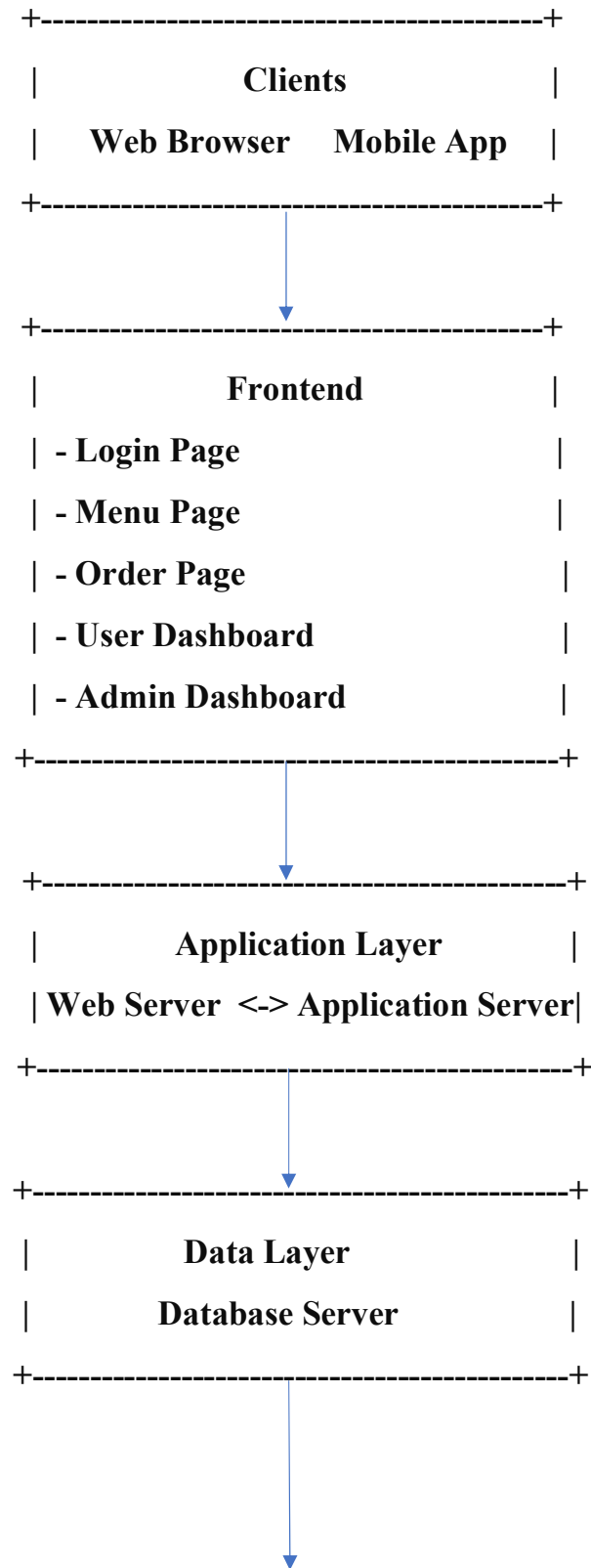
Version control system (e.g., Git) for managing code changes and collaboration among developers.

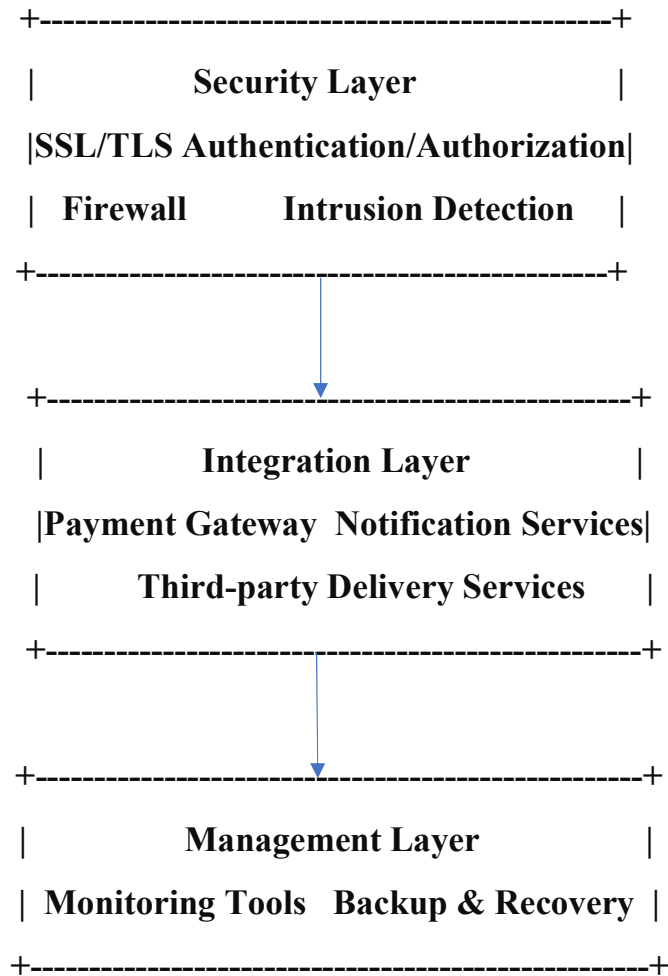
## **7. Monitoring and Management Tools**

Monitoring tools for tracking server performance, network traffic, and system health.

Backup and disaster recovery solutions to protect against data loss and system failures.

### 3.3 ARCHITECTURE DIAGRAM





This architecture ensures a modular and scalable system where the Java Swings client provides a responsive and intuitive user experience, the application server handles complex processing and business logic, and the MySQL database server manages data efficiently and securely. Together, these components create a cohesive online food ordering system that meets the needs of users and administrators alike.

### 3.4 ER DIAGRAM



This Entity-Relationship (ER) diagram represents an online food ordering system. Here is a brief explanation of its components:

#### 1. Entities and Attributes

- Customer: Has attributes like ID, name and password.
- Admin: Has attributes ID, password and name.
- Order Details: Associated with attributes like fname, order\_no, price, Quantity and cust\_id.



- Food: Has attributes like F\_name, Food\_id, Description and Price.

## **2. Relationships**

Places:

- A user places an order.
- An order is placed by one user.

Includes:

- An order includes multiple order items.
- An order item is part of one order.

Belongs to:

- An order item belongs to a menu item.
- A menu item can be part of multiple order items.

Categorizes:

- A menu item belongs to a category.
- A category can include multiple menu items.

## **3. Cardinality and Participation**

User and Order:

- A user can place zero or more orders.
- An order must be placed by one user.

Order and OrderItem:

- An order must include one or more order items.
- An order item must belong to one order.

OrderItem and MenuItem:

- An order item must reference one menu item.
- A menu item can be referenced by zero or more order items.

## 3.5 NORMALISATION

### 1NF (First Normal Form)

In the first normal form, the table should have atomic values, and each column should contain only one value per record.

