

FOOD DELIVERY MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

DEEPAK R	230701067
DHARANIKUMAR S	230701073

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 – 25

BONAFIDE CERTIFICATE

Certified that this project report “**FOOD DELIVERY MANAGEMENT SYSTEM**” is

the bonafide work of “**DEEPAK R (230701067), DHARANIKUMAR S (230701073)**”

who carried out the project work under my supervision.

Submitted for the Practical Examination held on_____

SIGNATURE

Mrs.Divya M
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College,
Thandalam, Chennai 602 105

SIGNATURE

Mr.Ragu G
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College,
Thandalam, Chennai 602 105

INTERNAL EXAMINER

EXTERNAL EXAMINER

Abstract

The Food Delivery Management System is a robust and scalable solution designed to modernize and streamline the entire food ordering and delivery process. It acts as a bridge between customers, restaurants, and delivery personnel, offering a centralized platform that enhances efficiency, reliability, and user satisfaction. Customers can explore menus, customize orders, and make secure online payments, with the added convenience of real-time delivery tracking. The system also allows users to leave reviews and ratings, fostering transparency and trust in the platform.

Restaurants can leverage the system to manage incoming orders, monitor inventory, and generate detailed sales reports, enabling better decision-making and service optimization. Delivery personnel are automatically matched to orders using algorithms that consider proximity, traffic conditions, and delivery time constraints, ensuring faster and more efficient service.

Technological underpinnings of the system include a user-friendly front-end developed using tools like JavaFX, and a robust back-end powered by SQL databases for storing and managing user data, orders, and transaction histories. Additional features include notifications, promotional discounts, loyalty programs, and customer support integration, which collectively elevate the user experience.

The system reduces operational inefficiencies, minimizes human errors, and enhances scalability, making it an essential asset for businesses aiming to thrive in the competitive food delivery market. By leveraging data analytics and automation, it empowers stakeholders to deliver high-quality services, meet customer expectations, and build long-term loyalty, positioning it as a cornerstone in the modern food delivery ecosystem.

TABLE OF CONTENTS

Chapter 1

1 INTRODUCTION

1.1 INTRODUCTION	6
1.2 OBJECTIVES	7
1.3 MODULES	7

Chapter 2

2 SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION	9
2.2 LANGUAGES	9
2.2.1 JAVA	9
2.2.2 SQL.....	9

Chapter 3

3 REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION	12
3.1.1 FUNCTIONAL REQUIREMENTS.....	12
3.1.2 NON FUNCTIONAL REQUIREMENTS.....	13
3.2 HARDWARE AND SOFTWARE REQUIREMENTS	14
3.3 ARCHITECTURE DIAGRAM	15
3.4 ER DIAGRAM	16
3.5 NORMALIZATION.....	17

Chapter 4

4 PROGRAM CODE

4.1 PROGRAM CODE.....	18
-----------------------	----

Chapter 5

5 RESULTS AND DISCUSSION

5.1 RESULTS AND DISCUSSION	37
----------------------------------	----

Chapter 6

6 CONCLUSION

6.1 CONCLUSION-----	41
---------------------	----

Chapter 7

6 REFERENCES

7.1 REFERENCES-----	42
---------------------	----

Chapter 1: Introduction

1.1 Introduction

Food delivery has become an integral part of modern life, offering convenience and accessibility to customers while streamlining operations for restaurants and delivery personnel. The Food Delivery Management System is designed to enhance the efficiency, accuracy, and reliability of the food ordering and delivery process. This system acts as a bridge between customers, restaurants, and delivery agents, providing a seamless and unified platform for managing orders, ensuring timely deliveries, and improving overall service quality.

The system offers a comprehensive range of features, including menu browsing, real-time order tracking, dynamic driver assignment, and secure payment integration. It maintains a well-structured database to store and manage critical information such as customer details, order histories, delivery routes, and restaurant data. These features ensure a smooth and efficient operation while reducing errors and manual intervention.

Developed using Java, SQL, VS Code, and MySQL, the Food Delivery Management System employs cutting-edge technologies to create a robust and scalable platform. SQL serves as the backbone for managing relational data, ensuring the efficient storage and retrieval of information. Java provides a versatile and powerful programming environment for implementing application logic and backend processes. VS Code offers a lightweight yet powerful integrated development environment (IDE) to streamline the coding and debugging process, while MySQL ensures secure and reliable database management.

This report outlines the development process, system architecture, and technologies used to create the Food Delivery Management System. It demonstrates how these components work together to deliver a high-performing solution that meets the needs of all stakeholders involved. The ultimate goal is to provide a reliable, user-friendly platform that simplifies food delivery operations, enhances customer satisfaction, and supports the growth of businesses in the competitive food delivery industry.

1.2 Objectives

- To create a centralized database for managing customers, restaurants, orders, and delivery personnel.
- To provide real-time tracking of food orders and delivery statuses.
- To facilitate secure and seamless payment transactions for customers.
- To optimize the process of assigning delivery agents based on proximity and availability.
- To improve operational efficiency and enhance customer satisfaction through automation and data-driven insights.

1.3 Modules

1. Customer Registration Module

The Customer Registration Module allows users to create accounts and manage their profiles. Customers can register using their name, contact information, and delivery address. The module securely stores this data in the database and ensures it is accessible during the ordering process. Features include an intuitive sign-up form, email verification, and password recovery options to enhance user convenience.

2. Restaurant Management Module

This module enables restaurants to register, upload their menus, and manage their operations. Restaurants can update item availability, pricing, and special offers through a user-friendly interface. The module also allows restaurants to track incoming orders and prepare food efficiently.

3. Order Placement and Tracking Module

The Order Placement Module enables customers to browse restaurant menus, select items, customize orders, and place them with ease. Once an order is

placed, the system provides real-time tracking, showing the order status from preparation to delivery. This module ensures transparency and keeps customers informed.

4. Delivery Assignment Module

The Delivery Assignment Module dynamically matches delivery agents with orders based on factors like location, availability, and delivery time. The module uses algorithms to optimize delivery routes, reducing time and cost. Delivery agents receive notifications about their assigned orders, ensuring timely pickups and deliveries.

5. Payment Processing Module

This module integrates secure payment gateways to facilitate online transactions. Customers can pay using credit cards, debit cards, or e-wallets. It ensures transaction security through encryption and compliance with payment regulations, offering both pre-payment and cash-on-delivery options.

6. Admin Dashboard Module

The Admin Dashboard Module serves as the control center for system administrators. It provides an overview of customer registrations, restaurant activities, order statuses, and delivery operations. The module includes tools for monitoring system performance, generating reports, and managing user roles. It features role-based access control to ensure that sensitive information is accessible only to authorized personnel.

