RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM-602 105

CS23333 OOPS Using Java

Laboratory Record Note Book

Name :

Year / Branch / Section :

University Register No. : ..

College Roll No. : .

Semester : .

Academic Year : .

RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution

BONAFIDE CERTIFICATE

Name: ……

Academic Year: …… **Semester:** …… **Branch:** ……

Register No.

Certified that this is the bonafide record of work done by the above student in the.....Laboratory during the academic year 2025- 2026

Signature of Faculty in-charge

Submitted for the Practical Examination held on……

Internal Examiner **External Examiner**

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	GITHUB QR
1		I/O, Data Types, Operators	
2		Control Structures	
3		Arrays	
4		Strings	
5		Classes & Objects	
6		Inheritance	
7		Interface	
8		Exceptions	
9		Collections	
10		Collections	
11		Project	
12		Lambda	

FOOD DELIVERY SYSTEM MANAGEMENT

A MINI-PROJECT REPORT

Submitted by

ALVIN B 240701034
ARJUN S 240701048

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI
NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project “FOOD DELIVERY SYSTEM” is the Bonafide work of “Alvin B, Arjun S who carried out the project work under my supervision.

SIGNATURE

Deepa S

ASSISTANT PROFESSOR SG

Dept. of Computer Science and Engg,

Rajalakshmi Engineering College

Chennai

This mini project report is submitted for the viva voce examination to be held on

_____

INTERNAL EXAMINER EXTERNAL EXAMINER

ABSTRACT

The **Food Delivery System is** an online platform designed to simplify the process of ordering and delivering food from restaurants to customers. It bridges the gap between customers, restaurants, and delivery personnel through a unified and efficient digital system. The platform allows users to browsemenus, place orders, make online payments, and track deliveries in real time. Restaurants can manage their menus, update order statuses, and monitor sales, while delivery agents can efficiently receive and fulfill delivery requests. By automating and digitizing the traditionalfoodorderingprocess, the system reduces human errors, saves time, and enhances user convenience. It also provides analytical insights to help restaurants improve their operations. Overall, the Food Delivery System offers a reliable, scalable, and user-friendly solutionthat meets the growing demand for online food ordering services in today's fast-paced lifestyle.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman MR. S. **MEGANATHAN** and the chairperson DR. M. THANGAM **MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head of the Department **Dr. E.M. MALATHY** and our Deputy Head of The Department **Dr. J. MANORANJINI** for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **Deepa S,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. Alvin B

2 Arjun S

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The **Food Delivery System** is an online platform that simplifies ordering and delivering food from restaurants to customers. It enables users to browse menus, place orders, and track deliveries in real time. The project aims to overcome issues of manual food ordering and delays. It s scope includes developing a reliable, user-friendly, and secure system for customers, restaurants, and delivery agents. The main objective is to automate the process, ensuring convenience, efficiency, and accuracy in food delivery services.

1.2 SCOPE OF THE WORK

The project develops a web or mobile-based food delivery platform connecting customers, restaurants, and delivery staff.It handles **user registration, menu browsing, order placement, payments, and real-time tracking.Admins can manage** users, **restaurants, and order histories** efficiently.The system is scalable, supporting future features like offers, reviews, and analytics.It can serve both small local businesses and large food delivery enterprises.

1s PROBLEM STATEMENT

In traditional food ordering systems, customers need to visit restaurants physicaliy or place orders through phone calls, which often leads to communication errors, long waiting times, and limited menu visibility. Managing orders manually also creates inefficiencies for restaurants and delivery personnel. There is a lack of a centralized system that can automate the entire process—from order placement to delivery tracking—while ensuring convenience and transparency for all parties involved. Therefore, there is a need fora digital food delivery system that provides real-time updates, accurate order management, and a seamless experience for customers, restaurants, and delivery agents.

1.4 AIM AND OBJECTIVES OF THE PROJECT

To design and develop an efficient, user-friendly, and reliable **Food Delivery** System that automates the process of ordering, managing, and delivering food through a digital platform.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

> Processor: Intel Core i3 or higher

RAM: Minimum 4 GB

Hard Disk: 250 GB or more

Display: 1024 < 768 resolution or higher

Internet Connection: Required for online access

2.2 SOFTWARE SPECIFICATIONS

Operating System: Windows / Linux / macOS

Frontend: Javawing

Backend: MySQL

Database: MySQL

CHAPTER 3

MODULE DESCRIPTION

MODULE DESCRIPTION

The Food Delivery System is designed with multiple functional modules that work together to ensure smooth and efficient operations. Each module is responsible for handling specific tasks, such as managing users, restaurants, orders, and deliveries. The integration of these modules helps maintain accuracy, real-time communication, and seamless data flow between customers, restaurants, delivery agents, and administrators.

3.1 User Module

The User Module allows customers to interact with the system effectively. It provides features for new users to register and existing users to log in securely. Customers can browse restaurant menus, add food items to their cart, place online orders, and make secure payments through integrated gateways. After placing an order, users can track their delivery status in real time and receive notifications about order updates. This module also maintains order history and supports features like ratings, feedback, and profile management to enhance the user experience.

3.2 Restaurant Module

The Restaurant Module is designed for restaurants to manage their operations efficiently. It enables restaurant owners or managers to log in, update menus, set food availability, and modify prices. Restaurants can view and confirm incoming orders, update order statuses (e.g., accepted, in preparation, ready for delivery), and monitor overall sales performance. The module also generates reports related to sales, customer feedback, and popular food items, helping restaurants make data-driven decisions and improve their service quality.

3.3 Delivery Module

The Delivery Module focuses on managing the delivery process and ensuring timely order fulfilment. Once an order is ready, it is assigned to an available delivery agent. The delivery personnel can log iii to view **their** assigned orders, check pickup and delivery addresses, and update delivery status in real time. The module also includes features such as route optimization, GPS tracking, and delivery confirmation to ensure that customers receive their food promptly and accurately. It maintains records of completed deliveries and supports performance tracking for delivery staff.

