PES UNIVERSITY
Department of Computer Science and Engineering
UE22CS341A: Software Engineering

Software Engineering Project Report
ON
Online Food Delivery System

SUBMITTED BY

Krishna K R (PES1UG22CS293)
Jathin Shetty N (PES1UG22CS249)

TABLE OF CONTENTS

PROPOSAL OF THE PROJECT

Introduction

The FoodFast Online Food Delivery Management System is an advanced software solution designed to modernize and streamline food ordering and delivery operations. By integrating various functionalities into a single platform, it addresses the multifaceted challenges of digital food service, connecting restaurants, delivery partners, and customers seamlessly.

Core Objectives

1. **Operational Optimization:**
    - **Streamline food ordering and delivery management processes**

- **Optimize delivery routes and restaurant partnerships**

2. **Real-Time Tracking:**
    - **Provide real-time location updates for food deliveries**
    - **Alert users about order status and estimated delivery times**
3. **User Experience Enhancement:**
    - **Offer intuitive interfaces for customers, restaurants, and delivery partners**
    - **Provide transparent order tracking and notifications**
4. **Restaurant and Delivery Partner Management:**
    - **Facilitate easy onboarding and performance tracking**
    - **Implement rating and review systems**

Key Features

1. **Order Management:**
    - **Create, modify, and cancel food orders**
    - **Support multiple payment methods**
2. **Restaurant Catalog:**
    - **Display restaurant menus with detailed item descriptions**
    - **Enable menu filtering and search functionality**
3. **Delivery Tracking:**

    - **Real-time GPS tracking of delivery partners**
    - **Generate estimated delivery times**

4. **User Reviews and Ratings:**
    - **Allow customers to rate restaurants and delivery partners**
    - **Provide feedback mechanisms**

5. **Recommendation System:**
    - **Suggest restaurants based on user preferences**
    - **Implement personalized food recommendations**

System Architecture

- **Web and mobile application platforms**
- **Backend powered by cloud-based infrastructure**
- **Microservices architecture for scalability**
- **Integration with payment gateways and mapping services**

Expected Outcomes

1. **Enhanced User Convenience:**
    - **Simplified food ordering process**
    - **Transparent delivery tracking**

2. **Improved Restaurant Visibility:**

- **Expanded customer reach**
- **Detailed performance analytics**

3. Efficient Delivery Management:

- **Optimized delivery routes**
- **Real-time partner performance monitoring**

4. Data-Driven Insights:

- **Comprehensive user and restaurant analytics**
- **Personalized marketing opportunities**

Conclusion

The FoodFast Online Food Delivery Management System will transform food service delivery, combining technology, user experience, and operational efficiency to address modern dining challenges.

SOFTWARE REQUIREMENT SPECIFICATION

Table of Contents

1. Introduction

1.1 Purpose

This document specifies the software requirements for the Online Food Delivery Management System. The system is designed to manage food orders, restaurant partnerships, delivery tracking, and user interactions.

1.2 Intended Audience

This document is intended for the development team, project managers, and testers involved in the project.

1.3 Product Scope

The Online Food Delivery Management System will provide a comprehensive solution for ordering food, tracking deliveries, managing restaurant partnerships, and facilitating seamless interactions between customers, restaurants, and delivery partners.

1.4 References

- **JathinShetty/Online-Food-Delivery-System**

1.5 Overview

The document details functional and non-functional requirements, system features, interface requirements, and additional specifications.

2. Overall Description

2.1 Product Perspective

A new, self-contained system designed to integrate with existing restaurant point-of-sale systems and payment gateways.

2.2 Product Functions

- **Manage food ordering process**
- **Track real-time food deliveries**
- **Manage restaurant and delivery partner profiles**
- **Process payments**
- **Generate user and restaurant analytics**

2.3 User Classes and Characteristics

- **Customers: Primary users ordering food**
- **Restaurants: Food service providers**
- **Delivery Partners: Individuals responsible for food delivery**
- **System Administrators: Manage platform operations**
- **Customer Support: Handle user queries and issues**

2.4 Operating Environment

- **Web and mobile application**
- **Backend: Cloud-based infrastructure**
- **Database: PostgreSQL/MongoDB**
- **Supported platforms: iOS, Android, Web browsers**

2.5 Design and Implementation Constraints

- **Compliance with food safety regulations**
- **Secure payment processing**
- **Data privacy standards**
- **Real-time geolocation tracking**

2.6 Assumptions and Dependencies

- **Continuous internet connectivity**
- **GPS-enabled devices for delivery tracking**
- **Integration with third-party payment gateways**

3. External Interface Requirements

3.1 User Interfaces

- **Interactive restaurant menus**
- **Order tracking dashboard**
- **User profile management**
- **Payment interface**

3.2 Software Interfaces

- **Payment gateway integration**
- **Mapping and geolocation services**
- **Restaurant inventory management systems**