7. Database Module

The Database Module forms the backbone of the Food Delivery Management System. It is responsible for storing and managing data related to customers, orders, menus, delivery agents, and transactions. Built using MySQL, the database ensures data integrity and allows for efficient retrieval through SQL queries. Features like indexing and structured relationships ensure scalability and reliability.

Chapter 2: Survey of Technologies

2.1 Software Description

The Food Delivery Management System integrates Java and SQL using JDBC (Java Database Connectivity) to establish a seamless connection between the application logic and the database. The system is developed in VS Code, leveraging its rich feature set for efficient development, and uses MySQL as the database engine. These technologies ensure a robust, high-performance solution capable of handling complex food delivery operations.

2.2 Languages and Tools

2.2.1 Java

- Role: Java is the primary language for the system's backend logic and core operations.
- Usage: Responsible for implementing features like order processing, delivery assignment, and database interactions via JDBC.
- Advantages:
 - Object-Oriented Design: Enables modular, reusable, and maintainable code.
 - Platform Independence: "Write once, run anywhere" compatibility ensures flexibility.
 - Extensive Libraries: Provides built-in support for a wide range of functionalities.

2.2.2 SQL

- Role: SQL is used to manage and manipulate relational database structures for storing and retrieving system data.
- Usage: Implements schemas to organize data on customers, restaurants, orders, and delivery agents.

- Advantages:
 - Structured Querying: Allows efficient data retrieval and updates with well-optimized queries.
 - Relational Integrity: Maintains consistency and prevents data duplication.

2.2.3 JDBC

- Role: JDBC (Java Database Connectivity) acts as the middleware that connects Java applications to the MySQL database.
- Usage: Enables seamless communication between the system's Java code and the database for executing SQL queries and retrieving results.
- Advantages:
 - Standardized API: Simplifies database access across various RDBMS platforms.
 - Dynamic SQL Execution: Allows parameterized queries to enhance flexibility and security.
 - Error Handling: Provides mechanisms for managing database-related exceptions.

2.2.4 MySQL

- Role: MySQL serves as the relational database management system (RDBMS) for storing and managing data.
- Usage: Facilitates data storage for user profiles, orders, payment details, and delivery status.
- Advantages:
 - Scalable Performance: Supports expanding data requirements for growing businesses.
 - Secure Transactions: Protects sensitive customer and transaction data.

2.2.5 VS Code

- Role: VS Code is the integrated development environment (IDE) used for developing, debugging, and managing the application.

- Usage: Supports Java and SQL development through extensions and tools that streamline coding.
- Advantages:
 - Customization: Extensions for Java and MySQL enhance development capabilities.
 - Efficiency: Lightweight yet powerful, with tools like IntelliSense and debugging.

By combining these technologies, including the JDBC connector, the Food Delivery Management System ensures efficient data exchange, reliable operations, and a scalable architecture tailored to modern food delivery challenges.

Chapter 3: Requirements and Analysis

3.1 Requirement Specification

3.1.1 Functional Requirements

1. User Authentication and Authorization

- User Registration and Login: Enable users (customers, drivers, and administrators) to securely register and log in to the system.
- Role-Based Access Control: Assign specific permissions based on user roles (e.g., customer, delivery agent, admin).

2. Customer Order Management

- Order Placement: Allow customers to place orders, including details such as food items, quantity, and delivery address.
- Order Tracking: Provide real-time order tracking for customers.

3. Delivery Management

- Driver Matching: Automatically assign delivery agents to orders based on location and availability.
- Delivery Status Updates: Enable drivers to update delivery statuses (e.g., picked up, in transit, delivered).

4. Payment Processing

- Invoice Generation: Automatically generate and store invoices for customer orders.
- Transaction Tracking: Maintain a record of payments for auditing purposes.

5. Admin Dashboard

- Data Monitoring: Provide an overview of orders, deliveries, and payments in a centralized dashboard.
- Report Generation: Generate detailed reports on system performance and key metrics.

3.1.2 Non-Functional Requirements

1. Security

- Data Encryption: Encrypt sensitive data such as login credentials and payment details.
- Access Restrictions: Implement strict access control for sensitive operations.

2. Performance

- Scalability: Design the system to accommodate a growing number of users and transactions.
- Response Time: Ensure quick response times for user queries and actions.

3. Reliability

- Availability: Guarantee high availability with minimal downtime.
- Backup: Regularly back up data to ensure recoverability in case of failure.

4. Usability

- Intuitive Interface: Design a straightforward interface for users to place orders and track deliveries.

5. Maintainability

- Modular Design: Ensure a modular architecture to simplify debugging and feature additions.

- Documentation: Provide detailed documentation for developers and users.

3.2 Hardware and Software Requirements

Hardware Requirements:

- Desktop PC or Laptop: A reliable device for development and deployment.
- Processor: Intel® Core™ i3 or higher for efficient processing.
- RAM: 4 GB or more to handle concurrent operations.
- Storage: At least 100 GB for database storage and system files.
- Monitor Resolution: 1024 x 768 resolution or higher for optimal display.