3.4 Admin Module

The Admin Module serves as the control centre of the en’tire system. Administrators can manage all users, restaurants, and delivery agents through this module. They can monitor ongoing operations, verify new restaurant registrations, and oversee order and payment records. The admin has access to generate reports, analyze system performance, and handle issues such as user complaints or technical errors. Security management, database maintenance, and access control are also managed under this module to ensure smooth and secure operation of the system.

CHAPTER 4

CODING

```
DROP DATABASE IF EXISTS food_delivery_db,
CREATE DATABASE food_delivery_db;
 USE food_delivery_db;

-- Table 1: users (for registration  and login)
CREATE TABLE  users (
    user  id INT PRIMARY KEY AUTO  INCREMENT,
    name VARCHAR(100)  NOT NULL,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(100)  NOT NULL,
    phone VARCHAR(15),
    address TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP


-- Table 2: food      roducts (stores all food items with calories)
CREATE TABLE food       roducts (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100)  NOT NULL,
     description  TEXT,
    price DECIMAL(10,2) NOT NULL,
    calories INT NOT NULL,
    category  VARCHAR(50),
    available BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP


-- Table 3: orders (stores placed orders)
CREATE TABLE orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL,
    delivery_address TEXT,
    payment_method VARCHAR(50),
    total_amount  DECIMAL(10,2),
    order_items  TEXT,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

-- SOUTH INDIAN FOOD MENU:

INSERT INTO food roducts (name, description, price, calories, category) VALUES
('Masala Dosa', 'Crispy rice crepe with spiced potato filling', 80.00, 350, 'South Indian'),
('Plain Dosa', 'Thin and crispy rice crepe', 60.00, 250, 'South Indian'),
('Idli (3 pcs)', 'Steamed rice cakes with sambar and chutney', 50.00, 180, 'South Indian'),
('Vada (2 pcs)', 'Deep fried lentil donuts with chutney', 40.00, 220, 'South Indian'),
('Rava Dosa', 'Crispy semolina crepe', 90.00, 300, 'South Indian'),
('Ghee Roast Dosa', 'Dosa roasted with pure ghee', 120.00, 420, 'South Indian'),
('Onion Uttapam', 'Thick pancake topped with onions', 85.00, 280, 'South Indian'),
('Medu Vada', 'Crispy lentil fritters', 45.00, 200, 'South Indian'),
('Sambar Rice', 'Rice mixed with lentil soup', 70.00, 320, 'South Indian'),
('Curd Rice', 'Rice with yogurt and tempering', 60.00, 250, 'South Indian'),
('Pongal', 'Rice and lentil comfort food', **65.00, 300,** 'South Indian'),
('Upma', 'Savory semolina porridge', **55.00,** 240, 'South Indian'),
('Pesarattu', 'Green gram dosa with upma filling', 95.00, 340, 'South Indian'),
('Bisi Bele Bath', 'Spicy rice with vegetables and lentils', 85.00, 380, 'South Indian'),
('Filter Coffee', 'Traditional South Indian filter coffee', 30.00, 50, 'South Indian');

-- NORTH INDIAN FOOD MENU:

INSERT INTO food roducts (name, description, price, calories, category) VALUES
('Butter Chicken', 'Creamy tomato-based chicken curry', 280.00, 520, 'North Indian'),
('Chicken Biryani', 'Aromatic rice dish with spiced chicken', 250.00, 650, 'North Indian'),
('Paneer Butter Masala', 'Cottage cheese in rich tomato gravy', 220.00, 450, 'North Indian'),
('DaI Makhani', 'Creamy black lentils with butter', 180.00, 380, 'North Indian'),
('Palak Paneer', 'Cottage cheese in spinach gravy', 200.00, 320, 'North Indian'),
('Chole Bhature', 'Spicy chickpeas with fried bread', **150.00, 550,** 'North Indian'),
('Rogan Josh', 'Aromatic lamb curry', 320.00, 480, 'North Indian'),
('Naan (2 pcs)', 'Soft leavened flatbread', 40.00, 260, 'North Indian');

-- PIZZA MENU:

INSERT INTO food roducts (name, description, price, calories, category) VALUES
('Margherita Pizza', 'Classic cheese pizza with tornato sauce', 250.00, 720, 'Pizza'),
('Pepperoni Pizza', 'Pizza with pepperoni and cheese', 320.00, 840, 'Pizza'),
('Paneer Tikka Pizza', 'Indian-style pizza with paneer tikka', 300.00, 780, 'Pizza'),
('Veggie Supreme Pizza', 'Pizza loaded with fresh vegetables', 280.00, 680, 'Pizza'),
('BBQ Chicken Pizza', 'Pizza with BBQ chicken and onions', **350.00,** 880, 'Pizza');

-- BURGER MENU:

INSERT INTO food roducts (name, description, price, calories, category) VALUES
('Chicken Burger', 'Grilled chicken burger with lettuce', **150.00, 550,** 'Burger'),
('Veggie Burger', 'Vegetarian burger with fresh vegetables', 120.00, 400, 'Burger'),
('Paneer Burger', 'Indian-style paneer patty burger', 140.00, 480, 'Burger'),
('Cheese Burger', 'Classic burger with cheese slice', 130.00, 520, 'Burger'),
('Chicken Tikka Burger', 'Burger with spiced chicken tikka', 170.00, 580, 'Burger');

-- CHINESE FOOD MENU:

INSERT INTO food roducts (name, description, price, calories, category) VALUES
('Veg Fried Rice', 'Stir-fried rice with vegetables', **150.00,** 420, 'Chinese'),
('Chicken Fried Rice', 'Fried rice with chicken pieces', 180.00, 520, 'Chinese'),
('Hakka Noodles', 'Stir-fried noodles with vegetables', 160.00, 450, 'Chinese'),
('Chicken Noodles', 'Noodles stir-fried with chicken', **190.00, 550,** 'Chinese'),
('Veg Manchurian', 'Fried veggie balls in spicy sauce', 170.00, 380, 'Chinese'),
('Chilli Chicken', 'Spicy chicken with bell peppers', 220.00, 480, 'Chinese');

-- STREET FOOD MENU:

INSERT INTO food roducts (name, description, price, calories, category) VALUES
('Pav Bhaji', 'Spiced vegetable mash with bread', **100.00,** 450, 'Street Food'),
('Vada Pav', 'Potato fritter in bread bun', 40.00, 320, 'Street Food'),
('Samosa (2 pcs)', 'Crispy fried pastry with potato filling', 30.00, 280, 'Street Food'),
('Pani Purr (6 pcs)', 'Crispy shells with spicy water', 50.00, 180, 'Street Food'),
('Bhel Purr', 'Puffed rice with tangy chutney', 60.00, 220, 'Street Food'),
('Dahi Puri', 'Crispy shells with yogurt and chutney', 70.00, 250, 'Street Food'),
('Raj Kachori', 'Large crispy shell with fillings', 80.00, 350, 'Street Food'),
('Aloo Tikki', 'Spiced potato patties', 50.00, 280, 'Street Food');

-- SANDWICH & WRAP MENU:

INSERT INTO food roducts (name, description, price, calories, category) VALUES
('Veg Sandwich', 'Grilled sandwich with vegetables', 80.00, 320, 'Sandwich'),
('Chicken Sandwich', 'Grilled chicken sandwich', 120.00, 420, 'Sandwich'),
('Paneer Wrap', 'Tortilla wrap with paneer filling', **140.00,** 480, 'Wrap'),
('Chicken Wrap', 'Tortilla wrap with grilled chicken', 160.00, 550, 'Wrap');

-- PASTA MENU:

INSERT INTO food roducts (name, description, price, calories, category) VALUES
('Pasta Alfredo', 'Creamy white sauce pasta', 200.00, 620, 'Pasta'),
('Pasta Arrabiata', 'Spicy tomato sauce pasta', 180.00, 520, 'Pasta'),
('Mac and Cheese', 'Macaroni in cheese sauce', 190.00, 580, 'Pasta');

-- SALAD MENU:

INSERT INTO food roducts (name, description, price, calories, category) VALUES
('Caesar Salad', 'Fresh salad with Caesar dressing', **150.00,** 280, 'Salad'),
('Greek Salad', 'Salad with feta cheese and olives', 170.00, 250, 'Salad');

-- APPETIZERS & SIDES MENU:

INSERT INTO food roducts (name, description, price, calories, category) VALUES
('Chicken Wings (6 pcs)', 'Spicy chicken wings', 220.00, 480, 'Appe'tizer'),
('French Fries', 'Crispy golden fries', 80.00, 320, 'Sides'),
('Onion Rings', 'Crispy fried onion rings', 100.00, 380, 'Sides');