ORDERID	ITEM_NAME	QUANTITY	PRICE
1	Fried rice	2	120
1	Buttermilk	1	45
2	Meals	1	190
2	Noodles	2	130
3	Tea	1	40
3	Coffee	2	30

### NF (Second Normal Form)

In the second normal form, the table should be in 1NF, and all non-key attributes should be fully functionally dependent on the primary key. To achieve this, we need to ensure that no partial dependency exists (i.e., no non-key attribute should depend on a part of a composite primary key).

To transform the 1NF table into 2NF, we need to remove any partial dependencies. We can do this by creating separate tables for orders and items.

#### Orders Table

ORDERID	ORDER_DATE	CUSTOMER_ID
1	2024-05-25	101
2	2024-05-26	102
3	2024-05-27	103

### Order\_Items Table

ORDERID	ITEM_ID	QUANTITY
1	1	2
1	2	1
2	3	1
2	4	2
3	1	1
3	3	2

### Items Table

ITEM_ID	ITEM_NAME	PRICE
1	Fried rice	120
2	Buttermilk	45
3	Meals	190
4	Noodles	130

### 3NF (Third Normal Form)

In the third normal form, the table should be in 2NF, and all the attributes should be functionally dependent only on the primary key (no transitive dependency).

In this case, the Items table already satisfies 3NF since item\_name and price depend only on the primary key item\_id.

Similarly, the Orders and Order\_Items tables do not have any transitive dependencies and are also in 3NF.

**Orders Table**

ORDERID	ORDER_DATE	CUSTOMER_ID
1	2024-05-25	101
2	2024-05-26	102
3	2024-05-27	103

**Order\_Items Table**

ORDERID	ITEM_ID	QUANTITY
1	1	2
1	2	1
2	3	1
2	4	2
3	1	1
3	3	2

**Items Table**

ITEM_ID	ITEM_NAME	PRICE
1	Fried rice	120
2	Buttermilk	45
3	Meals	190
4	Noodles	130

By separating the data into these three tables, we ensure that our database is normalized to 3NF, eliminating redundancy and ensuring data integrity.

## CHAPTER 4

### PROGRAM CODE

#### LOGIN PAGE:

```
package gpa;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.swing.JOptionPane;

public class gframe extends javax.swing.JFrame {

    ResultSet rs;

    PreparedStatement pst;

    public gframe() {
        initComponents();
        this.setResizable(false);
    }

    @SuppressWarnings("unchecked")
    private void initComponents() {

        IDT = new javax.swing.JTextField();
        PT = new javax.swing.JPasswordField();
        CloseButton = new javax.swing.JButton();
        LoginButton = new javax.swing.JButton();
        TC = new javax.swing.JComboBox<>();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```

        setMaximumSize(new java.awt.Dimension(1500, 700));
        setMinimumSize(new java.awt.Dimension(1500, 700));
        setPreferredSize(new java.awt.Dimension(1500, 700));
        getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());
        getContentPane().add(jLabel1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(283, 127, 161, 35));
        jPanel1.setBackground(new java.awt.Color(255, 255, 255));
        jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
        jPanel3.setBackground(new java.awt.Color(0, 153, 153));
        javax.swing.GroupLayout jPanel3Layout = new
javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 560, Short.MAX_VALUE)
        );
        jPanel3Layout.setVerticalGroup(
jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 0, Short.MAX_VALUE)
        );
        jPanel1.add(jPanel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(0,
490, 560, -1));
        jLabel3.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
        jLabel3.setForeground(new java.awt.Color(0, 153, 153));
        jLabel3.setText("PASSWORD");

```

```

jPanel1.add(jLabel3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 220, 100, 40));
jLabel4.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
jLabel4.setForeground(new java.awt.Color(0, 153, 153));
jLabel4.setText("TYPE");
jPanel1.add(jLabel4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(60, 280, 70, 40));
IDT.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
IDT.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 153, 153)));
IDT.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        IDTActionPerformed(evt);
    } });
jPanel1.add(IDT, new org.netbeans.lib.awtextra.AbsoluteConstraints(160,
150, 280, 40));
PT.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
PT.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 153, 153)));
PT.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        PTActionPerformed(evt);
    } });
jPanel1.add(PT, new org.netbeans.lib.awtextra.AbsoluteConstraints(160,
220, 280, 40));
CloseButton.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
CloseButton.setForeground(new java.awt.Color(0, 153, 153));
CloseButton.setText("Signup");

```

```

CloseButton.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(255, 255, 255)));

    CloseButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            CloseButtonActionPerformed(evt);
        }
    });

jPanel1.add(CloseButton, new
org.netbeans.lib.awtextra.AbsoluteConstraints(310, 420, 70, 30));

LoginButton.setBackground(new java.awt.Color(0, 153, 153));
LoginButton.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
LoginButton.setForeground(new java.awt.Color(255, 255, 255));
LoginButton.setText("Login");
LoginButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        LoginButtonActionPerformed(evt);
    }
});

jPanel1.add(LoginButton, new
org.netbeans.lib.awtextra.AbsoluteConstraints(160, 350, 160, 50));

jLabel5.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jLabel5.setForeground(new java.awt.Color(102, 102, 102));
jLabel5.setText("    Don't have an account ?");
jPanel1.add(jLabel5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(100, 420, 210, 30));

jPanel2.setBackground(new java.awt.Color(0, 153, 153));
jPanel2.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jLabel2.setBackground(new java.awt.Color(0, 153, 153));
jLabel2.setFont(new java.awt.Font("Cambria", 1, 48)); // NOI18N
jLabel2.setForeground(new java.awt.Color(255, 255, 255));
jLabel2.setText("        LOGIN ");

```



```

        jPanel2.add(jLabel2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(70, 30, 300, 50));
        jPanel1.add(jPanel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(0,
0, 470, 120));
        jLabel6.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
        jLabel6.setForeground(new java.awt.Color(0, 153, 153));
        jLabel6.setText("NAME");
        jPanel1.add(jLabel6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 150, 70, 40));
        TC.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
        TC.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] {
"User", "Admin" }));
        TC.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 153, 153)));
        TC.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                TCActionPerformed(evt);
            }
        });
        jPanel1.add(TC, new org.netbeans.lib.awtextra.AbsoluteConstraints(160,
290, 280, 40));
        getContentPane().add(jPanel1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(550, 110, 470, 500));
        pack();
        setLocationRelativeTo(null);
    }

    private void closeButtonActionPerformed(java.awt.event.ActionEvent evt) {
        this.hide();
        SignupFrame signup=new SignupFrame();
    }