3.3 Communications Interfaces

- **HTTPS for secure web communication**
- **WebSocket for real-time updates**
- **Push notifications**

4. System Features

4.1 Order Management

- **Create, modify, and cancel food orders**
- **Support multiple payment methods**
- **Generate order receipts**

4.2 Restaurant and Menu Management

- **Display real-time restaurant availability**
- **Update menu items and pricing**
- **Manage restaurant profiles**

4.3 Delivery Tracking

- **Real-time GPS tracking**
- **Estimated delivery time calculations**
- **Delivery partner assignment algorithms**

4.4 User Rating and Feedback System

- **Rate restaurants and delivery partners**
- **Submit detailed reviews**
- **Aggregate and display performance metrics**

4.5 Recommendation Engine

- **Personalized restaurant suggestions**
- **Dietary preference matching**
- **Historical order analysis**

5. Nonfunctional Requirements

5.1 Performance Requirements

- **Response time under 3 seconds**
- **Support 1000 concurrent users**
- **99.9% system uptime**

5.2 Safety Requirements

- **Food safety information display**
- **Allergen tracking**
- **Contactless delivery options**

5.3 Security Requirements

- **End-to-end encryption**
- **Multi-factor authentication**
- **Compliance with data protection regulations**

5.4 Software Quality Attributes

- **Intuitive user interface**
- **Scalable microservices architecture**
- **Cross-platform compatibility**

6. Other Requirements

6.1 Database Requirements

- **Scalable cloud database**
- **Regular data backups**
- **Performance optimization**

6.2 Business Rules

- **Transparent pricing**
- **Fair commission structures**
- **Compliance with local food delivery regulations**

Appendix A: Glossary

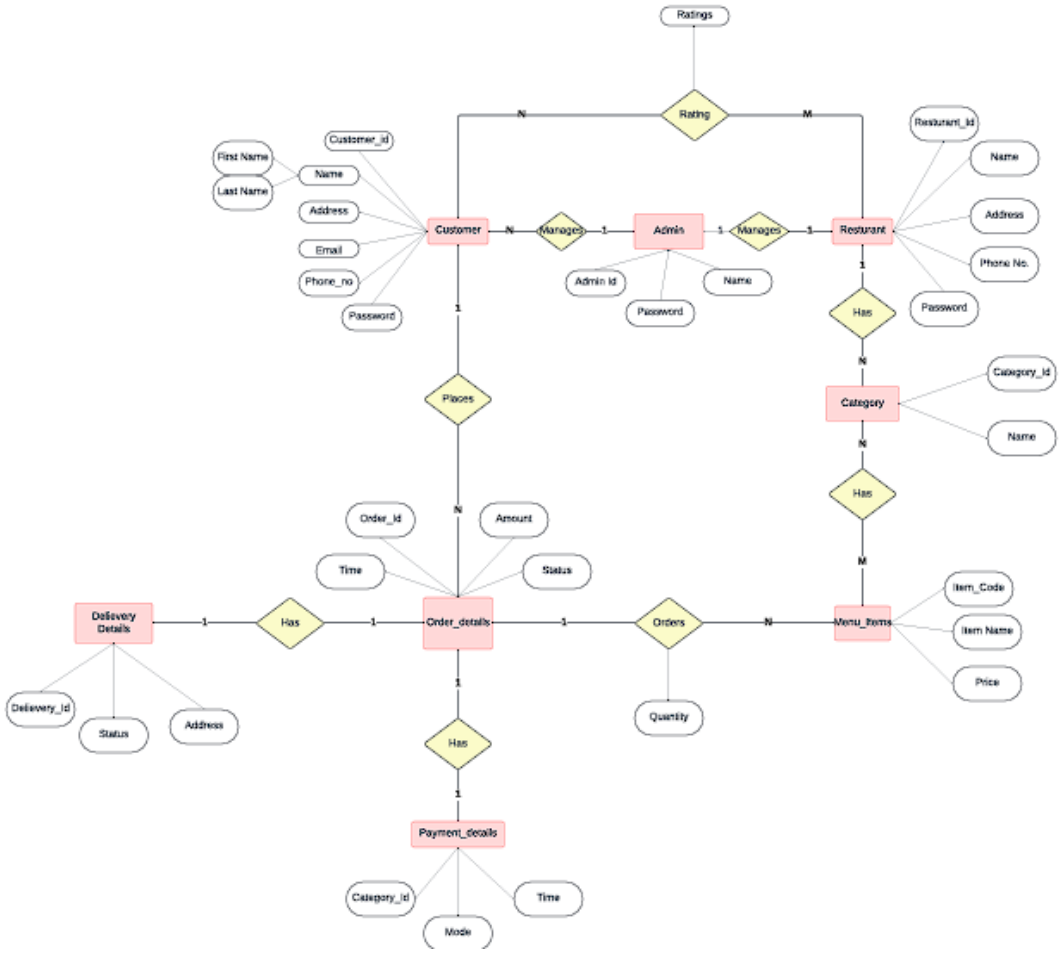
- **GPS: Global Positioning System**
- **API: Application Programming Interface**