```
-- DESSERTS MENU:

INSERT INTO food     roducts (name, description, price, calories,  category) VALUES
('Gulab Jamun (2 pcs)', 'Sweet milk  dumplings in sugar syrup', 60.00, 320, 'Dessert'),
('Rasmalai (2 pcs)', 'Cottage cheese dumplings in sweetened milk', 80.00, 280, 'Dessert'),
('Kulfi', 'Traditional  Indian ice cream', 70.00, 250, 'Dessert'),
('Chocolate Cake', 'Rich chocolate cake slice', 120.00, 450, 'Dessert'),
('Ice Cream (2 scoops)', 'Vanilla or chocolate ice cream', 90.00, 300, 'Dessert');


-- BEVERAGES MENU:

INSERT INTO food     roducts (name, description, price, calories, category) VALUES
('Masala  Chai', 'Indian spiced tea', 20.00, 80, 'Beverage'),
('Sweet  Lassi', 'Sweetened yogurt drink', 60.00, 180, 'Beverage'),
('Mango Lassi', 'Mango flavored yogurt drink', 80.00, 220, 'Beverage'),
('Badam Milk', 'Almond  flavored milk', 70.00, 200, 'Beverage'),
('Coke', 'Coca Cola 330ml', 40.00, 140, 'Beverage'),
('Fresh Lime Soda', 'Refreshing lemon soda', 50.00, 100, 'Beverage');

SELECT 'Complete database created  with Orders table!' AS Status;
SELECT COUNT(*) AS TotalFoodItems FROM food     roducts;
```

```
package fooddelivery;

import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModeI;
import javax.swing.table.DefaultTableCellRenderer;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.*;

public class SimpleFoodDelivery extends JFrame {
    // Database connection details
    private static final String DB_URL = "jdbc:mysql://localhost:3306/food_de1ivery_db";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "root";

    // Current user info
    private String currentUsername;
    private String currentUserAddress;

    // Components
    private JTabbedPane tabbedPane;
    private DefaultTableModel foodTableModel;
    private DefaultTableModel cartTableModel;
    private JTable foodTabIe;
    private JTable cartTable;
    private JLabel totalLabel;

    // Cart data structure
    private ArrayList<CartItem> cartltems = new ArrayList<>{};

    public SimpleFoodDelivery(String username, String address) {
        this.currentUsername = username;
        this.currentUserAddress = address;

        setTitle("Food Delivery System- Welcome " + username);
        setSize(1000, 700);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(nu8);

        initComponents{};
        loadFoodItems0i


    private void initComponents0 I
        tabbedPane = new JTabbedPane0i

        // Browse Foods Tab
        JPanel browsePanel = createBrowsePanel{};
        tabbedPane.addTab("Browse Foods", browsePanel);
```