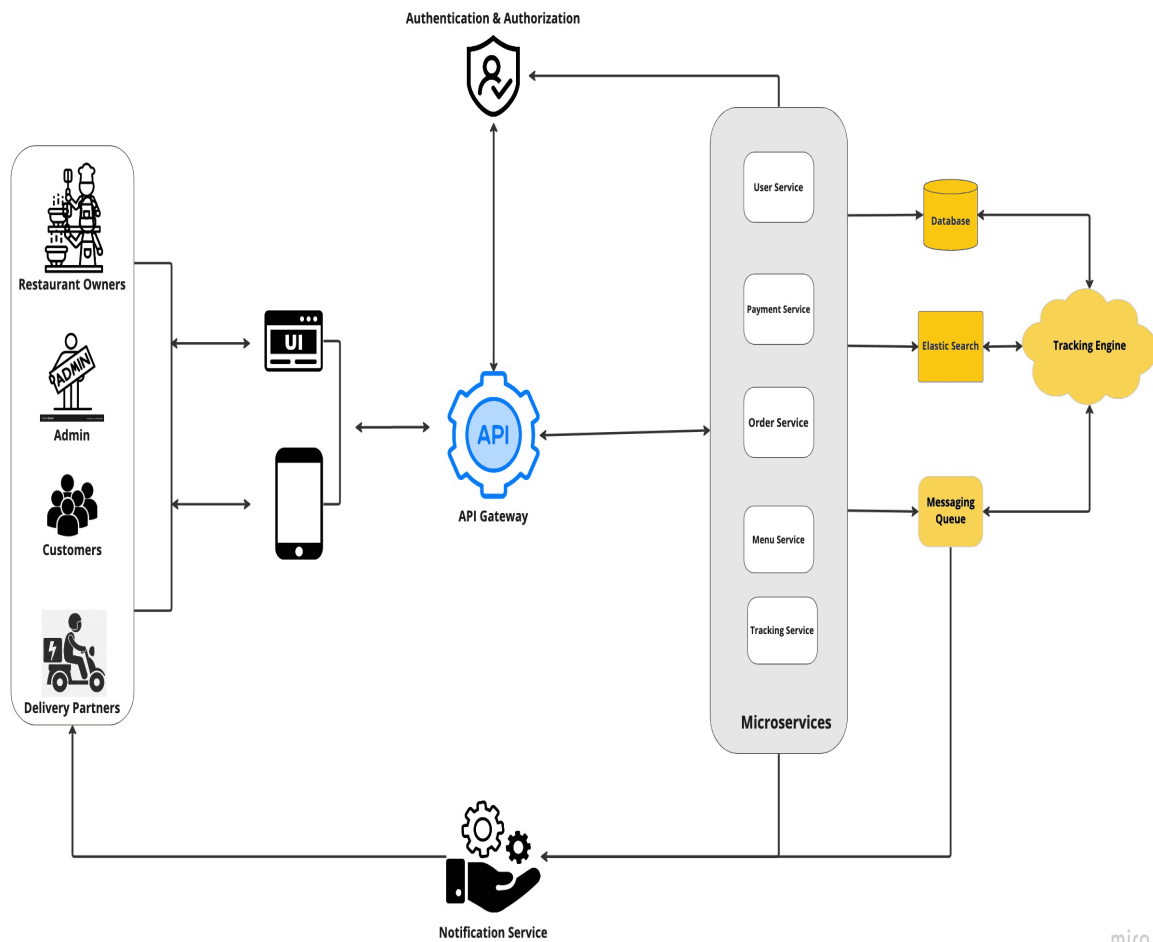
Software Requirements:

- Operating System: Windows 10 or later, or a compatible Linux distribution.
- Code Editor: Visual Studio Code with extensions for Java and SQL.
- Backend Language: Java.
- Database Management System: MySQL.
- Middleware: JDBC (Java Database Connectivity) for database integration.
- Version Control: Git for source code management.

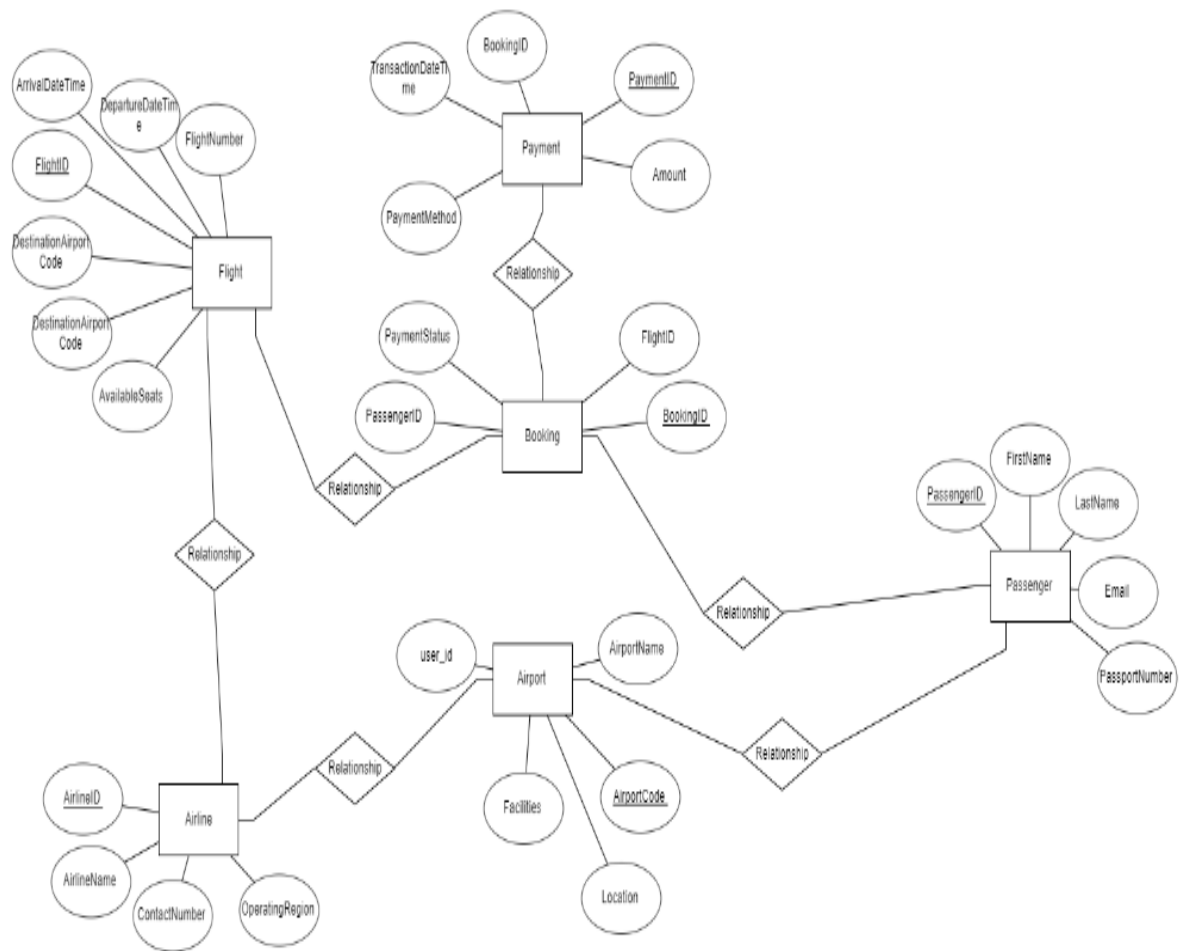
This setup ensures the system's reliability, scalability, and ease of maintenance while meeting the needs of all stakeholders.

3.3 ARCHITECTURE DIAGRAM

-A visual diagram that provides an overall view of the Food Delivery management system, identifying the external entities that interact with the system and the major data flows between these entities and the system.