Appendix B: Initial ER Diagram Concept

Appendix A: Glossary

GDPR - General Data Protection Regulation

Appendix B: ER Diagram



- **PROJECT PLAN**

- **Project Lifecycle**
- **Agile methodology chosen for project execution due to:**
  - **Incremental development support**
  - **Continuous feedback mechanisms**
  - **Flexibility in feature integration**
  - **Enhanced team collaboration**
- **Agile Implementation Plan**

- **Project Initiation**
  - **Create comprehensive product backlog**
  - **Prioritize features based on business value**
  - **Define initial project scope**
- **Sprint Execution**
  - **1-week sprint duration**
  - **Regular scrum meetings**
  - **Continuous testing and integration**
  - **Feature-driven development**
- **Reviews and Retrospection**
  - **End-of-sprint demonstrations**
  - **Collect stakeholder feedback**
  - **Reflect on sprint performance**
  - **Identify improvement areas**
- **Iterative Deployment**
  - **Refine product backlog**
  - **Reprioritize features**
  - **Continuous improvement cycles**
- **Final Release**
  - **Stabilize core functionalities**
  - **Comprehensive testing**
  - **Post-release maintenance plan**

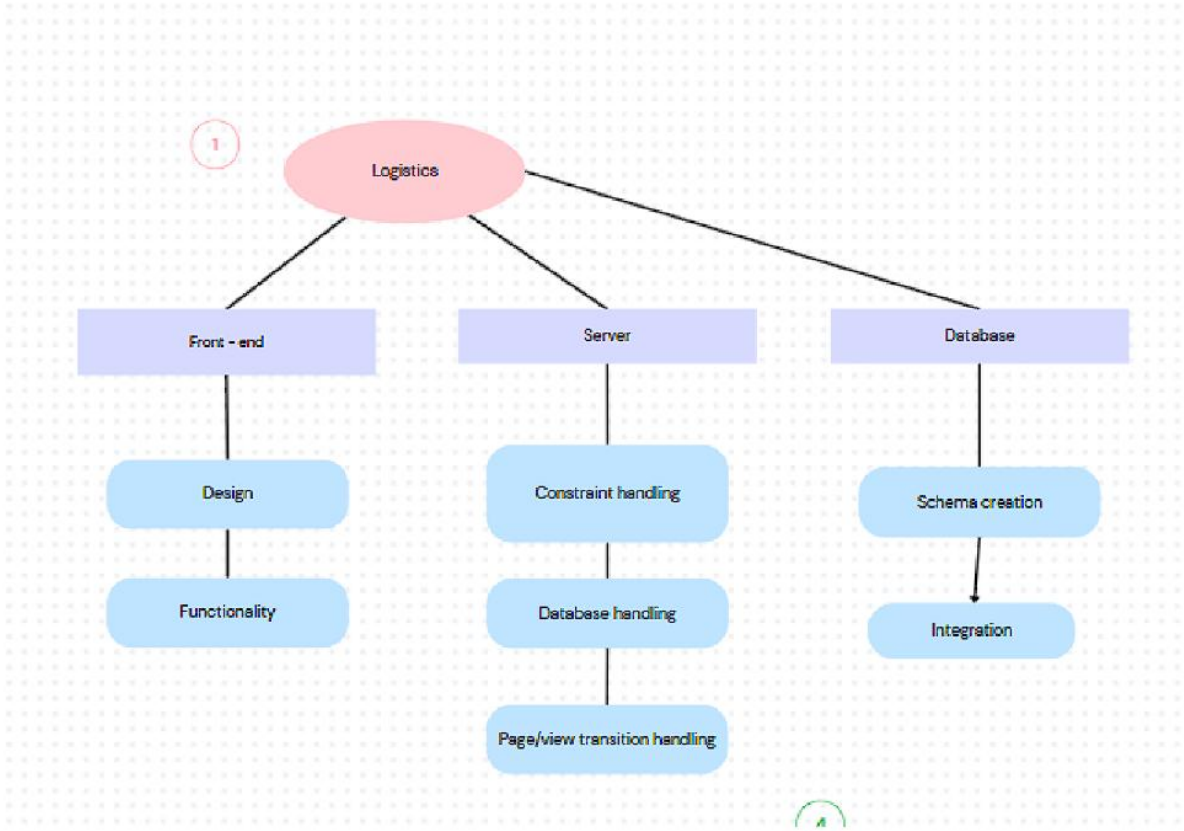
- **Tools Selection**
  - **Planning: Jira**
  - **Design: Figma**
  - **Version Control: GitHub**
  - **Bug Tracking: Jira**
  - **Development IDE: VSCode**
  - **Testing Tools:**
    - **Postman (API testing)**
    - **Selenium (Web testing)**
    - **Jest (Unit testing)**
- **Deliverables Categorization**
- **Reuse Components**
  - **User Authentication System**
  - **Payment Gateway Integration**
  - **Mapping Services**