```

```

        signup.show();
    }
    private void LoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String id= IDT.getText();
        String pw=PT.getText();
        String type=(String)TC.getSelectedItem();
        if(id.equals("user") && pw.equals("user") && type.equals("User")){
            this.hide();
            HomePage h=new HomePage();
            h.show();
        }
        if(id.equals("admin") && pw.equals("admin") && type.equals("Admin")){
            this.hide();
            Admin a=new Admin();
            a.show();
        } }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new gframe().setVisible(true);
            } });
    }
    private javax.swing.JButton CloseButton;
    private javax.swing.JTextField IDT;
    private javax.swing.JButton LoginButton;
    private javax.swing.JPasswordField PT;
    private javax.swing.JComboBox<String> TC;
}

```

## ADMIN PAGE:

```
package gpa;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Vector;

public class Admin extends javax.swing.JFrame {

    public Admin() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        monthComboBox = new javax.swing.JComboBox();
        jPanel6 = new javax.swing.JPanel();
        jSeparator1 = new javax.swing.JSeparator();
        jSeparator3 = new javax.swing.JSeparator();
        jSeparator4 = new javax.swing.JSeparator();
        jLabel2 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        table = new javax.swing.JTable();
        jComboBox1 = new javax.swing.JComboBox<>();
        jButton1 = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();
    }
}
```

```

jPanel1.setBackground(new java.awt.Color(0, 153, 153));
jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jLabel1.setBackground(new java.awt.Color(0, 153, 153));
jLabel1.setFont(new java.awt.Font("Cambria", 1, 36)); // NOI18N
jLabel1.setForeground(new java.awt.Color(255, 255, 255));
jLabel1.setText("ADMIN PAGE");

jPanel1.add(jLabel1,neworg.netbeans.lib.awtextra.AbsoluteConstraints(1370,
40, 260, 50));
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setBackground(new java.awt.Color(255, 255, 255));
getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());
monthComboBox.setBackground(new java.awt.Color(255, 255, 255));
monthComboBox.setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());
jPanel6.setBackground(new java.awt.Color(0, 153, 153));
jPanel6.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jPanel6.add(jSeparator1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(-14, 0, 280, 0));
jPanel6.add(jSeparator3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(-10, 93, 270, 0));
jPanel6.add(jSeparator4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(160, 90, -1, -1));
jLabel2.setFont(new java.awt.Font("Cambria", 1, 36)); // NOI18N
jLabel2.setForeground(new java.awt.Color(255, 255, 255));
jLabel2.setText("ADMIN PAGE");
jPanel6.add(jLabel2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(630, 20, 250, 50));

```

```

monthComboBox.add(jPanel6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1550, 110));
table.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
table.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        },
    new String [] {
        "Order ID", "OrderDate", "Items", "Quantity", "Price"
    } ) {
        Class[] types = new Class [] {
            java.lang.Integer.class, java.lang.Integer.class, java.lang.String.class,
java.lang.Integer.class, java.lang.Double.class
        };
        public Class getColumnClass(int columnIndex) {
            return types [columnIndex];
        };
    });
table.setRowHeight(30);
jScrollPane1.setViewportView(table);
monthComboBox.add(jScrollPane1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(150, 210, 1250, 460));
jComboBox1.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jComboBox1.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December" }));
jComboBox1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jComboBox1ActionPerformed(evt);
    }
});

```

```

        monthComboBox.add(jComboBox1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(720, 140, 170, 40));
        jButton1.setBackground(new java.awt.Color(0, 153, 153));
        jButton1.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
        jButton1.setForeground(new java.awt.Color(255, 255, 255));
        jButton1.setText("Back");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        monthComboBox.add(jButton1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(710, 700, 150, 40));
        jLabel3.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
        jLabel3.setForeground(new java.awt.Color(0, 102, 102));
        jLabel3.setText("Select Month :");
        monthComboBox.add(jLabel3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(590, 140, 120, 40));
        getContentPane().add(monthComboBox, new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 1550, 830));
        pack();
        setLocationRelativeTo(null);    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        this.hide();
        JFrame g=new JFrame();
        g.show();
    }
    private void JcomboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
        int selectedMonth = jComboBox1.getSelectedIndex() + 1;
        fetchOrderDetailsForMonth(selectedMonth);
    }

```

```

    }
    private void fetchOrderDetailsForMonth(int month) {
try {
    Connection con;

    Class.forName("com.mysql.cj.jdbc.Driver");
con=java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/pfo",
"root", "Rec124.edu.in");

    System.out.println("Connected to database.");

    DefaultTableModel model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);

    String query = "SELECT o.OrderID, o.OrderDate, i.ItemName, i.Quantity,
i.Price " +
        "FROM OrdersTable o " +
        "JOIN OrderItems i ON o.OrderID = i.OrderID " + "WHERE
MONTH(o.OrderDate) = ?";

    PreparedStatement pst = con.prepareStatement(query);
    pst.setInt(1, month);
    ResultSet rs = pst.executeQuery();
    while (rs.next()) {
        Vector<Object> row = new Vector<>();
        row.add(rs.getInt("OrderID"));
row.add(rs.getDate("OrderDate").toLocalDate().format(DateTimeFormatter.ofP
attern("dd-MM-yyyy"))));
        row.add(rs.getString("ItemName"));
        row.add(rs.getInt("Quantity"));
        row.add(rs.getDouble("Price"));
        model.addRow(row);
    }
    con.close();

```

```

    } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error: " + e.getMessage(),
"Database Error", JOptionPane.ERROR_MESSAGE);
    }
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Admin().setVisible(true);
        }
    });
}

private javax.swing.JButton jButton1;
private javax.swing.JComboBox<String> jComboBox1;
private javax.swing.JPanel monthComboBox;
private javax.swing.JTable table;
}

```

## **HOME PAGE:**

```

import java.io.IOException;
import java.sql.Date;
import java.time.LocalDate;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JCheckBox;
import javax.swing.JLabel;
import javax.swing.JSpinner;
import javax.swing.JOptionPane;
import java.sql.PreparedStatement;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;