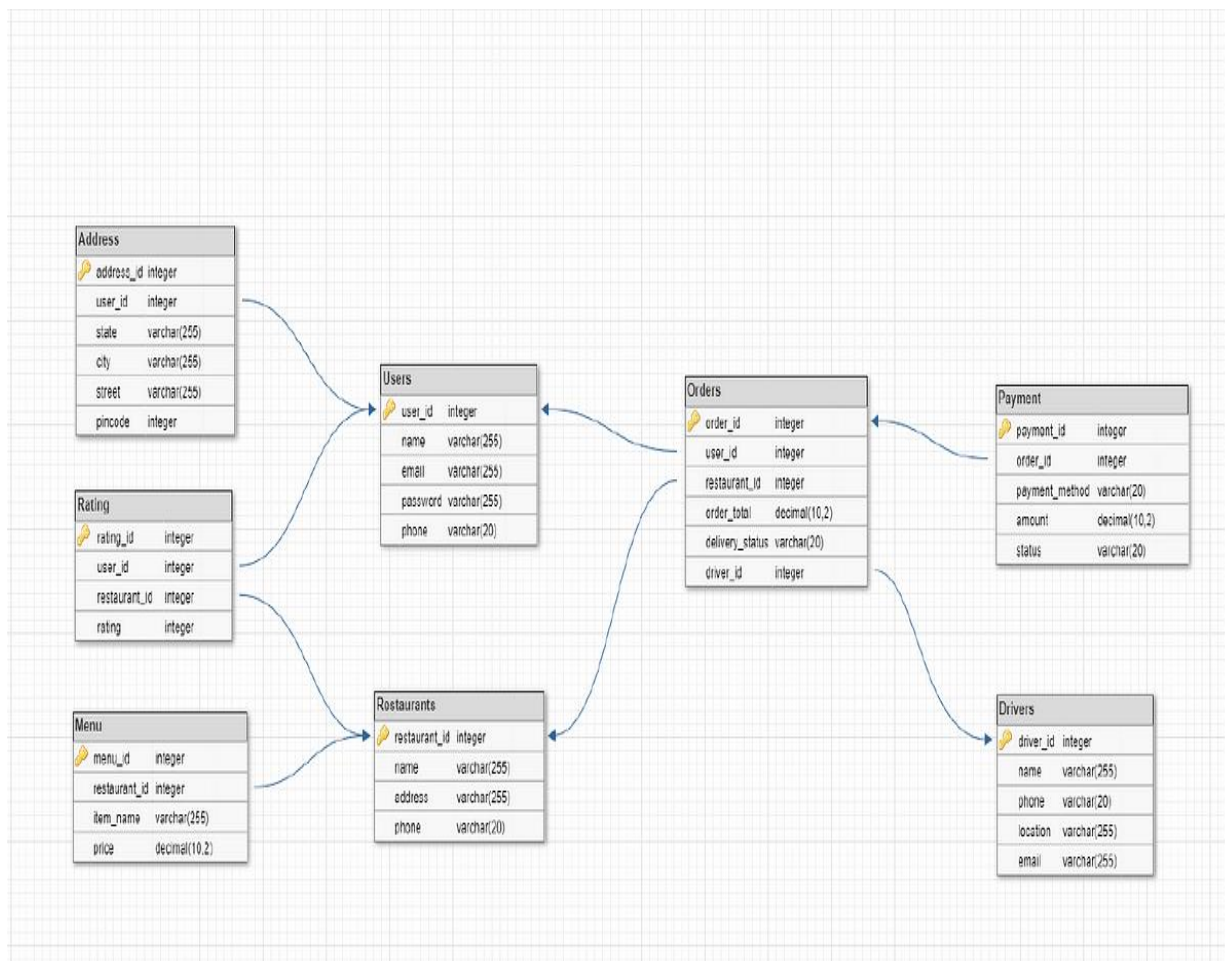
3.4 ER DIAGRAM



3.5 Normalization

-Normalization is the process of structuring data in a database to reduce redundancy and ensure data integrity. For the Food Delivery Management System, normalization involves organizing the database into multiple related tables and defining relationships between them to improve efficiency and maintainability.

RAW DATABASE



CHAPTER 4: Program Code

4.1 Program Code

Frontend (java) code

```
import java.awt.*;
import java.sql.*;
import java.util.ArrayList;
import javax.swing.*;

class FoodItem {
    int id;
    String name;
    double price;

    FoodItem(int id, String name, double price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    @Override
    public String toString() {
        return id + ". " + name + " - ₹" + price;
    }
}

class CartItem {
    FoodItem foodItem;
```

```
int quantity;
```

```
CartItem(FoodItem foodItem, int quantity) {
```

```
    this.foodItem = foodItem;
```

```
    this.quantity = quantity;
```

```
}
```

```
double getTotalPrice() {
```

```
    return foodItem.price * quantity;
```

```
}
```

```
@Override
```

```
public String toString() {
```

```
    return foodItem.name + " x " + quantity + " = ₹" + getTotalPrice();
```

```
}
```

```
}
```

```
public class FoodDeliveryAppSwing extends JFrame {
```

```
    private static Connection connection;
```

```
    private static ArrayList<FoodItem> menu = new ArrayList<>();
```

```
    private static ArrayList<CartItem> cart = new ArrayList<>();
```

```
    private static String loggedInUsername;
```

```
    private JList<String> menuList;
```

```
    private DefaultListModel<String> menuListModel;
```

```
    private JList<String> cartList;
```

```
    private DefaultListModel<String> cartListModel;
```

```

    private JButton addToCartButton, editCartButton, deleteCartButton,
    placeOrderButton, viewProfileButton;

    private JLabel totalAmountLabel;

    // New components for Login and Signup

    private JTextField usernameField, signUpUsernameField, signUpNameField,
    signUpAddressField, signUpEmailField;

    private JPasswordField passwordField, signUpPasswordField;

    public FoodDeliveryAppSwing() {

        // Initialize JFrame

        setTitle("Food Delivery App - Sign Up");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null); // Center the window

        // Sign-Up Page UI

        JLabel nameLabel = new JLabel("Name: ");
        JLabel addressLabel = new JLabel("Address: ");
        JLabel emailLabel = new JLabel("Email: ");
        JLabel usernameLabel = new JLabel("Username: ");
        JLabel passwordLabel = new JLabel("Password: ");
        signUpNameField = new JTextField();
        signUpAddressField = new JTextField();
        signUpEmailField = new JTextField();
        signUpUsernameField = new JTextField();
        signUpPasswordField = new JPasswordField();
        JButton signUpSubmitButton = new JButton("Sign Up");