- **Build Components**
  - **Order Management Module**
  - **Restaurant Catalog System**
  - **Delivery Tracking System**
  - **User Recommendation Engine**
  - **Review and Rating Module**
  - **Notification System**
- **Implementation Plan**
- **Technology Stack**
  - **Backend: Node.js with Express.js**
  - **Frontend: React.js**
  - **Mobile: React Native**
  - **Database: PostgreSQL**
  - **Deployment: AWS/Heroku**
- **1. Backend Setup**
- **API Design**
  - **User management APIs**
  - **Restaurant and menu APIs**
  - **Order processing APIs**
  - **Delivery tracking APIs**
  - **Payment integration endpoints**
- **2. Frontend Setup**
- **Key Features**
  - **Responsive restaurant browsing**
  - **Real-time order tracking**
  - **User profile management**
  - **Payment integration**
  - **Recommendation interfaces**
- **3. Database Design**
- **Key Entities**
  - **Users**
  - **Restaurants**
  - **Menus**
  - **Orders**
  - **Delivery Partners**
  - **Payments**
  - **Reviews**

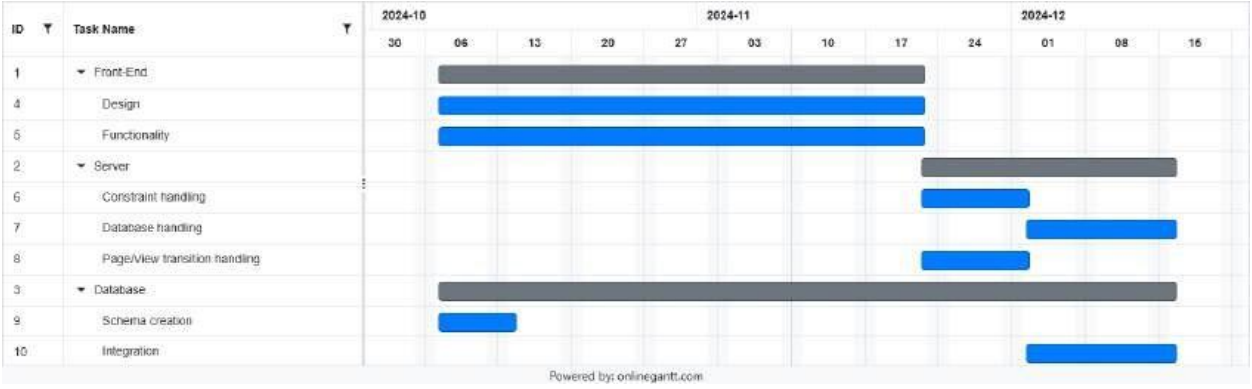
- **4. Version Control Strategy**
  - **Feature branch workflow**
  - **Continuous integration**
  - **Automated testing**
  - **Code review processes**
- **5. Deployment Strategy**
  - **Containerization with Docker**
  - **Microservices architecture**
  - **Continuous deployment**
  - **Scalable cloud infrastructure**

DESIGN DIAGRAMS

Work Breakdown Structure



Gantt Chart



Test Cases

Module: User Registration and Authentication

Test Case ID	Module Name	Test Case Description	Pre-Conditions	Test Steps	Test Data	Expected Results	Actual Results	Test Result
UT_01	User Registration	Test user registration with valid data	Database must be running	1. Connect to the user database 2. Run query to insert new user 3. Verify output	INSERT INTO users (username, email, password) VALUES (jathin', 'jathin@example.com', 'hashedpassword123');	New user should be added to the database	As expected	Pass
UT_02	User Authentication	Test user login with valid credentials	Database must be populated with user data	1. Connect to the database 2. Run query to fetch details 3. Verify credentials	SELECT * FROM users WHERE username = jathin'' AND password = 'hashedpassword123';	New user should be added to the database	As expected	Pass

Module: Restaurant Management

Test Case ID	Module Name	Test Case Description	Pre-Conditions	Test Steps	Test Data	Expected Results	Actual Results	Test Result

UT_03	Restaurant Listing	Test retrieving all active restaurants	Database must be populated with restaurant data	1. Connect to the restaurant database 2. Run query to fetch	SELECT * FROM restaurants WHERE status = 'active';	List of active restaurants should be displayed	As expected	Pass
				active restaurants 3. Verify output				
UT_04	Menu Management	Test adding a new menu item	Restaurant must exist in the database	1. Connect to the menu database 2. Run query to insert new menu item 3. Verify insertion	INSERT INTO menu_items (restaurant_id, name, price) VALUES (101, 'Margherita Pizza', 1299); ;	New menu item should be added to the database	As expected	Pass