```



```
import java.util.logging.Logger;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.Connection;
import java.util.ArrayList;
import java.util.List;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class HomePage extends javax.swing.JFrame {

    private Object[] C;

    public HomePage() {
        initComponents();
        this.setResizable(false);
    }

    @SuppressWarnings("unchecked")
    private void initComponents() {

        jButton2 = new javax.swing.JButton();
        OrderButton = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        ListArea = new javax.swing.JTextArea();
        jButton1 = new javax.swing.JButton();
        C1 = new javax.swing.JCheckBox();
        C4 = new javax.swing.JCheckBox();
        C6 = new javax.swing.JCheckBox();
        C3 = new javax.swing.JCheckBox();
        C5 = new javax.swing.JCheckBox();
        C2 = new javax.swing.JCheckBox();
        S2 = new javax.swing.JSpinner();
        S3 = new javax.swing.JSpinner();
    }
}
```

```

S4 = new javax.swing.JSpinner();
S5 = new javax.swing.JSpinner();
S6 = new javax.swing.JSpinner();
S1 = new javax.swing.JSpinner();
C7 = new javax.swing.JCheckBox();
S7 = new javax.swing.JSpinner();
S8 = new javax.swing.JSpinner();
C8 = new javax.swing.JCheckBox();
C9 = new javax.swing.JCheckBox();
C10 = new javax.swing.JCheckBox();
S9 = new javax.swing.JSpinner();
S10 = new javax.swing.JSpinner();
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());
    jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
    jPanel2.setBackground(new java.awt.Color(255, 255, 255));
    jPanel2.setMaximumSize(new java.awt.Dimension(1700, 800));
    jPanel2.setMinimumSize(new java.awt.Dimension(1700, 800));
    jPanel2.setPreferredSize(new java.awt.Dimension(1700, 800));
    jPanel2.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
    jPanel3.setBackground(new java.awt.Color(0, 153, 153));
    jPanel3.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
    jLabel1.setFont(new java.awt.Font("Cambria", 1, 36)); // NOI18N
    jLabel1.setForeground(new java.awt.Color(255, 255, 255));
    jLabel1.setText("FOOD ORDER APP");
    jPanel3.add(jLabel1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(590, 10, 320, 40));
    jButton2.setBackground(new java.awt.Color(0, 153, 153));

```

```

jButton2.setFont(new java.awt.Font("Cambria", 1, 14)); // NOI18N
jButton2.setForeground(new java.awt.Color(255, 255, 255));
jButton2.setText("History");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});
jPanel3.add(jButton2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(1190, 20, 110, 30));
jPanel2.add(jPanel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(0,
0, 1550, 70));
jPanel4.setBackground(new java.awt.Color(255, 255, 255));
jPanel4.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.B
orderFactory.createLineBorder(new java.awt.Color(0, 153, 153), 3), "Menu",
javax.swing.border.TitledBorder.CENTER,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Cambria", 1, 24), new java.awt.Color(0, 153, 153))); // NOI18N
jPanel4.setPreferredSize(new java.awt.Dimension(1700, 900));
jPanel4.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
jPanel7.setBackground(new java.awt.Color(255, 255, 255));
jPanel7.setBorder(javax.swing.BorderFactory.createTitledBorder(new
javax.swing.border.LineBorder(new java.awt.Color(0, 153, 153), 2, true),
"Order Details",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Cambria", 1, 18), new java.awt.Color(0, 153, 153)));
jPanel7.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
OrderButton.setBackground(new java.awt.Color(0, 153, 153));
OrderButton.setFont(new java.awt.Font("Cambria", 1, 18));

```

```

OrderButton.setForeground(new java.awt.Color(255, 255, 255));
OrderButton.setText(" Order ");
OrderButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        OrderButtonActionPerformed(evt);
    } });
jPanel7.add(OrderButton, new
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 550, 110, 40));
ListArea.setColumns(20);
ListArea.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N
ListArea.setRows(5);
jScrollPane1.setViewportView(ListArea);
jPanel7.add(jScrollPane1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(20, 40, 380, 490));
jButton1.setBackground(new java.awt.Color(0, 153, 153));
jButton1.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jButton1.setForeground(new java.awt.Color(255, 255, 255));
jButton1.setText("Back");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    } });
jPanel7.add(jButton1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(240, 550, 120, 40));
jPanel4.add(jPanel7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(1070, 40, 420, 610));
jPanel9.setBackground(new java.awt.Color(255, 255, 255));
jPanel9.setBorder(javax.swing.BorderFactory.createTitledBorder(new
javax.swing.border.LineBorder(new java.awt.Color(0, 153, 153), 2, true),

```

```

"MAIN COURSE",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Cambria", 1, 14), new java.awt.Color(0, 153, 153))); // NOI18N
    jPanel9.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
    C1.setBackground(new java.awt.Color(255, 255, 255));
    C1.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
    C1.setText("Biryani");
    C1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            C1ActionPerformed(evt);
        }
    });
    jPanel9.add(C1, new org.netbeans.lib.awtextra.AbsoluteConstraints(80, 30,
90, 30));
    C4.setBackground(new java.awt.Color(255, 255, 255));
    C4.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
    C4.setText("Naan with");
    C4.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            C4ActionPerformed(evt);
        }
    });
    jPanel9.add(C4, new org.netbeans.lib.awtextra.AbsoluteConstraints(80,
300, 120, 30));
    C6.setBackground(new java.awt.Color(255, 255, 255));
    C6.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
    C6.setText("Meals");
    C6.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(255, 255, 255)));
    C6.addActionListener(new java.awt.event.ActionListener() {

```

```

        public void actionPerformed(java.awt.event.ActionEvent evt) {
            C6ActionPerformed(evt);
        }
    });

    jPanel9.add(C6, new org.netbeans.lib.awtextra.AbsoluteConstraints(670,
30, 90, 30));

    C3.setBackground(new java.awt.Color(255, 255, 255));
    C3.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
    C3.setText("Fried Rice");
    C3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            C3ActionPerformed(evt);
        }
    });

    jPanel9.add(C3, new org.netbeans.lib.awtextra.AbsoluteConstraints(460,
30, -1, -1));

    C5.setBackground(new java.awt.Color(255, 255, 255));
    C5.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
    C5.setText("Chapathi Kurma(2 Pcs) ");
    C5.addActionListener(new java.awt.event.ActionListener() {
        public
void actionPerformed(java.awt.event.ActionEvent evt) {
            C5ActionPerformed(evt); }
    });

    jPanel9.add(C5, new org.netbeans.lib.awtextra.AbsoluteConstraints(810,
40, -1, 20));

    jLabel6.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gpa/Biriyani (2).png"))); //
NOI18N

    jPanel9.add(jLabel6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(20, 70, 200, 130));

```

```

        jLabel2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gpa/Naan.png"))); // NOI18N
        jPanel9.add(jLabel2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(20, 370, 210, 130));
        jLabel7.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gpa/Noodles.png"))); //
NOI18N
        jPanel9.add(jLabel7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(240, 70, 170, 130));
        jLabel8.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gpa/Chapati.png"))); //
NOI18N
        jPanel9.add(jLabel8, new
org.netbeans.lib.awtextra.AbsoluteConstraints(820, 70, 180, 130));
        C2.setBackground(new java.awt.Color(255, 255, 255));
        C2.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
        C2.setText("Noodles");
        C2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                C2ActionPerformed(evt);
            }
        });
        jPanel9.add(C2, new org.netbeans.lib.awtextra.AbsoluteConstraints(270,
30, -1, -1));
        jLabel12.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gpa/Fried Rice.png"))); //
NOI18N
        jPanel9.add(jLabel12, new
org.netbeans.lib.awtextra.AbsoluteConstraints(420, 70, 190, 130));