```
// My Cart Tab
      JPaneI cartPanel  = createCartPanel0:
      tabbedPane.addTab(”My Cart", cartPanel),

      add(tabbedPane);


   private JPanel createBrowsePanel{} {
      JPanel panel = new JPaneI(new BorderLayout(10, 10)),
      panel.setBorder(new  EmptyBorder(20, 20, 20, 20));

      ff Title
      JLabel titleLabeI = new JLabel(" O  Browse Our Menu", JLabel.CENTER);
      titleLabe6setFont(new Font("Arial", Font.BOLD, 24)),
      panel.add(titleLabel, BorderLayout.NORTH),

      ff Food Table
      String[] columns = {”Category",  "Name", "Description", "Price (1)", ”Calories"};
      foodTableModel = new DefauItTableModel(columns, 0) {
         @Override
         public boolean isCellEditable(int row, int column) (
            return false;



      foodTable = new JTable(foodTableModel),
      foodTable.setRowHeight(30),
      foodTable.setFont(new Font("Arial", Font.PLAIN, 14));
      foodTabIe.getTableHeader{}.setFont(new Font("Arial", Font.BOLD, 14));
      foodTable.getTableHeader{}.setBackground(new Color(255, 140, 0));
      foodTable.getTableHeader{}.setForeground(Color.WHITE),

      ff Make category column bold
      DefaultTableCellRenderer categoryRenderer = new DefaultTableCellRenderer{}{
         @Override
         public Component getTableCellRendererComponent(JTable table, Object value,
               boolean isselected,  boolean hasFocus, int row, int column) {
            Component c = super.getTableCellRendererComponent(table, value, isselected,
hasFocus, row, column),

            // Check if this is a category header row
            if (column      0 && value != null) {
               String val = value.toString{},
               if (val.startsWith("═══"))  (
                  setFont(new Font("Arial", Font.BOLD, 16)),
                  setBackground(new Color(255,  200, 100)),
                  setForeground(Color.BLACK);
               } else if (val.isEmpty{}) {
                  setFont(table.getFont  );
                  setBackground(table.getBackground{}),
               } else {
                  setFont(new Font("Arial", Font.BOLD, 14)),
                  setBackground(table.getBackground{});
```

```
            setForeground(Color.BLACK);

        } else {
            setFont(table.getFont0)i
            if (!isSelected) {
                setBackground(table.getBackground0)i
                setForeground(table.getForeground(I);


        return c;


    foodTabIe.getCoIumnModeI0 getColumn(0).setCellRenderer(categoryRenderer);

    JScrollPane scrolIPane = new JScrollPane(foodTable);
    paneLadd(scrollPane, BorderLayout.CENTER);

    ff Bottom panel with Add to Cart
    JPanel bottomPaneI = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 10));

    JLabel qtyLabel = new JLabel(”Quantity:");
    qtyLabel.setFont(new Font(”Arial", Font.BOLD, 14));

    SpinnerNumberModeI spinnerModel = new SpinnerNumberModel(1, 1, 10, 1);
    JSpinner qtyspinner  = new JSpinner(spinnerModel);
    qtyspinner.setFont(new Font(”Arial”, Font.PLAIN, 14));
    ((JSpinner.DefaultEditor) qtyspinner.getEditor0)-getTextField0.setCoIumns(5);

    JButton addToCartBtn = new JButton(”Add  to Cart”);
    addToCartBtn.setFont(new Font(”Arial",  Font.BOLD, 16));
    addToCartBtn.setBackground(new Color(34, 139, 34));
    addToCartBtn.setForeground(Color.WHITE);
    addToCartBtn.setFocusPainted(false);
    addToCartBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));

    addToCartBtn.addActionListener(e  -> {
        int selectedRow  = foodTable.getSeIectedRow{};
        if (selectedRow      -1) {
            JOptionPane.showMessageDialog(this, "Please select a food item!", "No
Selection", JOptionPane.WARNING_MESSAGE),
            return;


        // Check if it's a category header
        String category = foodTableModel.getValueAt(selectedRow, 0).toString{},
        if (category.startsWith(”═══") || category.isEmpty  ) {
            JOptionPane.showMessageDialog(this, "Please select a food item, not a
category!", "Invalid Selection", JOptionPane.WARNING_MESSAGE);
            return;
```

```
        String name = foodTabIeModel.getValueAt(selectedRow, 1).toString{};
        String description = foodTableModel.getVaIueAt(selectedRow, 2).toString ;
        double price = Double.parseDouble(foodTableModel.getVaIueAt(selectedRow,
3).toString0.replace(”Z", "”).trim0)i
        int calories = Integer.parseInt(foodTableModefgetVaIueAt(selectedRow,
4).toStri•z0 replace(” cal", "")->› 0):
        int quantity = (Integer) qtySpinner.getValue0:

        addToCart(name, description, price, calories, quantity);


    bottomPanel.add(qtyLabel);
    bottomPanel.add(qtyspinner);
    bottomPanel.add(addToCartBtn);

    pane€add(bottomPanel, BorderLayout.SOUTH);

    return panel;


  private JPanel createCartPanel0 I
    JPanel panel = new JPanel(new BorderLayout(10, 10));
    pane£setBorder(new EmptyBorder(20, 20, 20, 20));

    ff Title
    JLabel titleLabel = new JLabel("      My Cart”, JLabel.CENTER);
    titleLabe€setFont(new Font("Arial", Font.BOLD, 24));
    pane£add(titleLabel, BorderLayout.NORTH);

    ff Cart Table
    String[] columns = {”Item Name", "Price (T)”, ”Calories", ”Quantity", "Subtotal

    cartTableModel = new DefaultTableModel(columns, 0) (
      @Override
      public boolean isCellEditable(int row, int column) (
        return false;



    cartTabIe = new JTable(cartTableModel);
    cartTable.setRowHeight(30);
    cartTable.setFont(new Font(”Arial”, Font.PLAIN, 14));
    cartTable.getTableHeader{}.setFont(new Font(”Arial”, Font.BOLD, 14));
    cartTable.getTableHeader{}.setBackground(new Color(255, 69, 0));
    cartTabIe.getTableHeader{}.setForeground(Color.WHI"I“E);

    JScrollPane scrolIPane = new JScrollPane(cartTable);
    panel.add(scrollPane, BorderLayout.CENTER);
```

```
        ff Bottom panel
        JPaneI bottomPaneI = new JPanel(new BorderLayout0).