Module: Order Processing

TestCas	**Module Name**	**Test Case Description**	**PreConditions**	**Test Steps**	**Test Data**	**Expected Results**	**Actual Results**	**TestResult**

eID								
UT_05	Order Creation	Test creating a new order	User and restaurant must exist in the database	1. Connect to the order database 2. Run query to create new order 3. Verify order creation	INSERT INTO orders (user_id, restaurant_id, total_amount) VALUES (201, 101, 25.98);	New order should be created with pending status	As expected	Pass
IT_01	Order Status Update	Test updating order status	Order must exist in the database	1. Connect to the order database 2. Run query to	UPDATE orders SET status = 'preparing' WHERE order_id = 301;	Order status should be updated to 'preparing'	As expected	Pass

				update order status 3. Verify order creation				

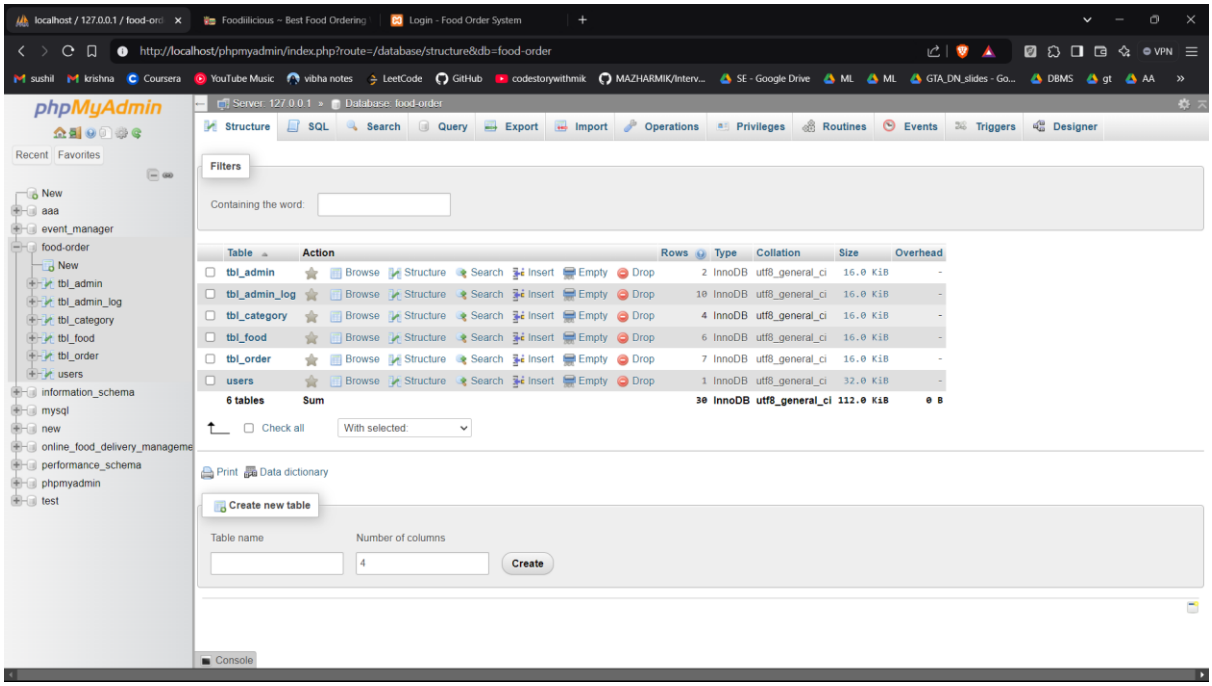
Module Delivery Management

Test Case ID	Module Name	Test Case Description	Pre-Conditions	Test Steps	Test Data	Expected Results	Actual Results	
ST_01	Delivery Assignment	Test assigning delivery to available driver	Order and drivers must exist in the database	1. Connect to the delivery database 2 Run query to find available driver 3. Verify location update	UPDATE deliveries SET driver_id = 401 WHERE order_id = 301;	Order should be assigned to an available driver	As expected	