```

```

jLabel9.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gpa/Meals.png"))); // NOI18N
jPanel9.add(jLabel9, new
org.netbeans.lib.awtextra.AbsoluteConstraints(610, 70, 200, 130));
jLabel13.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jLabel13.setText("₹ 130");
jPanel9.add(jLabel13, new
org.netbeans.lib.awtextra.AbsoluteConstraints(270, 220, 50, 30));
jLabel14.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jLabel14.setText("₹ 270");
jPanel9.add(jLabel14, new
org.netbeans.lib.awtextra.AbsoluteConstraints(90, 520, 50, 30));
jLabel15.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jLabel15.setText("₹ 210");
jPanel9.add(jLabel15, new
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 220, 50, 30));
jLabel16.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jLabel16.setText("₹ 90");
jPanel9.add(jLabel16, new
org.netbeans.lib.awtextra.AbsoluteConstraints(860, 220, 40, 30));
jLabel17.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jLabel17.setText("₹ 190");
jPanel9.add(jLabel17, new
org.netbeans.lib.awtextra.AbsoluteConstraints(660, 220, 50, 30));
jLabel22.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jLabel22.setText("₹ 120");
jPanel9.add(jLabel22, new
org.netbeans.lib.awtextra.AbsoluteConstraints(470, 220, 50, 30));
S2.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N

```



```

jPanel9.add(S2, new org.netbeans.lib.awtextra.AbsoluteConstraints(330,
220, 60, 30));
S3.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jPanel9.add(S3, new org.netbeans.lib.awtextra.AbsoluteConstraints(530,
220, 60, 30));
S4.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jPanel9.add(S4, new org.netbeans.lib.awtextra.AbsoluteConstraints(150,
520, 60, 30));
S5.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jPanel9.add(S5, new org.netbeans.lib.awtextra.AbsoluteConstraints(920,
220, 60, 30));
S6.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jPanel9.add(S6, new org.netbeans.lib.awtextra.AbsoluteConstraints(720,
220, 60, 30));
S1.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jPanel9.add(S1, new org.netbeans.lib.awtextra.AbsoluteConstraints(140,
220, 60, 30));
jLabel23.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
jLabel23.setText("Panner Butter Masala");
jPanel9.add(jLabel23, new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 330, 190, 30));
jLabel5.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gpa/Tea.png"))); // NOI18N
jPanel9.add(jLabel5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(240, 370, 180, 130));
C7.setBackground(new java.awt.Color(255, 255, 255));
C7.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
C7.setText("Tea");
C7.addActionListener(new java.awt.event.ActionListener() {

```

```

        public void actionPerformed(java.awt.event.ActionEvent evt) {
            C7ActionPerformed(evt);
        }
    });

    jPanel9.add(C7, new org.netbeans.lib.awtextra.AbsoluteConstraints(290,
320, 70, 20));

    S7.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
    jPanel9.add(S7, new org.netbeans.lib.awtextra.AbsoluteConstraints(340,
520, 60, 30));

    jLabel3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gpa/Coffee.png"))); //
NOI18N
    jPanel9.add(jLabel3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(430, 370, 180, 140));

    S8.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
    jPanel9.add(S8, new org.netbeans.lib.awtextra.AbsoluteConstraints(540,
520, 60, 30));

    C8.setBackground(new java.awt.Color(255, 255, 255));
    C8.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
    C8.setText("Filter Coffee");
    C8.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            C8ActionPerformed(evt);
        }
    });

    jPanel9.add(C8, new org.netbeans.lib.awtextra.AbsoluteConstraints(450,
320, 130, -1));

    jLabel18.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
    jLabel18.setText("₹ 40");
    jPanel9.add(jLabel18, new
org.netbeans.lib.awtextra.AbsoluteConstraints(290, 520, 50, 30));

```

```

jLabel19.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jLabel19.setText("₹ 30");
jPanel9.add(jLabel19, new
org.netbeans.lib.awtextra.AbsoluteConstraints(480, 520, 50, 30));
jLabel10.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gpa/Ice cream.png"))); //
NOI18N
jPanel9.add(jLabel10, new
org.netbeans.lib.awtextra.AbsoluteConstraints(630, 370, 180, 130));
C9.setBackground(new java.awt.Color(255, 255, 255));
C9.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
C9.setText("Vanilla Icecream");
C9.addActionListener(new java.awt.event.ActionListener() {
    Public void ctionPerformed(java.awt.event.ActionEvent evt) {
        C9ActionPerformed(evt);
    }
});
jPanel9.add(C9, new org.netbeans.lib.awtextra.AbsoluteConstraints(630,
320, 170, -1));
jLabel11.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gpa/Buttermilk.png"))); //
NOI18N
jPanel9.add(jLabel11, new
org.netbeans.lib.awtextra.AbsoluteConstraints(830, 350, 180, 150));
C10.setBackground(new java.awt.Color(255, 255, 255));
C10.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N
C10.setText("Buttermilk");
C10.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        C10ActionPerformed(evt);
    }
});

```

```

        } });

jPanel9.add(C10, new org.netbeans.lib.awtextra.AbsoluteConstraints(850,
320, 130, -1));

jLabel20.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jLabel20.setText("₹ 60");
jPanel9.add(jLabel20, new
org.netbeans.lib.awtextra.AbsoluteConstraints(680, 520, 50, 30));

S9.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jPanel9.add(S9, new org.netbeans.lib.awtextra.AbsoluteConstraints(730,
520, 60, 30));

S10.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jPanel9.add(S10, new org.netbeans.lib.awtextra.AbsoluteConstraints(930,
520, 60, 30));

jLabel21.setFont(new java.awt.Font("Cambria", 0, 18)); // NOI18N
jLabel21.setText("₹ 45");
jPanel9.add(jLabel21, new
org.netbeans.lib.awtextra.AbsoluteConstraints(880, 520, 50, 30));

jPanel4.add(jPanel9, new
org.netbeans.lib.awtextra.AbsoluteConstraints(20, 40, 1040, 610));

jPanel2.add(jPanel4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 80, 1510, 690));

jPanel1.add(jPanel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(0,
0, 1550, 780));

getContentPane().add(jPanel1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, -1, -1));

pack();
}

private void OrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {

```

```

Connection con;
Class.forName("com.mysql.cj.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/pfo", "root",
"Rec124.edu.in");
System.out.println("Connected to database.");
String[] foodItems = {"Biryani", "Noodles", "Fried rice", "Naan with
Panner Butter Masala",
    "Chapathi with kurma(2 pcs)", "Meals", "Tea", "Coffee", "Vanilla Ice
cream", "Buttermilk"};
double[] foodPrices = {210.0, 130.0, 120.0, 270.0, 90.0, 190.0, 40.0, 30.0,
60.0, 45.0};
JCheckBox[] C = {C1, C2, C3, C4, C5, C6, C7, C8, C9, C10}; //
Assuming C1 to C10 are your checkbox variables
JSpinner[] S = {S1, S2, S3, S4, S5, S6, S7, S8, S9, S10}; // Assuming S1
to S10 are your spinner variables
Date orderDate = Date.valueOf(LocalDate.now());
double total = 0.00;
for (int i = 0; i < foodItems.length; i++) {
    if (C[i].isSelected()) {
        int quantity = (int) S[i].getValue();
        double price = foodPrices[i];
        total += price * quantity;
    } }
int orderId = insertOrder(con, orderDate, total);
for (int i = 0; i < foodItems.length; i++) {
    if (C[i].isSelected()) {
        String itemName = foodItems[i];
        int quantity = (int) S[i].getValue();
        double price = foodPrices[i];

