



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

Sl.No.	Topic	Page No.
1	Proposal of the Project	3
2	Software Requirement Specification	5
3	Project Plan	12
4	Design Diagrams	16
5	Test Cases	19
6	Results (Screen shots)	25

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.....	3
1.1 Purpose.....	3
1.2 Intended Audience.....	3
1.3 Product Scope.....	3
1.4 References.....	3
1.5 Overview.....	3
2. Overall Description.....	3

2.1 Product Perspective.....	3
2.2 Product Functions.....	3
2.3 User Classes and Characteristics.....	4
2.4 Operating Environment.....	4
2.5 Design and Implementation Constraints.....	4
2.6 Assumptions and Dependencies.....	4
3. External Interface Requirements.....	4
3.1 User Interfaces.....	4
3.2 Software Interfaces.....	5
3.3 Communications Interfaces.....	5
4. System Features.....	5
4.1 Travel Log Management.....	5
4.2 Goods and Passenger Tracking.....	5
4.3 Goods Regulations Enforcement.....	6
4.4 No-Fly List Management.....	6
4.5 Tax Calculation.....	6
5. Nonfunctional Requirements.....	7
5.1 Performance Requirements.....	7
5.2 Safety Requirements.....	7
5.3 Security Requirements.....	7
5.4 Software Quality Attributes.....	7

6. Other Requirements.....	7
6.1 Database Requirements.....	7
6.2 Business Rules.....	7
A: Glossary.....	7
Appendix B: ER Diagram.....	8

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

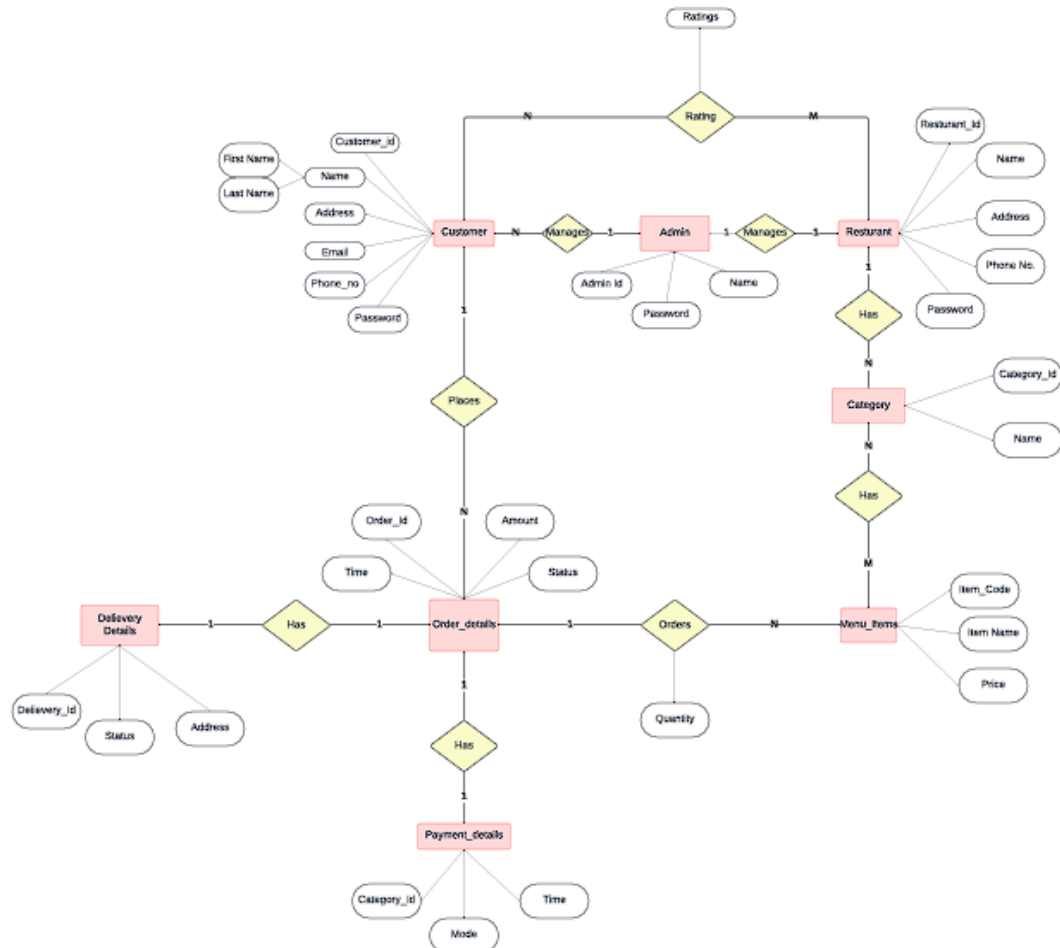
- **GPS: Global Positioning System**
- **API: Application Programming Interface**