        ff Total panel
        JPanel totalPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, 20, 10));
        totalLabel = new JLabel("Total: Z0.00");
        totalLabel.setFont(new Font(”Arial", Font.BOLD, 20));
        totalLabel.setForeground(new Color(255, 69, 0));
        totalPanel.add(totalLabel);

        ff Buttons panel
        JPaneI buttonsPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 10));

        JButton removeBtn  = new JButton("Remove Selected”);
        removeBtn.setFont(new Font(”Arial",  Font.BOLD, 14));
        removeBtn.setBackground(new Color(220, 20, 60));
        removeBtn.setForeground(Color.WHl’l”E);
        removeBtn.setFocusPainted(false);
        removeBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));

        removeBtn.addActionListener(e  ->  {
           int selectedRow = cartTabIe.getSelectedRow0:
           if (selectedRow != -1) {
              cartltems.remove(selectedRow);
              refreshCart0:
           } else (
              JOptionPane.showMessageDialog(this, "Please select an item to remove!", "No
Selection", JOptionPane.WARNING_MESSAGE);



        JButton clearBtn = new JButton(”Clear Cart”);
        clearBtn.setFont(new Font("Arial”, Font.BOLD, 14));
        clearBtn.setBackground(new Color(255, 140, 0));
        clearBtn.setForeground(Color.WHITE);
        clearBtn.setFocusPainted(false);
        clearBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));

        clearBtn.addActionListener(e -> {
           if (!cartItems.isEmpo0)  I
              int confirm = JOptionPane.showConfirmDialog(this,  "Clear all items from
cart?", "Confirm", JOptionPane.YES_NO_OPTION);
              if (confirm       JOptionPane.YES_OPTION) {
                 cartItems.elear0i
                 refreshCart{};



        JButton checkoutBtn = new JButton(”Proceed to Checkout”);
        checkoutBtn.setFont(new Font(”Arial”, Font.BOLD, 16));
        checkoutBtn.setBackground(new Color(34, 139, 34));
        checkoutBtn.setForeground(Color.WHl’l“E);
```

```
        checkoutBtn.setFocusPainted(false);
        checkoutBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));

        checkoutBtn.addActionListener(e  ->  proceedToCheckou  0)s

        buttonsPanel.add(removeBtn);
        buttonsPanel.add(clearBtn);
        buttonsPanel.add(checkoutBtn);

        bottomPanel.add(totalPanel, BorderLayout.NORTH);
        bottomPanel.add(buttonsPanel, BorderLayout.CENTER);

        panel.add(bottomPaneg BorderLayout.SOUTH);

        return panel;


    private void loadFoodItems0 t
        try (Connection  conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
             Statement stmt = conn.createStatement0i
             ResuItSet  rs = stmt.executeQuery("SELECT * FROM food    roducts ORDER BY
category, name")) {

            String  currentCategory =  ”";

            while (rs.next{}) {
                String category = rs.getString(”category”);
                String  name = rs.getString("name");
                String description  = rs.getString("description");
                double price = rs.getDouble("price”);
                int calories = rs.getInt("calories”);

                // Add category  header if  new category
                if (!category.equals(currentCategory)) {
                    foodTableModel.addRow(new Object[]{” ═══ "  + category.toUpperCase{} +
  ═══
                    currentCategory = category;


                // Add food item
                foodTableModel.addRow(new Object[]{category, name, description,  "2"  +
price, calories + ” cal”});


        } catch (SQLException  e) {
            JOptionPane.showMessageDialog(this, "Error loading food items: " +
e.getMessage{}, "Database Error", JOptionPane.ERROR_MESSAGE);
            e.printStackTrace0i
```

```
 private void addToCart(String name, String description, double price, int calories, int
quantity) {
       ff Check if item already exists in cart
       for (CartItem item : cartltems) {
          if (item.name.equals(name)) {
             item.quantity  += quantity;
             refreshCart0:
             JOptionPane.showMessageDialog(this, "Updated quantity in cart!", "Success",
JOptionPane.INFORMATION_MESSAGE);
             return;



       ff Add new item
       cartItems.add(new CartItem(name, price, calories, quantity));
       refreshCart0i
       JOptionPane.showMessageDialog(this,  "Added to cart!", "Success",
JOptionPane.INFORMATION_MESSAGE);


   private void refreshCart0  I
       cartTabIeModel.setRowCount(0);
       double total = 0;

       for (CartItem item : cartltems) {
          double subtotal = item.price * item.quantity;
          total += subtotal;
          cartTableModel.addRow(new Object[]{
             item.name,
             "Z" + item.price,
             item.calories + " cal",
             item.quantity,
             "Z"  +  String.format("%.2f", subtotal)



       totalLabel.setText("Total:  Z" + String.format("%.2f", total));


   private void proceedToCheckout0 I
       if (cartItems.isEmpU0) I
          JOptionPane.showMessageDialog(this, "Your cart is empty!", "Empty Cart",
JOptionPane.WARNING_MESSAGE);
          return;


       ff Create checkout dialog
       JDialog checkoutDialog = new JDialog(this,  "Checkout", true);
       checkoutDialog.setSize(500, 400);
       checkoutDialog.setLocationRelativeTo(this);
       checkoutDialog.setLayout(new  BorderLayout(10, 10));
```

```
JPaneI mainPanel = new JPanel(new GridLayout(5, 2, 10, 10));
mainPanel.setBorder(new EmptyBorder(20, 20, 20, 20));

// Address
mainPanel.add(new JLabel("Delivery Address:"));
JTextField addressField = new JTextField(currentUserAddress);
mainPanel.add(addressFieId);

// Payment method
mainPanel.add(new JLabel("Payment Method:"));
String[] paymentMethods = {"Cash on Delivery", "Credit/Debit Card", "UPI"};
JComboBox<String> paymentCombo = new JComboBox<>(paymentMethods);
mainPanel.add(paymentCombo);

// Card details (conditionally shown)
JLabel cardLabel = new JLabel("Card Number:");
JTextField cardField = new JTextField0i
mainPanel.add(cardLabel);
mainPanel.add(cardField);