```

```

JButton loginButton = new JButton("Already registered? Login");

signUpSubmitButton.addActionListener(e -> signUpUser());
loginButton.addActionListener(e -> showLoginPage());

JPanel signUpPanel = new JPanel(new GridLayout(6, 2));
signUpPanel.add(nameLabel);
signUpPanel.add(signUpNameField);
signUpPanel.add(addressLabel);
signUpPanel.add(signUpAddressField);
signUpPanel.add(emailLabel);
signUpPanel.add(signUpEmailField);
signUpPanel.add(usernameLabel);
signUpPanel.add(signUpUsernameField);
signUpPanel.add(passwordLabel);
signUpPanel.add(signUpPasswordField);
signUpPanel.add(loginButton);
signUpPanel.add(signUpSubmitButton);

add(signUpPanel, BorderLayout.CENTER);
}

private void showLoginPage() {
    // Remove Sign-Up Page UI before showing Login page
    getContentPane().removeAll();
    revalidate();
    repaint();
}

```

```

JFrame loginFrame = new JFrame("Login");
loginFrame.setSize(400, 300);
loginFrame.setLocationRelativeTo(null);

JLabel usernameLabel = new JLabel("Username: ");
JLabel passwordLabel = new JLabel("Password: ");
usernameField = new JTextField();
passwordField = new JPasswordField();
JButton loginSubmitButton = new JButton("Login");

loginSubmitButton.addActionListener(e -> loginUser());

JPanel loginPanel = new JPanel(new GridLayout(3, 2));
loginPanel.add(usernameLabel);
loginPanel.add(usernameField);
loginPanel.add(passwordLabel);
loginPanel.add(passwordField);
loginPanel.add(new JLabel());
loginPanel.add(loginSubmitButton);

loginFrame.add(loginPanel);
loginFrame.setVisible(true);
}

private void loginUser() {
    String username = usernameField.getText();

```

```

String password = new String(passwordField.getPassword());

if (username.isEmpty() || password.isEmpty()) {
    showMessage("Error", "Please enter both username and password.");
    return;
}

try {
    String query = "SELECT * FROM users WHERE username = ? AND
password = ?";

    PreparedStatement pst = connection.prepareStatement(query);
    pst.setString(1, username);
    pst.setString(2, password);
    ResultSet rs = pst.executeQuery();

    if (rs.next()) {
        loggedInUsername = username; // Set the logged-in user
        showMessage("Success", "Login successful!");
        showDashboard(); // Show the dashboard after login
    } else {
        showMessage("Error", "Invalid credentials.");
    }
} catch (SQLException e) {
    showMessage("Error", "Error checking user credentials: " +
e.getMessage());
}
}

```

```

private void signUpUser() {
    String name = signUpNameField.getText();
    String address = signUpAddressField.getText();
    String email = signUpEmailField.getText();
    String username = signUpUsernameField.getText();
    String password = new String(signUpPasswordField.getPassword());

    if (name.isEmpty() || address.isEmpty() || email.isEmpty() ||
username.isEmpty() || password.isEmpty()) {
        showMessage("Error", "Please fill in all the fields.");
        return;
    }

    try {
        String query = "INSERT INTO users (name, address, email, username,
password) VALUES (?, ?, ?, ?, ?)";
        PreparedStatement pst = connection.prepareStatement(query);
        pst.setString(1, name);
        pst.setString(2, address);
        pst.setString(3, email);
        pst.setString(4, username);
        pst.setString(5, password);
        int result = pst.executeUpdate();

        if (result > 0) {
            showMessage("Success", "Sign up successful! You can now login.");
            showLoginPage(); // Show login page after successful sign-up
        } else {

```



```
        showMessage("Error", "Error signing up. Please try again.");
    }
} catch (SQLException e) {
    showMessage("Error", "Error signing up: " + e.getMessage());
}
}
```

```
private void showDashboard() {
    // Remove previous UI before showing the dashboard
    getContentPane().removeAll();
    revalidate();
    repaint();

    JFrame dashboardFrame = new JFrame("Dashboard");
    dashboardFrame.setSize(600, 400);
    dashboardFrame.setLocationRelativeTo(null);

    // Add buttons to view menu and profile
    JButton viewMenuButton = new JButton("View Menu");
    JButton viewProfileButton = new JButton("Profile");

    viewMenuButton.addActionListener(e -> showMenuPage());
    viewProfileButton.addActionListener(e -> showUserProfile());

    JPanel dashboardPanel = new JPanel(new GridLayout(2, 1));
    dashboardPanel.add(viewMenuButton);
    dashboardPanel.add(viewProfileButton);
}
```

```

        dashboardFrame.add(dashboardPanel, BorderLayout.CENTER);
        dashboardFrame.setVisible(true);
    }

    private void showUserProfile() {
        // Fetch user profile details
        try {
            String query = "SELECT * FROM users WHERE username = ?";
            PreparedStatement pst = connection.prepareStatement(query);
            pst.setString(1, loggedInUsername);
            ResultSet rs = pst.executeQuery();

            if (rs.next()) {
                String name = rs.getString("name");
                String address = rs.getString("address");
                String email = rs.getString("email");

                // Profile page with Logout button
                JFrame profileFrame = new JFrame("User Profile");
                profileFrame.setSize(400, 300);
                profileFrame.setLocationRelativeTo(null);

                JLabel nameLabel = new JLabel("Name: " + name);
                JLabel addressLabel = new JLabel("Address: " + address);
                JLabel emailLabel = new JLabel("Email: " + email);
            }
        }
    }

```

```

        JButton logoutButton = new JButton("Logout");
        logoutButton.addActionListener(e -> logoutUser());

        JPanel profilePanel = new JPanel(new GridLayout(4, 1));
        profilePanel.add(nameLabel);
        profilePanel.add(addressLabel);
        profilePanel.add(emailLabel);
        profilePanel.add(logoutButton);

        profileFrame.add(profilePanel, BorderLayout.CENTER);
        profileFrame.setVisible(true);
    }
} catch (SQLException e) {
    showMessage("Error", "Error fetching user profile: " + e.getMessage());
}
}

private void logoutUser() {
    loggedInUsername = null;

    // Close all open frames except for the main application frame
    Window[] windows = Window.getWindows();
    for (Window window : windows) {
        if (window instanceof JFrame && window.isShowing()) {
            window.dispose();
        }
    }
}