```

```

        insertOrderItem(con, orderId, itemName, quantity, price);}}
    con.close();
} catch (ClassNotFoundException | SQLException e) {
    e.printStackTrace();
} updateOrderDetails(); }

Private void Button1ActionPerformed(java.awt.event.ActionEvent evt) {
    this.hide();
    JFrame g=new JFrame();
    g.show(); }

private static void insertOrderItem(Connection con, int orderId, String
itemName, int quantity, double price) throws SQLException {
    String query = "INSERT INTO OrderItems (OrderID, ItemName, Quantity,
Price) VALUES (?, ?, ?, ?)";
    PreparedStatement pst = con.prepareStatement(query);
    pst.setInt(1, orderId);
    pst.setString(2, itemName);
    pst.setInt(3, quantity);
    pst.setDouble(4, price);
    int rowsInserted = pst.executeUpdate();
    if (rowsInserted > 0) {
        System.out.println("Order for " + itemName + " inserted successfully.");
    }
}

private static int insertOrder(Connection con, Date orderDate, double total)
throws SQLException {
    String insertOrderQuery = "INSERT INTO OrdersTable (OrderDate,
TotalPrice) VALUES (?, ?)";
    PreparedStatement pst = con.prepareStatement(insertOrderQuery,
Statement.RETURN_GENERATED_KEYS);
    pst.setDate(1, orderDate);

```

```

pst.setDouble(2, total);
pst.executeUpdate();
ResultSet rs = pst.getGeneratedKeys();
if (rs.next()) {
    return rs.getInt(1);
} return -1;}

public void updateOrderDetails() {
    StringBuilder orderDetails = new StringBuilder();
    String[] foodItems = {"Biriyani", "Noodles", "Fried rice", "Naan",
        "Chapathi", "Meals", "Tea", "Coffee", "Ice cream", "Buttermilk"};
    double[] foodPrices = {210.00, 130.00, 120.00, 270.00, 90.00, 190.00,
40.00, 30.00, 60.00, 45.00};

    LocalDateTime now = LocalDateTime.now();
    DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern("yyyy-
MM-dd HH:mm:ss");
    String formattedDate = now.format(dateFormatter);
    JCheckBox[] C = {C1, C2, C3, C4, C5, C6, C7, C8, C9, C10}; //
Assuming C1 to C10 are your checkbox variables
    JSpinner[] S = {S1, S2, S3, S4, S5, S6, S7, S8, S9, S10};
    double totalamount = 0.00;
    double total=0.00;
    ListArea.setText(ListArea.getText() + "\n");
    ListArea.setText(ListArea.getText() + "\n");
    ListArea.setText(ListArea.getText() + "
TASTE
BITES    \n");
    ListArea.setText(ListArea.getText() + "
DATE & TIME: " +
formattedDate + "\n");
    ListArea.setText(ListArea.getText() + "\n");

```

```

        ListArea.setText(ListArea.getText() + "-----\n");
        ListArea.setText(ListArea.getText() + "    Item Name " + "\t" + "
Item Price" + "\t" + "    Quantity" + "\t" + "    Price" + "\n");
        ListArea.setText(ListArea.getText() + "-----\n");

        ListArea.setText(ListArea.getText() + "\n");
        for (int i = 0; i < foodItems.length; i++) {
            if (C[i].isSelected()) {
                String itemName = foodItems[i];
                int quantity = (int) S[i].getValue();
                double price = foodPrices[i];
                total = price * quantity;
                totalamount+=total;

                ListArea.setText(ListArea.getText() + "    "+ itemName + "\t" + "
"+ price + "\t" + "    "+ quantity + "\t" + "    "+total+ "\n");
            } }

        ListArea.setText(ListArea.getText() + "\n");
        ListArea.setText(ListArea.getText() + "-----\n");

        ListArea.setText(ListArea.getText() + "
" + "Total Amount : " + "    " + totalamount + "\n");

        ListArea.setText(ListArea.getText() + "-----\n");

        ListArea.setText(ListArea.getText() + "\n");
        ListArea.setText(ListArea.getText() + "\n");
        ListArea.setText(ListArea.getText() + "*****THANKS
FOR ORDERING*****");

        JOptionPane.showMessageDialog(null,"Order Sucessful");    }

```

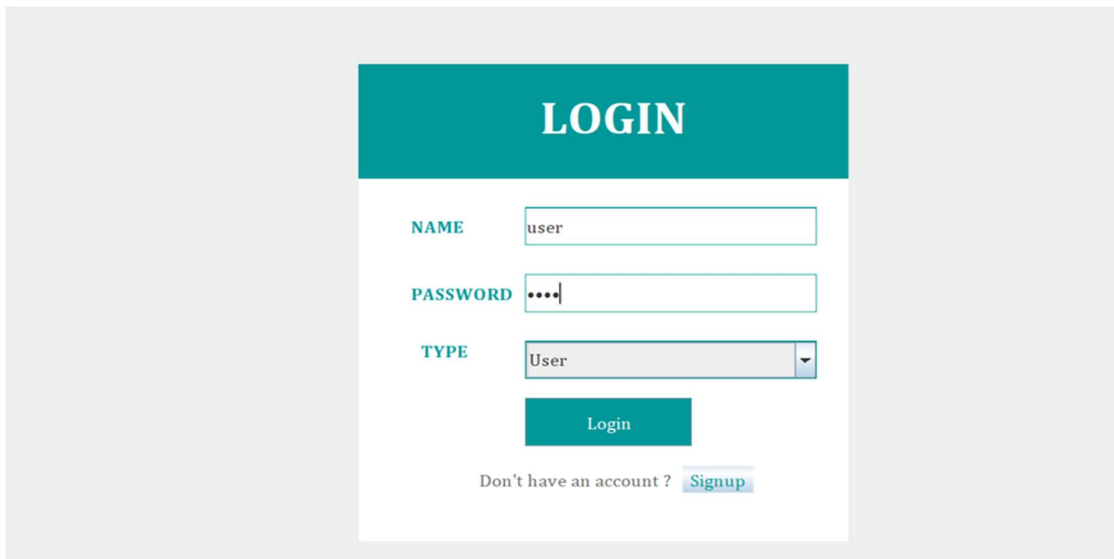


```
public static void main(String args[]) {  
    java.awt.EventQueue.invokeLater(new Runnable() {  
        public void run() {  
            new HomePage().setVisible(true); } }); }  
private javax.swing.JCheckBox C1;  
private javax.swing.JCheckBox C10;  
private javax.swing.JCheckBox C2;  
private javax.swing.JCheckBox C3;  
private javax.swing.JCheckBox C4;  
private javax.swing.JCheckBox C5;  
private javax.swing.JCheckBox C6;  
private javax.swing.JCheckBox C7;  
private javax.swing.JCheckBox C8;  
private javax.swing.JCheckBox C9;  
private javax.swing.JTextArea ListArea;  
private javax.swing.JButton OrderButton;  
private javax.swing.JSpinner S1;  
private javax.swing.JSpinner S10;  
private javax.swing.JSpinner S2;  
private javax.swing.JSpinner S3;  
private javax.swing.JSpinner S4;  
private javax.swing.JSpinner S5;  
private javax.swing.JSpinner S6;  
private javax.swing.JSpinner S7;  
private javax.swing.JSpinner S8;  
private javax.swing.JSpinner S9;  
private javax.swing.JButton jButton1;  
private javax.swing.JButton jButton2;  
}
```