JLabel cvvLabel = new JLabel("CVV:");
JTextField cvvFieId = new JTextFiel 0i
mainPanel.add(cvvLabel);
mainPanel.add(cvvField);

// Show/hide card fields based on payment method
cardLabel.setVisible(false);
cardField.setVisible(false);
cvvLabel.setVisible(false);
cvvFieId.setVisibIe(false);

paymentCombo.addActionListener(e -> {
    boolean showCard = paymentCombo.getSelectedItem0.equals("Credit/Debit
Card");
    cardLabel.setVisible(showCard);
    cardField.setVisible(showCard);
    cvvLabel.setVisible(showCard);
    cvvField.setVisible(showCard);


checkoutDialog.add(mainPanel, BorderLayout.CENTER);

// Buttons
JPanel btnPaneI = new JPanel(new FIowLayout{});
JButton placeOrderBtn = new JButton("Place Order"),
placeOrderBtn.setBackground(new Color(34, 139, 34));
placeOrderBtn.setForeground(Color.WHITE);
placeOrderBtn.setFont(new Font("Arial", Font.BOLD, 14));
```

```
    placeOrderBtn.addActionListener(e -> {
        String address = addressFieId.getText{}.t•  0'
        if (address.isEmpo0) t
            JOptionPane.showMessageDialog(checkoutDialog, "Please enter delivery
address!", "Error", JOptionPane.ERROR_MESSAGE);
            return;


        if (placeOrder(address, paymentCombo.getSelectedItem0-toString0)) I
            checkoutDialog.dispose0i
            cartltems.clear0:
            refreshCart0:
            JOptionPane.showMessageDiaIog(this, "Order placed successfully!\nDelivery
to: " + address, "Success", JOptionPane.INFORMATION_MESSAGE);



    JButton cancelBtn = new JButton("Cancel");
    cancelBtn.addActionListener(e -> checkoutDialog.dispose0)i

    btnPanel.add(placeOrderBtn);
    btnPanel.add(cancelBtn);
    checkoutDialog.add(btnPaneg BorderLayout.SOUTH);

    checkoutDialog.setVisibIe(true);


  private boolean placeOrder(String address, String paymentMethod) {
    double total = 0;
    for (CartItem item : cartltems) {
        total += item.price * item.quantity;


    try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) (
        String sql = "INSERT INTO orders (username, delivery_address,
payment_method, total_amount, order_items) VALUES (?, ?, ?, ?, ?)",
        PreparedStatement pstmt = conn.prepareStatement(sql),
        pstmt.setString(1, currentUsername);
        pstmt.setString(2, address),
        pstmt.setString(3, paymentMethod);
        pstmt.setDouble(4, total),

        StringBuilder items = new StringBuilder ;
        for (CartItem item : cartltems) {
            items.append(item.name).append(" x").append(item.quantity).append(", ");
```

```
            pstmt.setString(5, items.toString0):

            pstmt.executeUpdate0:
            return true;

        } catch (SQLException e) (
            JOptionPane.showMessageDialog(this, "Error placing order: ” + e.getMessage0.
"Database Error”, JOptionPane.ERROR_MESSAGE);
            e.printStaekTraee{};
            return false;



    // Cart Item class
    static class CartItem {
        String name;
        double price;
        int calories;
        int quantity;

        CartItem(String name, double price, int calories, int quantity) {
            this.name = name;
            this.price  = price;
            this.calories  = calories;
            this.quantity = quantity;



    // ——————— REGISTER WINDOW ——————————
    static class RegisterWindow extends JFrame {
        private JTextField nameField, usernameField, phoneFieId, addressField;
        private JPasswordField passwordField;

        public RegisterWindow{} (
            setTitle(”Register  - Food Delivery System");
            setSize(450,  500);
            setDefauItCloseOperation(JFrame.EXIT_ON_CLOSE);
            setLocationRelativeTo(null);
            setLayout(new BorderLayout{});

            // Title
            JLabel titleLabeI  = new JLabel(”Create Account”, JLabeLCENTER);
            titleLabeLsetFont(new Font("Arial", Font.BOLD, 28));
            titleLabeLsetBorder(new  EmptyBorder(20, 0, 20, 0));
            add(titleLabel,  BorderLayout.NORTH);

            // Form
            JPanel formPanel = new JPanel(new GridLayout(5, 2, 10, 15));
            formPanel.setBorder(new EmptyBorder(20, 40, 20, 40));

            formPanel.add(new JLabel(”Full  Name:"));
            nameField  = new JTextFieI  0:
            formPanel.add(nameField);
```

```
formPanel.add(new JLabel(”Username:”));
usernameField  = new JTextFiel*0i
formPanel.add(usernameField);

formPanel.add(new JLabel(”Password:”));
passwordField  = new JPasswordField0:
formPanel.add(passwordField);

formPanel.add(new JLabel(”Phone:”));
phoneFieId  = new JTextFieId0:
formPanel.add(phoneField);

formPanel.add(new JLabel(”Address:"));
addressField  = new JTextField0:
formPanel.add(addressField);

add(formPanel,  BorderLayout.CENTER);

ff Buttons
JPanel btnPanel = new JPaneI(new FIowLayout(FlowLayout.CENTER, 20, 20));

JButton registerBtn  = new JButton("Register”);
registerBtn.setFont(new Font("Arial”, Font.BOLD, 16));
registerBtn.setBackground(new Color(34, 139, 34));
registerBtn.setForeground(Color.WHITE);
registerBtn.setFocusPainted(false);
registerBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));
registerBtn.addActionListener(e  -> handIeRegister0)i

JButton IoginBtn  = new JButton(”Already have account? Login");
IoginBtn.setFont(new Font(”Arial”, Font.PLAIN, 12));
IoginBtn.setForeground(Color.BLUE);
loginBtn.setBorderPainted(false);
loginBtn.setContentAreaFilled(false);
IoginBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));
IoginBtn.addActionListener(e -> {
   dispose0i
   new LoginWindow0.setVisible(true);


btnPaneLadd(registerBtn);
btnPaneLadd(loginBtn);
add(btnPanel,  BorderLayout.SOUTH);


private void handleRegister0  I
   String name = nameField.getText{}.trim0i
   String username = usernameField.getTex 0        0i
   String password = new String(passwordField.getPassword0)->     0i
   String  phone = phoneField.getText{}.trim0i
   String address = addressField.getTest{}.trim0i
```