```

```
// Show a success message
showMessage("Success", "You have been logged out.");

// Open the login page
showLoginPage();
}
```

```
private void showMenuPage() {
    // Remove previous UI before showing the menu
    getContentPane().removeAll();
    revalidate();
    repaint();

    JFrame menuFrame = new JFrame("Menu");
    menuFrame.setSize(600, 400);
    menuFrame.setLocationRelativeTo(null);

    menuListModel = new DefaultListModel<>();
    cartListModel = new DefaultListModel<>();
    menuList = new JList<>(menuListModel);
    cartList = new JList<>(cartListModel);
    totalAmountLabel = new JLabel("Total Amount: ₹0");

    addToCartButton = new JButton("Add to Cart");
    editCartButton = new JButton("Edit Cart");
}
```

```

deleteCartButton = new JButton("Delete Cart");
placeOrderButton = new JButton("Place Order");

addToCartButton.addActionListener(e -> addItemToCart());
editCartButton.addActionListener(e -> editCartItem());
deleteCartButton.addActionListener(e -> deleteCartItem());
placeOrderButton.addActionListener(e -> placeOrder());

// Load Menu items
loadMenu();

JPanel menuPanel = new JPanel(new GridLayout(1, 2));
menuPanel.add(new JScrollPane(menuList));
menuPanel.add(new JScrollPane(cartList));

JPanel buttonPanel = new JPanel(new FlowLayout());
buttonPanel.add(addToCartButton);
buttonPanel.add(editCartButton);
buttonPanel.add(deleteCartButton);
buttonPanel.add(placeOrderButton);

menuFrame.add(menuPanel, BorderLayout.CENTER);
menuFrame.add(buttonPanel, BorderLayout.SOUTH);
menuFrame.setVisible(true);
}

private void loadMenu() {

```

```

try {
    String query = "SELECT * FROM menu";
    Statement stmt = connection.createStatement();
    ResultSet rs = stmt.executeQuery(query);

    menu.clear();
    menuListModel.clear();

    while (rs.next()) {
        int id = rs.getInt("id");
        String name = rs.getString("name");
        double price = rs.getDouble("price");
        FoodItem item = new FoodItem(id, name, price);
        menu.add(item);
        menuListModel.addElement(item.toString());
    }
} catch (SQLException e) {
    showMessage("Error", "Error loading menu: " + e.getMessage());
}

}

private void addItemToCart() {
    int selectedIndex = menuList.getSelectedIndex();
    if (selectedIndex == -1) {
        showMessage("Error", "Please select a food item to add to the cart.");
        return;
    }
}

```

```

FoodItem selectedItem = menu.get(selectedIndex);
String quantityString = JOptionPane.showInputDialog("Enter Quantity: ");
try {
    int quantity = Integer.parseInt(quantityString);
    if (quantity > 0) {
        cart.add(new CartItem(selectedItem, quantity));
        updateCart();
    } else {
        showMessage("Error", "Invalid quantity.");
    }
} catch (NumberFormatException e) {
    showMessage("Error", "Please enter a valid number for quantity.");
}
}

```

```

private void editCartItem() {
    int selectedIndex = cartList.getSelectedIndex();
    if (selectedIndex == -1) {
        showMessage("Error", "Please select an item from the cart to edit.");
        return;
    }
}

```

```

CartItem cartItem = cart.get(selectedIndex);

String newQuantityString = JOptionPane.showInputDialog("Enter new
quantity for " + cartItem.foodItem.name + ": ");
try {
    int newQuantity = Integer.parseInt(newQuantityString);

```

```

        if (newQuantity > 0) {
            cartItem.quantity = newQuantity;
            updateCart();
        } else {
            showMessage("Error", "Invalid quantity.");
        }
    } catch (NumberFormatException e) {
        showMessage("Error", "Please enter a valid number for quantity.");
    }
}

```

```

private void deleteCartItem() {
    int selectedIndex = cartList.getSelectedIndex();
    if (selectedIndex != -1) {
        cart.remove(selectedIndex);
        updateCart();
    }
}

```

```

private void updateCart() {
    cartListModel.clear();
    double totalAmount = 0;

    for (CartItem cartItem : cart) {
        cartListModel.addElement(cartItem.toString());
        totalAmount += cartItem.getTotalPrice();
    }
}

```



```

        totalAmountLabel.setText("Total Amount: ₹" + totalAmount);
    }

    private void placeOrder() {
        if (cart.isEmpty()) {
            showMessage("Error", "Your cart is empty.");
            return;
        }

        StringBuilder orderSummary = new StringBuilder("Your Order
        Summary:\n");
        double totalAmount = 0;
        for (CartItem cartItem : cart) {
            orderSummary.append(cartItem.toString()).append("\n");
            totalAmount += cartItem.getTotalPrice();
        }
        orderSummary.append("Total Amount: ₹").append(totalAmount);

        // Show message that the order has been placed
        JOptionPane.showMessageDialog(this, orderSummary.toString() +
        "\n\nYour order has been placed.", "Order Placed",
        JOptionPane.INFORMATION_MESSAGE);
        cart.clear(); // Clear cart after placing the order
        updateCart();
    }

    private void showMessage(String title, String message) {

```

```

        JOptionPane.showMessageDialog(this, message, title,
JOptionPane.INFORMATION_MESSAGE);
    }

    public static void main(String[] args) {
        try {
            // Establish the database connection

            connection =
DriverManager.getConnection("jdbc:mysql://localhost/fooddelivery", "root",
"Dharani@2006");

            // Show the Sign-Up page initially

            SwingUtilities.invokeLater(() -> new
FoodDeliveryAppSwing().setVisible(true));
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

Backend (SQL) code

```
CREATE DATABASE fooddelivery;
```

```
USE fooddelivery;
```

```
CREATE TABLE menu (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(255) NOT NULL,  
    price DOUBLE NOT NULL  
);
```

```
CREATE TABLE orders (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    address VARCHAR(255) NOT NULL,  
    total DOUBLE NOT NULL,  
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

```
CREATE TABLE order_details (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    order_id INT NOT NULL,  
    food_id INT NOT NULL,  
    quantity INT NOT NULL,  
    total_price DOUBLE NOT NULL,  
    FOREIGN KEY (order_id) REFERENCES orders(id),  
    FOREIGN KEY (food_id) REFERENCES menu(id)  
);
```

```
INSERT INTO menu (name, price) VALUES
```

```
('Burger', 60.00),
```

```
('Pizza', 120.00),
```

```
('Pasta', 40.00),
```

```
('Salad', 55.00),
```

```
('Soda', 30.00);
```

```
select * from orders;
```

```
ALTER TABLE order_details ADD COLUMN price DOUBLE;
```

```
ALTER TABLE order_details MODIFY total_price DOUBLE NULL;
```

```
describe order_details;
```

```
ALTER TABLE order_details DROP COLUMN price;
```

```
CREATE TABLE users (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    username VARCHAR(255) UNIQUE NOT NULL,
```

```
    password VARCHAR(255) NOT NULL,
```

```
    name VARCHAR(255),
```

```
    email VARCHAR(255),
```

```
    address VARCHAR(255) NOT NULL
```

```
);
```