## CHAPTER 5

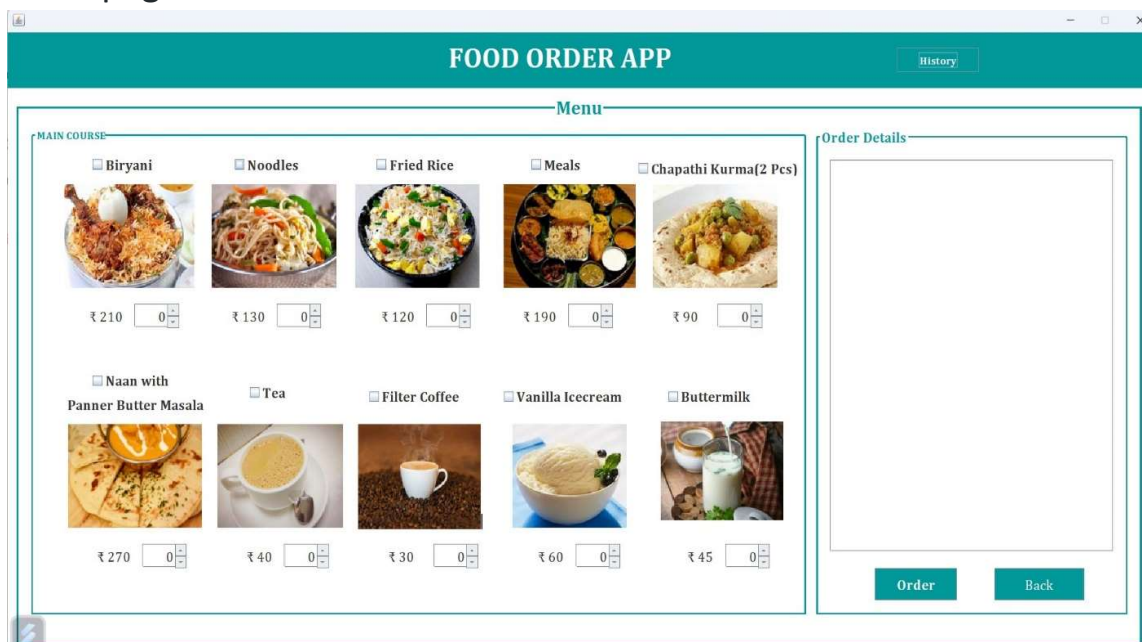
### RESULTS AND DISCUSSION

**Step 1:** The system initially displays the login page, prompting the user to log in.



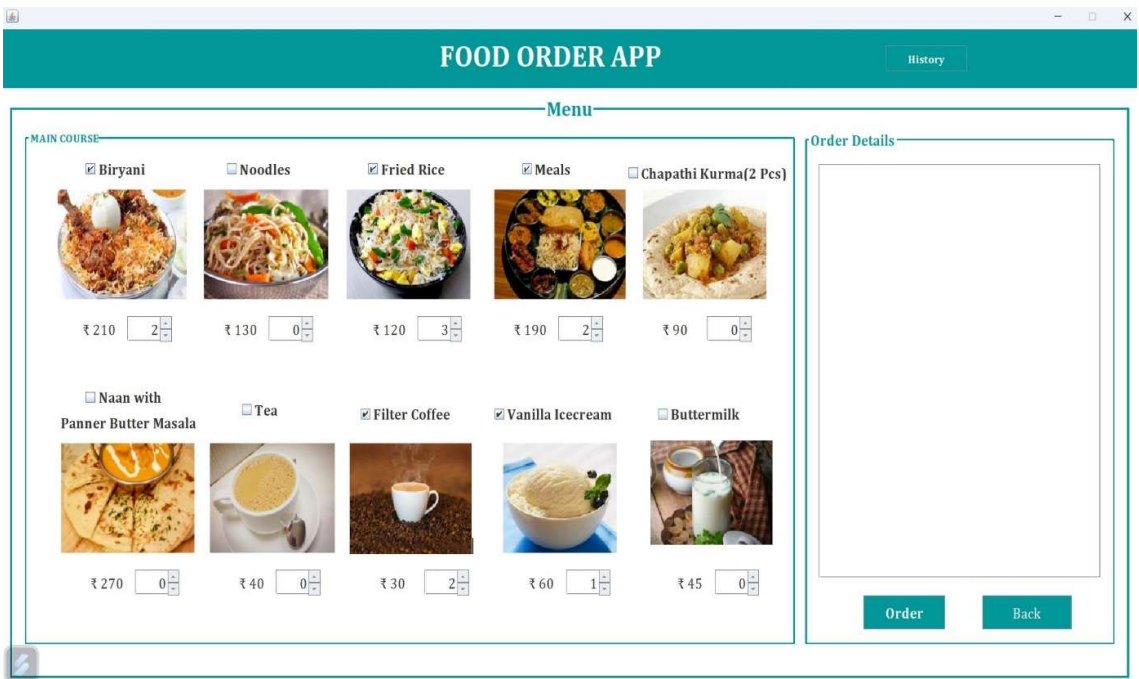
The login page features a teal header with the word "LOGIN" in white. Below the header, there are three input fields: "NAME" with the text "user", "PASSWORD" with four dots, and "TYPE" with a dropdown menu showing "User". A teal "Login" button is positioned below these fields. At the bottom, there is a link "Don't have an account ? Signup" in blue text.