```
        if (name.isEmp90 11 username.isEmpty{} || password.isEmp90 Il Phone.isEmpty{}
|| address.isEmpty{}) {
            JOptionPane.showMessageDialog(this, "Please fill all fields!", "Error",
JOptionPane.ERROR_MESSAGE);
            return;


        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) (
            String sql = "INSERT INTO users (name, username, password, phone, address)
vALUEs (?, ?, ?, ?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, name);
            pstmt.setString(2, username);
            pstmt.setString(3, password);
            pstmt.setString(4, phone);
            pstmt.setString(5, address);

            pstmt.executeUpdate0i
            JOptionPane.showMessageDialog(this, "Registration successful! Please login.",
"Success", JOptionPane.INFORMATION_MESSAGE);

            dispose0i
            new LoginWindow0.setVisible(true);

        } catch (SQLExeeption e) {
            if (e.getMessage0-contains(”Duplicate entry”)) {
                JOptionPane.showMessageDiaIog(this, "Username already eaists!", "Error",
JOptionPane.ERROR_MESSAGE);
            } else {
                JOptionPane.showMessageDiaIog(this, "Registration failed: " +
e.getMessage{}, "Error", JOptionPane.ERROR_MESSAGE);
```

```
ff ====------ LOGIN WINDOW -------
static class LoginWindow extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordFieId;

    public LoginWindow0 I
        setTitle(”Login - Food Delivery System");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout{});

        // Title
        JLabel titleLabeI = new JLabel(”Welcome Back!", JLabel.CENTER);
        titleLabeLsetFont(new Font(”Arial”, Font.BOLD, 28));
        titleLabeLsetBorder(new EmptyBorder(30, 0, 30, 0));
```

```
        add(titleLabel, BorderLayout.NORTH);

        // Form
        JPanel formPanel = new JPanel(new GridLayout(2, 2, 10, 15));
        formPanel.setBorder(new EmptyBorder(20, 40, 20, 40));

        formPanel.add(new JLabel("Username:"));
        usernameField = new JTextFiel°0:
        formPanel.add(usernameField);

        formPanel.add(new JLabel("Password:"));
        passwordField = new JPasswordField0:
        formPanel.add(passwordField);

        add(formPanel, BorderLayout.CENTER);

        // Buttons
        JPanel btnPanel = new JPaneI(new FIowLayout(FlowLayout.CENTER, 20, 20));

        JButton IoginBtn = new JButton("Login");
        IoginBtn.setFont(new Font("Arial", Font.BOLD, 16));
        IoginBtn.setBackground(new Color(34, 139, 34));
        loginBtn.setForeground(Color.WHl'l"E);
        IoginBtn.setFocusPainted(false);
        IoginBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));
        IoginBtn.addActionListener(e -> handIeLogin{});

        JButton registerBtn = new JButton("Create new account");
        registerBtn.setFont(new Font("Arial", Font.PLAIN, 12));
        registerBtn.setForeground(Color.BLUE);
        registerBtn.setBorderPainted(false);
        registerBtn.setContentAreaFilled(false);
        registerBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));
        registerBtn.addActionListener(e -> {
            dispose0i
            new RegisterWindow0-setVisible(true);


        btnPanel.add(loginBtn);
        btnPanel.add(registerBtn);
        add(btnPanel, BorderLayout.SOUTH);


    private void handleLog>0 1
        String username = usernameField.getText0 trim{};
        String password = new String(passwordField.getPassword0).trim0i

        if (username.isEmptr0 II password.isEmpty{}) {
            JOptionPane.showMessageDialog(this, "Please enter username and password!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
```

```
        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) (
            String sql = "SELECT * FROM users WHERE username = ? AND password =

            PreparedStatement pstmt  = conn.prepareStatement(sql);
            pstmt.setString(1, username);
            pstmt.setString(2, password);

            ResultSet rs = pstmt.executeQuery0.

            if (rs.nex 0) I
               String address = rs.getString(”address”);
               JOptionPane.showMessageDialog(this,  "Login successful!", "Success",
JOptionPane.INFORMATION_MESSAGE);

               dispose{};
               new SimpleFoodDelivery(username, address).setVisible(true),
            } else {
               JOptionPane.showMessageDiaIog(this, "Invalid username or password!",
"Error", JOptionPane.ERROR_MESSAGE);


        } catch (SQLException  e) {
            JOptionPane.showMessageDiaIog(this,  "Login failed: " + e.getMessage0.
"Error", JOptionPane.ERROR_MESSAGE);