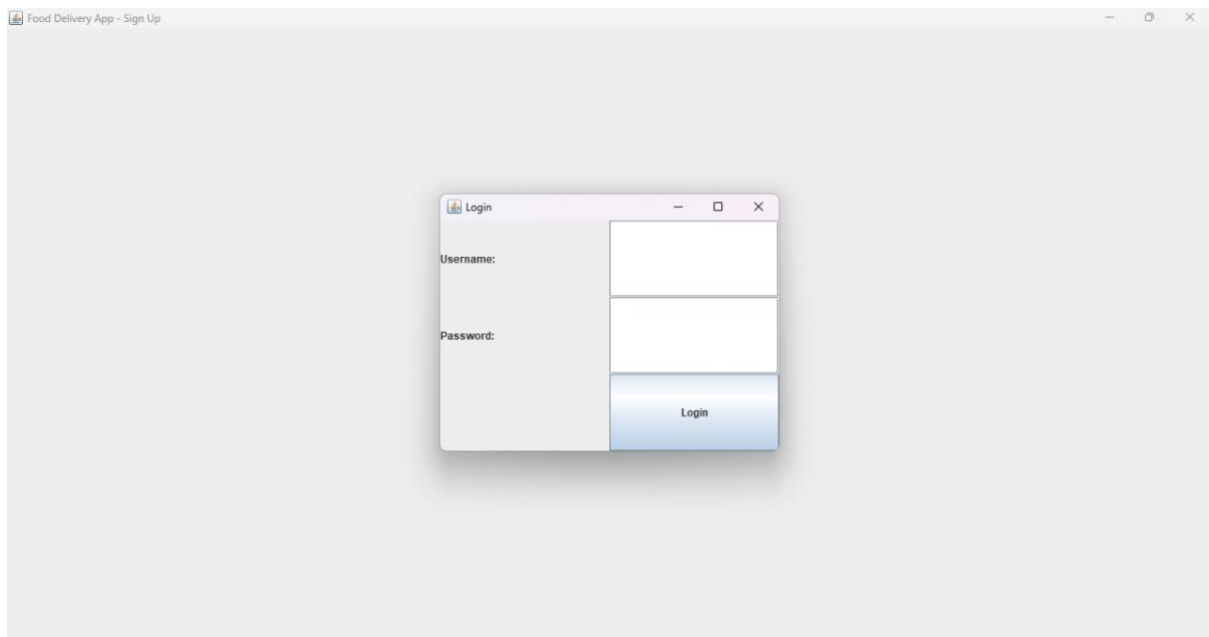
Chapter 5: Results And Discussions

Sign up page



The screenshot shows a web browser window titled "Food Delivery App - Sign Up". The page has a light gray background. On the left side, there are five labels: "Name:", "Address:", "Email:", "Username:", and "Password:". To the right of these labels are five corresponding white input fields. At the bottom of the page, there are two blue buttons: "Already registered? Login" on the left and "Sign Up" on the right.

Login page



The screenshot shows a web browser window titled "Food Delivery App - Sign Up" with a light gray background. In the center of the page, there is a smaller window titled "Login". This "Login" window has a light gray background and contains two labels: "Username:" and "Password:". To the right of these labels are two corresponding white input fields. Below the input fields is a blue button labeled "Login".