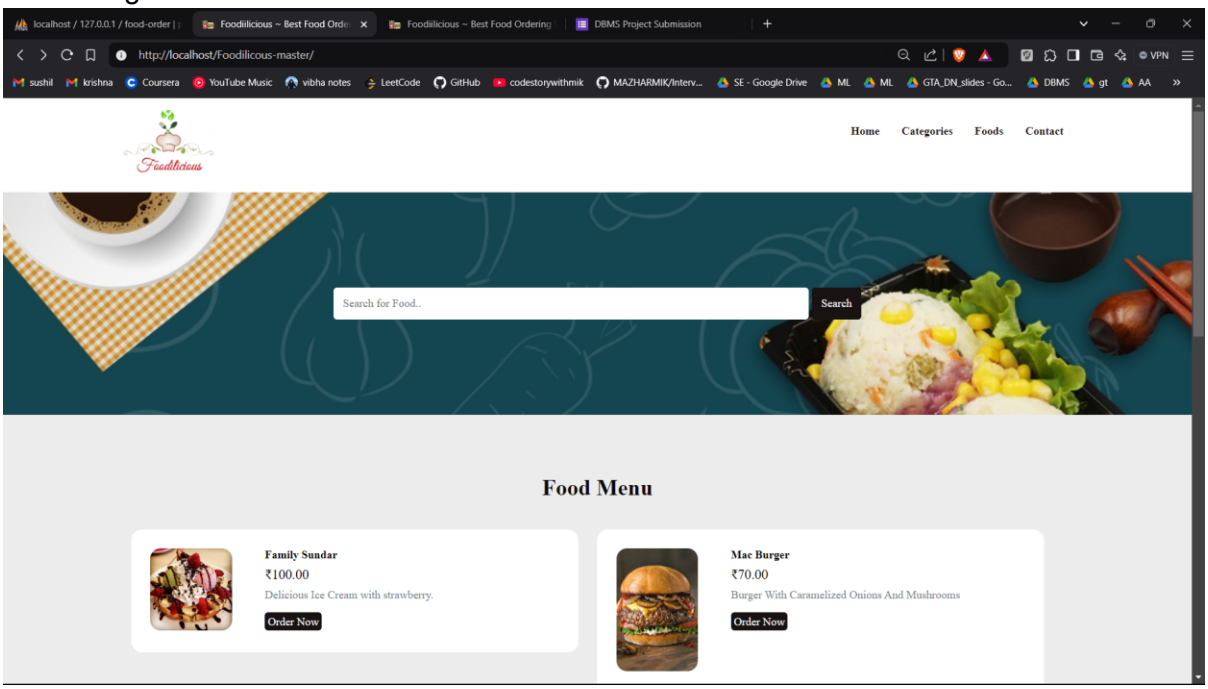
ST_02	Delivery Tracking	Test updating delivery location	Delivery must be in progress	1. Connect to the delivery database 2. Run query to update location 3. Verify location update	UPDATE deliveries SET current_location = 'GPS:40.7128,74.0060' WHERE delivery_id = 501;	Delivery location should be updated in the database	As expected

RESULTS

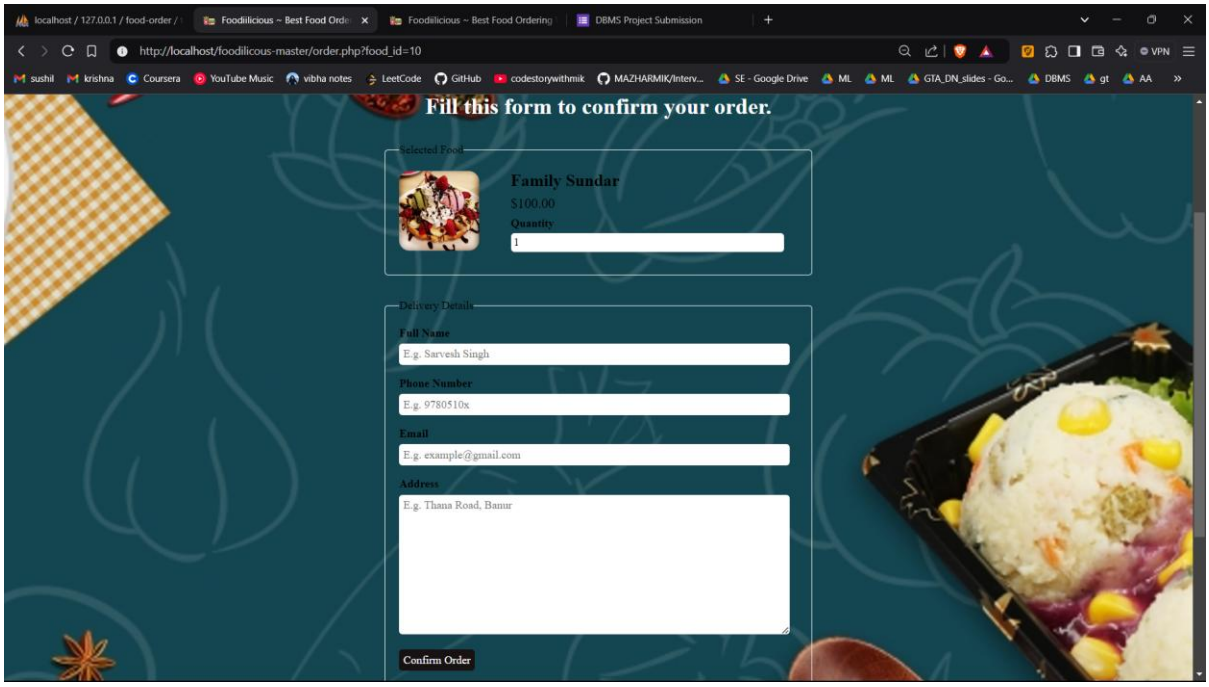
MySQL Database:



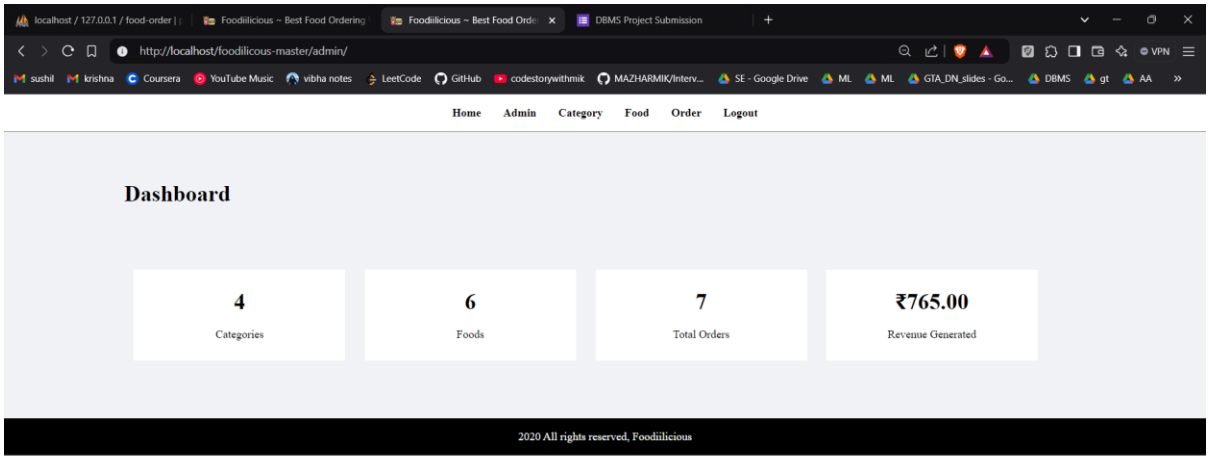
Home Page



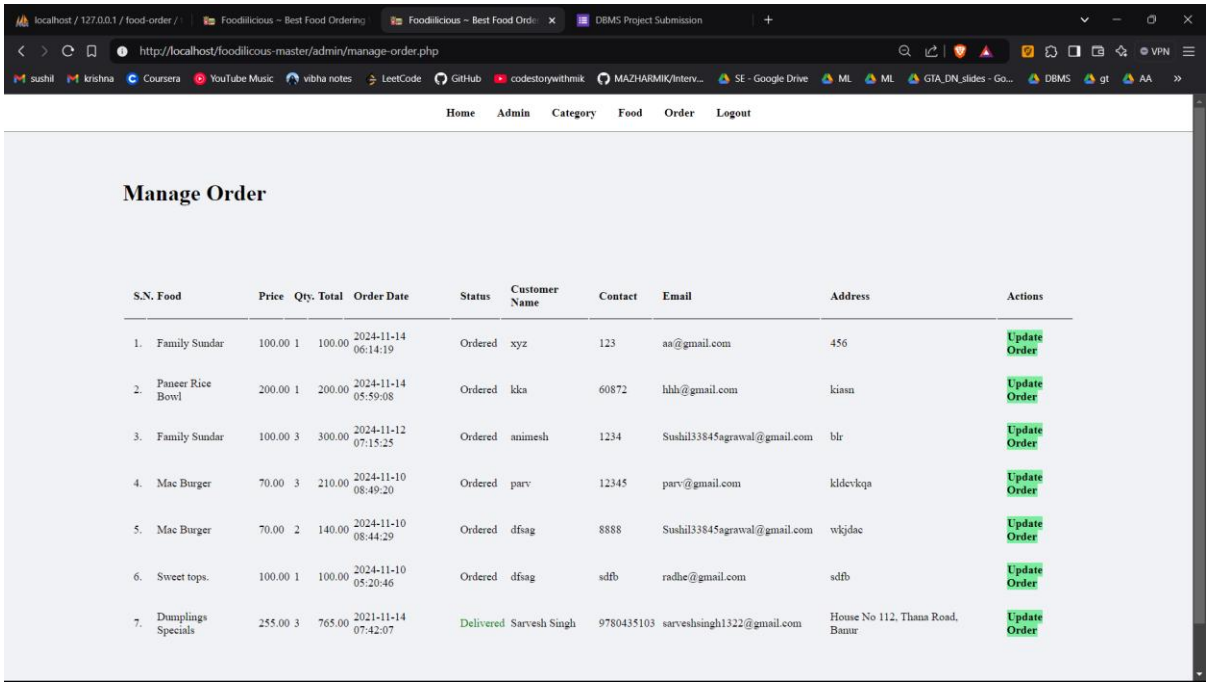
User Food Ordering



Admin DashBoard



Manage Order Page

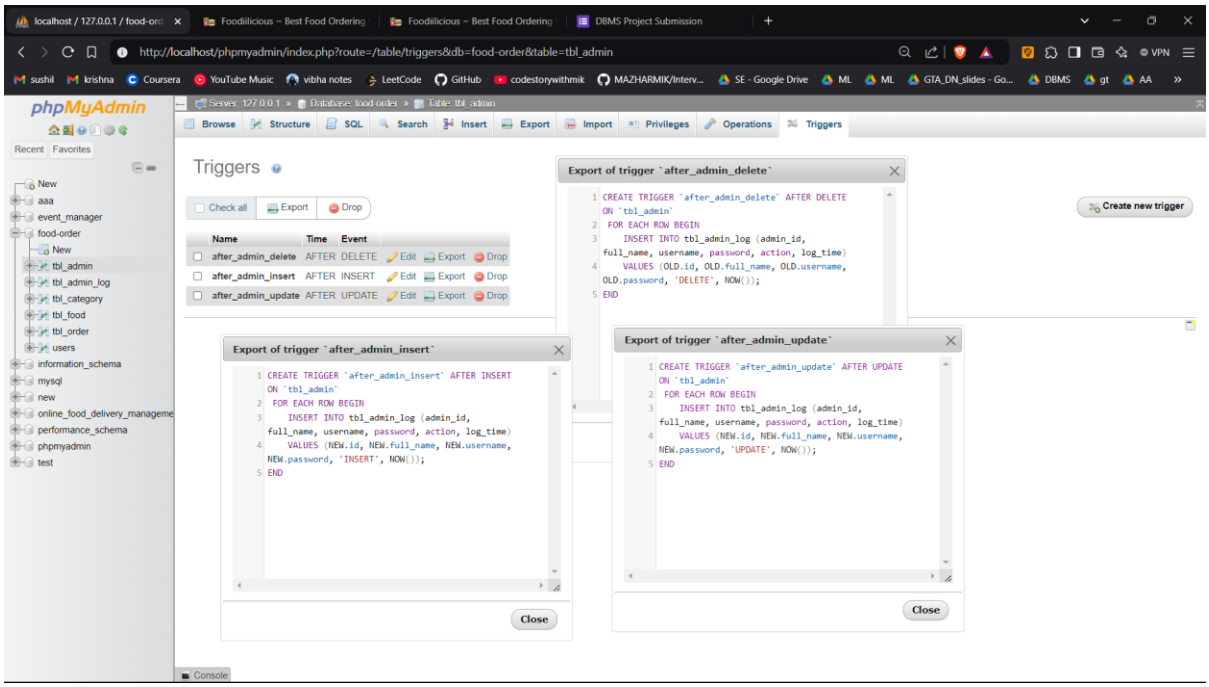


Use of Aggregate Functions of MySQL in Admin DashBoard

```php
<?php
    $sql4 = "SELECT SUM(total) AS Total FROM tbl_order WHERE status='Delivered'";
    $res4 = mysqli_query($conn, $sql4);
    $row4 = mysqli_fetch_assoc($res4);
    $total_revenue = $row4['Total'];

?>
```

Use of Trigger to keep track of admin



Code Snippet for Delete Admin

```php
// 1. get the ID of Admin to be deleted
$id = $_GET['id'];

//2. Create SQL Query to Delete Admin
$sql = "DELETE FROM tbl_admin WHERE id=$id";

//Execute the Query
$res = mysqli_query($conn, $sql);
```

Code Snippet for ADD Admin

```php
if(isset($_POST['submit']))
{
    $full_name = $_POST['full_name'];
    $username = $_POST['username'];
    $password = md5($_POST['password']);
    $sql = "INSERT INTO tbl_admin SET
        full_name='$full_name',
        username='$username',
        password='$password'
    ";
    $res = mysqli_query($conn, $sql) or die(mysqli_error($conn));
```

Code Snippet for Update Admin

```php
if(isset($_POST['submit']))
{
    //echo "Button Clicked";
    //Get all the values from form to update
    $id = $_POST['id'];
    $full_name = $_POST['full_name'];
    $username = $_POST['username'];

    //Create a SQL Query to Update Admin
    $sql = "UPDATE tbl_admin SET
    full_name = '$full_name',
    username = '$username'
    WHERE id='$id'
    ";

    //Execute the Query
    $res = mysqli_query($conn, $sql);
```