   ff———————— MAIN  METHOD  -- ————---
   public static void main(String[] args) {
      SwingUtiIities.invokeLater(0  -> I
         new RegisterWindow0 setVisible(true);
```

CHAPTER 5

SCREEN SHOTS

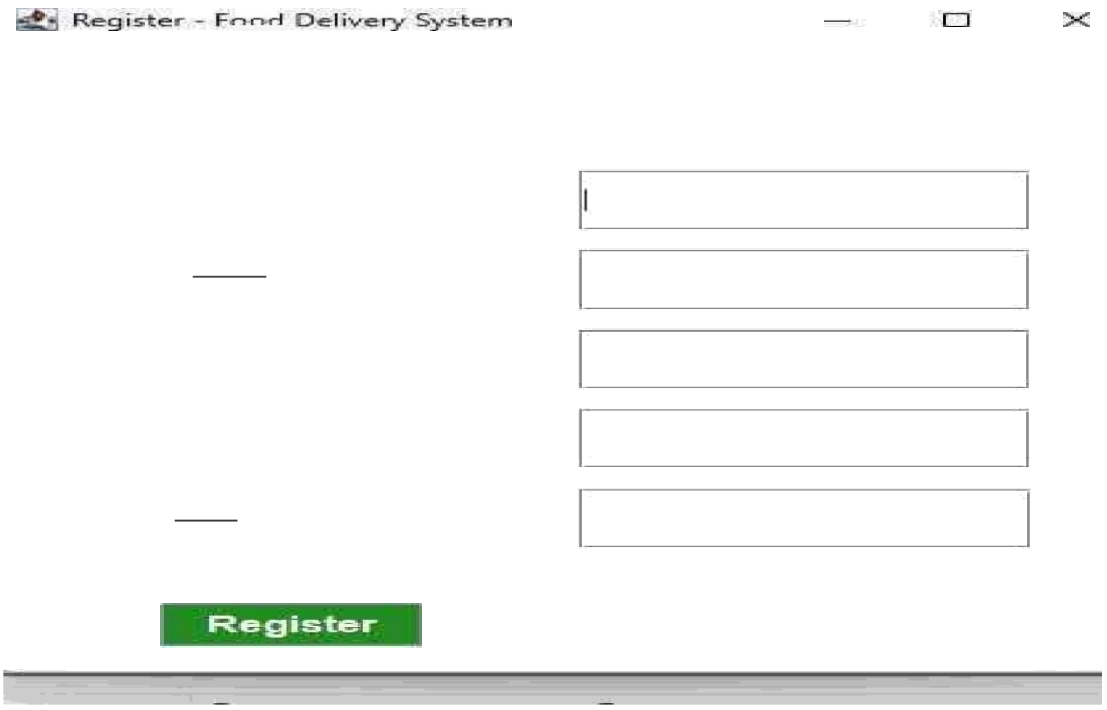


Fig 5.2 Dashboard

Create Account

Fig 5.3 UserDashboard

Fig 5.4 Menu Page

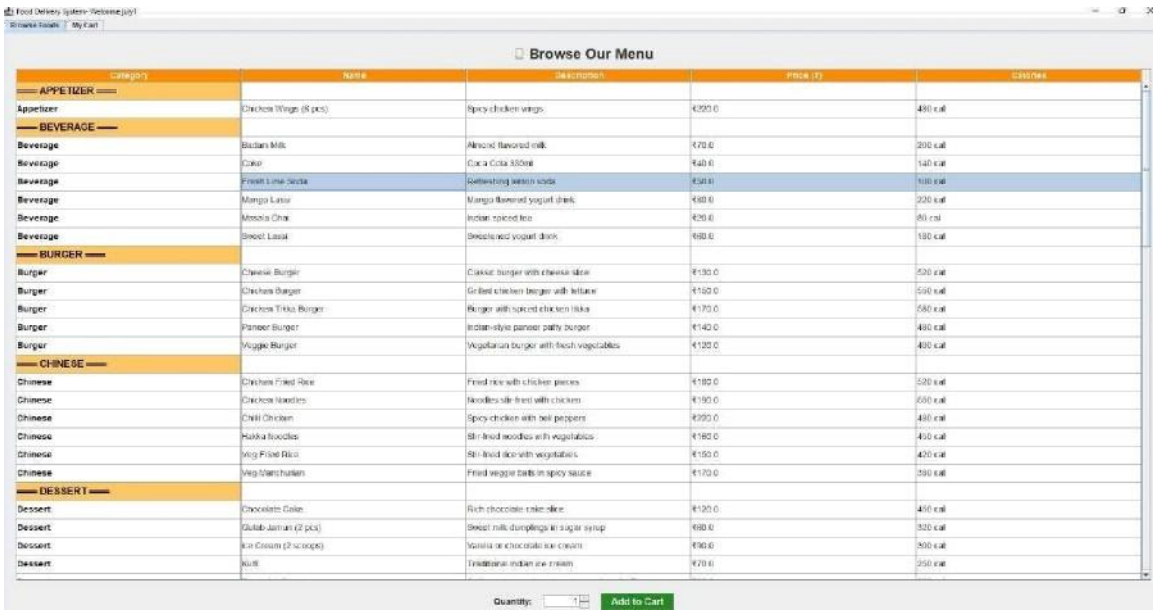
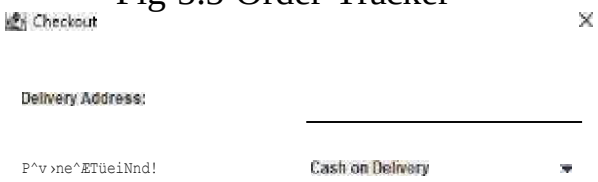


Fig 5.5 Order Tracker



CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The **Food Delivery System** successfully automates the process of ordering and delivering food through a user-friendly digital platform. It simplifies communication between customers, restaurants, and delivery agents, ensuring faster service and improved accuracy. The system effectively reduces manual work, minimizes human errors, and enhances customer satisfaction through real-time tracking and secure online payments. Overall, this project provides a reliable, efficient, and scalable solution for the growing demand in the online food service industry.

Future Enhancement

- Integration of **AI-based recommendations** to suggest food items based on user preferences and order history.
- Implementation of **GPS-based live tracking** for accurate and real-time delivery monitoring.
- Addition of **smart route optimization** to help delivery agents choose the fastest delivery path.
- Introduction of **multilingual support** to make the system accessible to users of different languages.
- Inclusion of **voice-based ordering** to enhance convenience and accessibility for users.
- Development of **a loyalty and reward** system to increase customer engagement and retention.
- Integration of data **analytics dashboards** for restaurants to analyze sales trends and customer behaviours.
- Migration to **cloud-based infrastructure** for better scalability, reliability, and performance.
- Implementation of **enhanced security features** for safe online payments and data protection.
- Integration with **third-party delivery and payment services** to expand the system's flexibility.

REFERENCES

1. https://www.w3schools.com/sql/
2. https://www.tutorialspoint.com/sq1ite/index.htm
3. https://www.wikipedia.org/
4. https://www.learnpython.org/
5. https://www.codecademy.com/learn/learn-python