**Step 2:** Upon clicking the login button, the user is directed to the homepage.

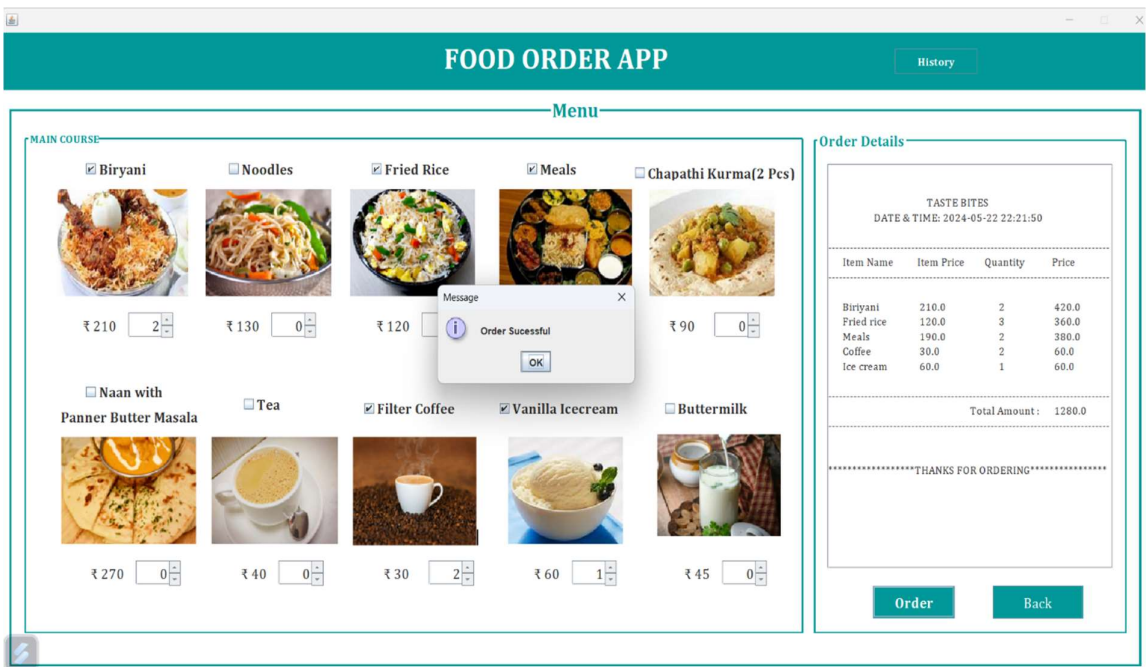


The homepage of the "FOOD ORDER APP" is displayed. It has a teal header with the app name and a "History" button. Below the header is a "Menu" section with a grid of food items. Each item includes a checkbox, a name, an image, and a price with a quantity selector. The items are: Biryani (₹ 210), Noodles (₹ 130), Fried Rice (₹ 120), Meals (₹ 190), Chapathi Kurma(2 Pcs) (₹ 90), Naan with Panner Butter Masala (₹ 270), Tea (₹ 40), Filter Coffee (₹ 30), Vanilla Icecream (₹ 60), and Buttermilk (₹ 45). To the right of the menu is an "Order Details" section with a large empty box and "Order" and "Back" buttons at the bottom.

**Step 3:** The user selects the food items they wish to order from the menu.



**Step 4:** After clicking the order button, the system shows the ordered receipt.



**Step 5:** By clicking the back button, the user returns to the login page, allowing an admin to log in by entering their credentials.

LOGIN

NAME

admin

PASSWORD

•••••

TYPE

Admin

Login

Don't have an account ?

Signup

**Step 6:** Upon successful admin login, the system displays the admin page.

ADMIN PAGE

Select Month :

January

Order ID	OrderDate	Items	Quantity	Price
----------	-----------	-------	----------	-------

Back

**Step 7:** The admin selects the month for which they wish to view order details.

ADMIN PAGE

Select Month :

January

January

February

March

April

May

June

July

August

Order ID	OrderDate		Quantity	Price
----------	-----------	--	----------	-------

Back

**Step 8:** The system displays the order details for the selected month.

ADMIN PAGE

Select Month : February

Order ID	OrderDate	Items	Quantity	Price
1	11-02-2024	Buttermilk	2	45
1	11-02-2024	Chapathi with kurma(2 pcs)	2	90
1	11-02-2024	Fried rice	2	120
1	11-02-2024	Vanilla Ice cream	2	60
4	23-02-2024	Biriyani	8	210
4	23-02-2024	Chapathi with kurma(2 pcs)	1	90
4	23-02-2024	Fried rice	3	120
4	23-02-2024	Vanilla Ice cream	5	60
8	07-02-2024	Buttermilk	1	45
8	07-02-2024	Fried rice	6	120
8	07-02-2024	Meals	2	190
8	07-02-2024	Noodles	3	130
8	07-02-2024	Vanilla Ice cream	5	60

Back

# CONCLUSION

The Online Food Ordering System project successfully delivers a robust and user-friendly platform for managing food orders online. Utilizing Java Swings for the frontend and MySQL for the backend, this system ensures a seamless and efficient experience from browsing the menu to placing orders and tracking deliveries.

## Key Achievements:

- **User-Friendly Interface:** The system features an intuitive GUI designed with Java Swings, making it easy for customers, restaurant staff, and administrators to navigate and perform their respective tasks.
- **Seamless Order Processing:** The integration of MySQL for data management ensures that order details, menu items, and user information are efficiently handled, providing real-time updates and accurate order tracking.
- **Efficient Management:** The application server manages the core business logic, facilitating smooth operations such as order processing, menu updates, and inventory management, enhancing overall efficiency.
- **Secure Transactions:** Implementing secure data transmission protocols and user authentication safeguards the integrity of the system, ensuring that customer information and transaction details are protected.
- **Comprehensive Documentation:** Detailed project documentation, including setup instructions, usage guidelines, and maintenance procedures, ensures that users and developers can easily deploy and manage the system.

This project demonstrates the potential of integrated software solutions in modernizing traditional food ordering processes, offering convenience, speed, and reliability. Future enhancements could include additional features such as advanced data analytics, support for multiple payment gateways, and integration with third-party delivery services.

Overall, the Online Food Ordering System stands as a testament to how technology can transform the food service industry, improving operational efficiency, customer satisfaction, and business growth.

## REFERENCES

### WEBSITES :

1. NetBeans IDE - <https://netbeans.apache.org/kb/docs/index.html>
2. Java Swing Tutorial – <https://www.geeksforgeeks.org/java-swing/>
3. NetBeansTutorials-  
<https://netbeans.org/kb/docs/java/quickstart.html>
4. ConnectingJavawithMySQLUsingJDBC-  
<https://www.javacodegeeks.com/2012/10/jdbc-tutorial-connecting-to-mysql-database-using-java.html>
5. MySQL - <https://www.w3schools.com/MySQL/default.asp>

### BOOKS :

1. **"Java: The Complete Reference" by Herbert Schildt:**  
Comprehensive guide to Java programming, covering core features, APIs, and frameworks.
2. **"Java Swing" by Marc Loy, Robert Eckstein, and Dave Wood:**  
Detailed guide to creating GUIs with Java Swing toolkit.
3. **"NetBeans: The Definitive Guide" by Tim Boudreau, Jesse Glick, Simeon Greene, and Jack J. Woehr:** Complete guide to using NetBeans IDE for Java development.