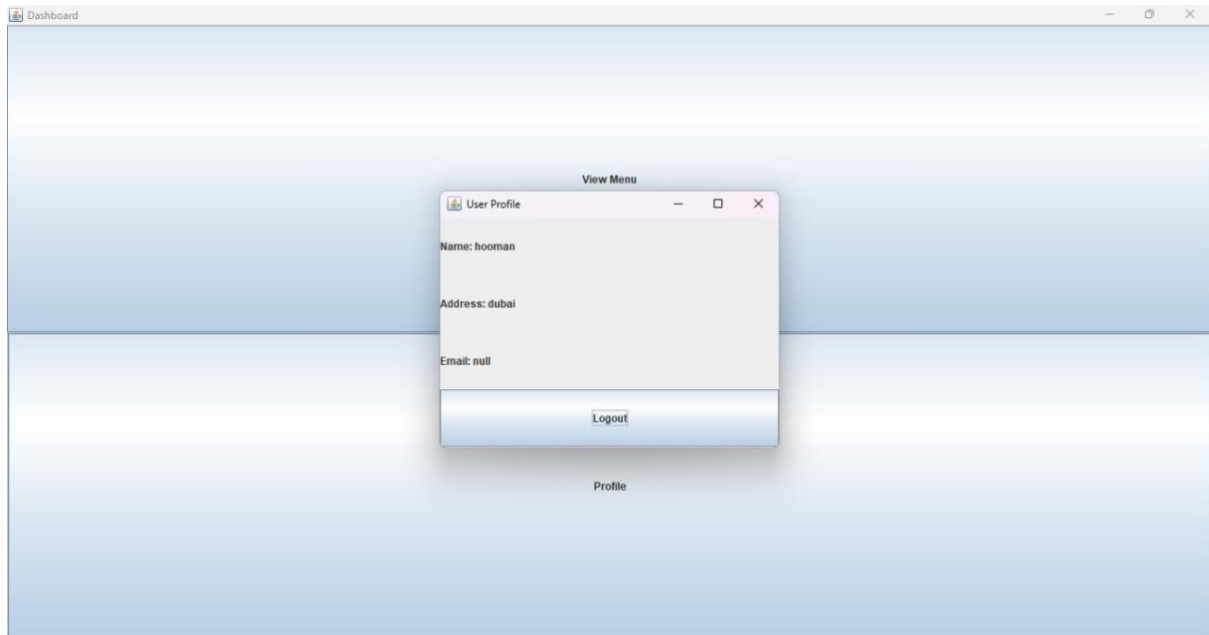
Home page



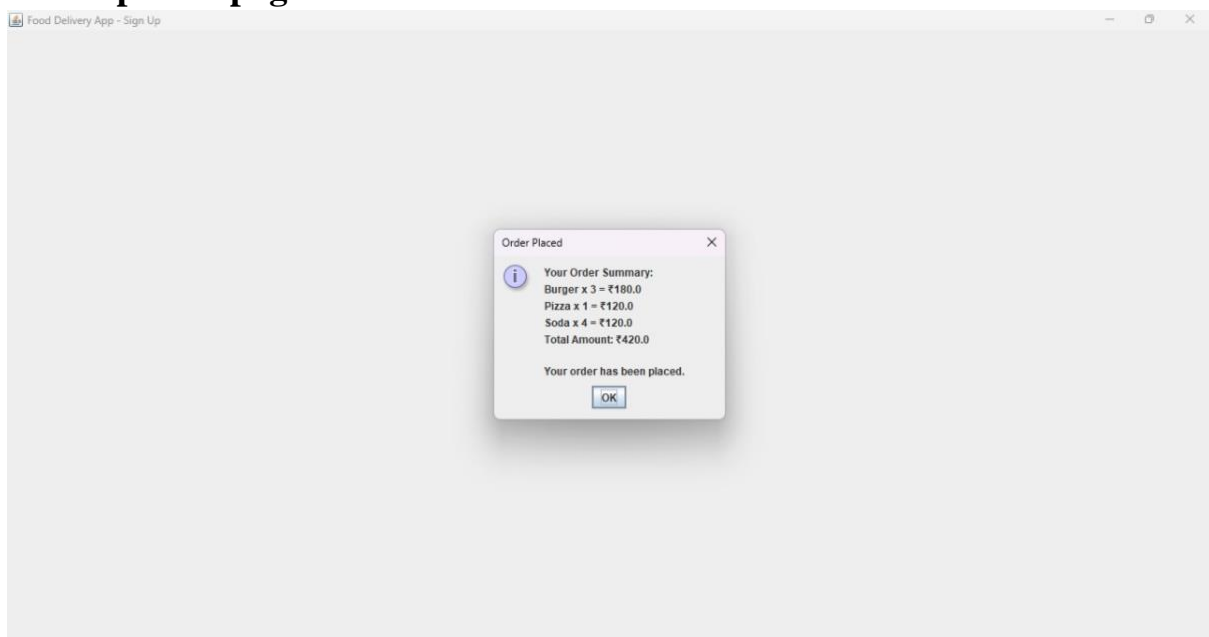
Menu page



Profile page



Order placed page



History of login page

The screenshot shows the MySQL Workbench interface with the 'fooddelivery' database selected. The 'Schemas' pane on the left shows the database structure. The 'Query' pane displays the following SQL code:

```
51 ALTER TABLE users
52 ADD COLUMN address VARCHAR(255) NOT NULL;
53
54 select *from users;
55
56 select *from orders;
57 select *from order_details;
```

The 'Result Grid' shows the results of the query 'select *from order_details;'. The table has 5 columns: id, order_id, food_id, quantity, and total_price. The data is as follows:

id	order_id	food_id	quantity	total_price
1	1	3	2	14.98
2	2	1	2	120
3	2	5	3	90

The 'Output' pane shows the execution history of the queries:

#	Time	Action	Message	Duration / Fetch
1	20:23:00	select from users LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
2	20:23:36	select from orders LIMIT 0, 1000	16 row(s) returned	0.000 sec / 0.000 sec
3	20:23:51	select from order_details LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

History of address placed

The screenshot shows the MySQL Workbench interface with the 'fooddelivery' database selected. The 'Schemas' pane on the left shows the database structure. The 'Query' pane displays the following SQL code:

```
51 ALTER TABLE users
52 ADD COLUMN address VARCHAR(255) NOT NULL;
53
54 select *from users;
55
56 select *from orders;
57 select *from order_details;
```

The 'Result Grid' shows the results of the query 'select *from orders;'. The table has 5 columns: id, address, total, and order_date. The data is as follows:

id	address	total	order_date
1	Ambalfur	34.98	2024-11-12 18:01:06
2	Avech	210	2024-11-13 08:15:53
3	New york	120	2024-11-13 10:39:15
4	New york	120	2024-11-13 10:39:56
5	New york	120	2024-11-13 10:42:08
6	New york	120	2024-11-13 10:48:13

The 'Output' pane shows the execution history of the queries:

#	Time	Action	Message	Duration / Fetch
1	20:23:00	select from users LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
2	20:23:36	select from orders LIMIT 0, 1000	16 row(s) returned	0.000 sec / 0.000 sec

Chapter 6: Conclusion

6.1 Conclusion

The development of the Food Delivery Management System represents a significant advancement in streamlining the food delivery process while enhancing efficiency, user satisfaction, and overall operational excellence. By leveraging Java, MySQL, and JDBC for database connectivity, the system achieves its core objectives of facilitating seamless order management, efficient driver assignment, and real-time tracking, catering to the needs of customers, delivery agents, and administrators alike.

The use of Java ensures a robust and secure backend capable of handling complex operations and data transactions. MySQL, combined with JDBC, provides efficient database management, enabling secure storage and quick retrieval of critical data such as orders, customer profiles, and delivery statuses. The integration of these technologies results in a reliable system that offers a user-friendly interface and smooth functionality.

The system successfully meets its functional requirements by enabling features such as customer order placement, driver assignment, real-time delivery tracking, and administrative reporting. Non-functional requirements, including scalability, reliability, and security, further ensure that the system can adapt to increasing demands while maintaining high performance and data integrity.

In summary, the Food Delivery Management System effectively addresses the challenges of managing food delivery services by offering a well-integrated solution. It not only optimizes delivery operations but also enhances the user experience for all stakeholders. The system's design and modular implementation ensure scalability and adaptability, positioning it as a benchmark for similar applications in the food delivery industry.

Chapter 7: References

7.1 References

- [1] [MySQL Documentation] (<https://dev.mysql.com/doc/>)
- [2] [Java Documentation - Oracle] (<https://docs.oracle.com/javase/8/docs/>)
- [3] [JDBC Guide - Oracle] (<https://docs.oracle.com/javase/tutorial/jdbc/>)
- [4] [Food Delivery Management System Design and Development – ResearchGate](<https://www.researchgate.net/>)
- [5] [Software Engineering Practices - IEEE Xplore] (<https://ieeexplore.ieee.org/>)
- [6] [Database Normalization Principles - W3Schools] (https://www.w3schools.com/sql/sql_normalization.asp)
- [7] [Visual Studio Code Documentation] (<https://code.visualstudio.com/docs>)
- [8] [Java Programming and Problem Solving - GeeksforGeeks] (<https://www.geeksforgeeks.org/>)
- [9] [Building Scalable Database Applications - Medium](<https://medium.com/>)
- [10] [Relational Database Design Principles – TutorialsPoint] (<https://www.tutorialspoint.com/sql/sql-tutorial.htm>)

These references provide foundational knowledge about the technologies and methodologies used in your project, offering detailed explanations and best practices.