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**

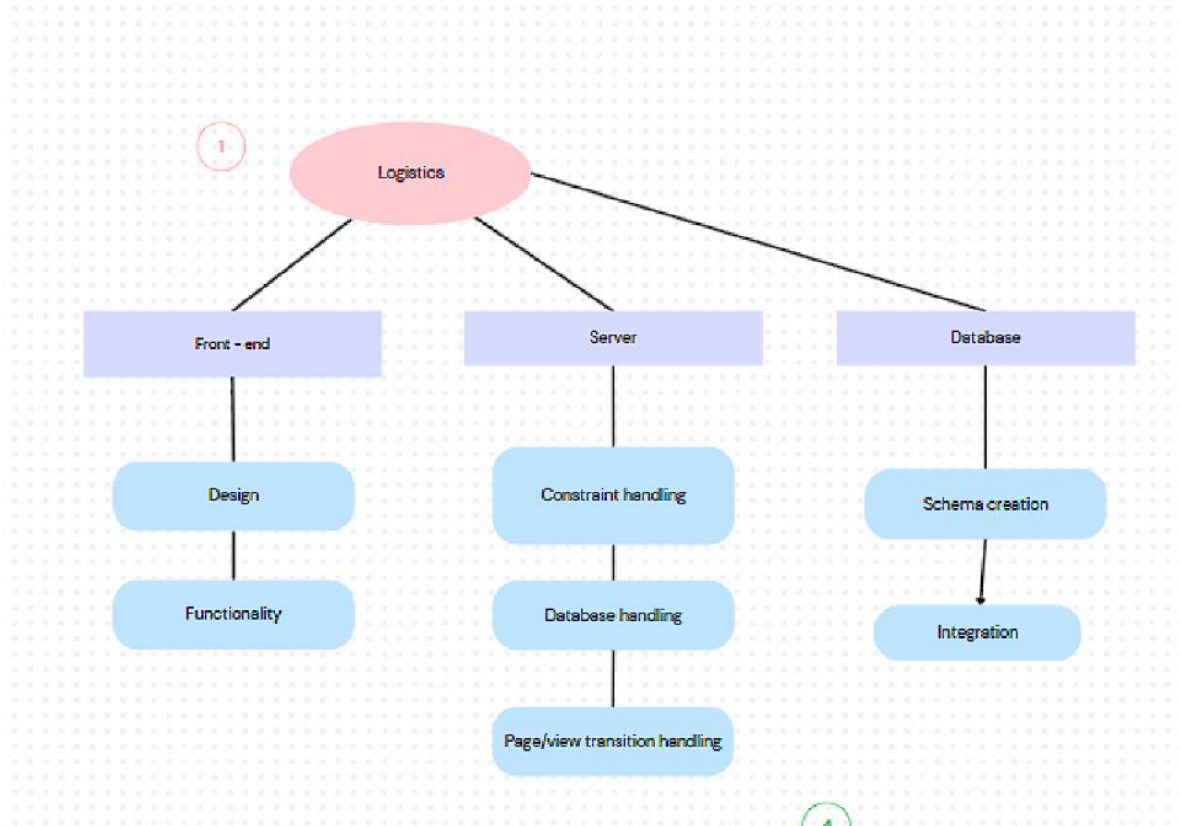
- **Project Initiation**
 - Create comprehensive product backlog
 - Prioritize features based on business value
 - Define initial project scope
- **Sprint Execution**
 - 1-week sprint duration
 - Regular scrum meetings
 - Continuous testing and integration
 - Feature-driven development
- **Reviews and 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**

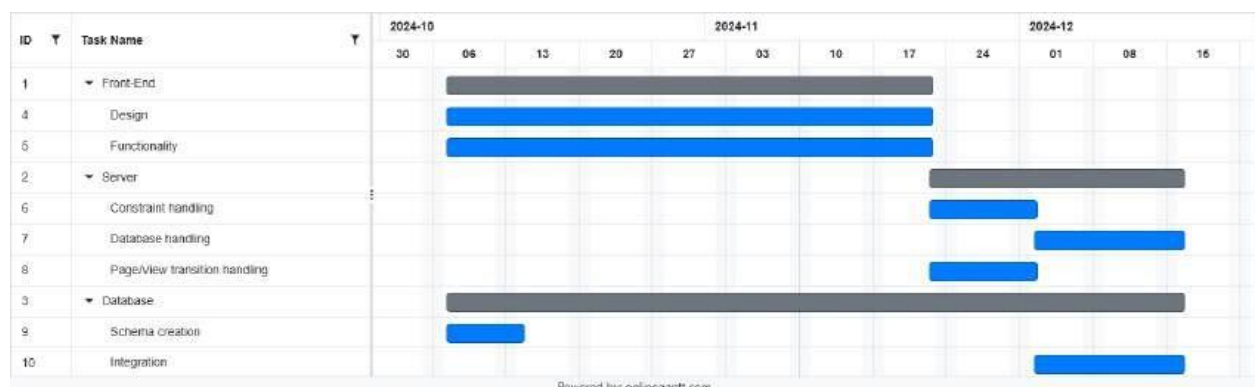
- **4. Version Control Strategy**
 - **Feature branch workflow**
 - **Continuous integration**
 - **Automated testing**
 - **Code review processes**
- **5. Deployment Strategy**
 - **Containerization with Docker**
 - **Microservices architecture**
 - **Continuous deployment**
 - **Scalable cloud infrastructure**

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

Test Cases	Module Name	Test Case Description	PreConditions	Test Steps	Test Data	Expected Results	Actual Results	Test Result
------------	-------------	-----------------------	---------------	------------	-----------	------------------	----------------	-------------

e I D								
U T - 0 5	Ord er Crea tion	Test creatin g a new order	User and resta ra n t must exist in the databas e	1. Con nect to the orde r data base 2. Run quer y to creat e new orde r 3. Verif y order creati on	INSERT INTO orders (user_id, resta ra n t_id, total_am ount) VALUES (201, 101, 25.98);	New order shoul d be create d with pendi ng status	As exp ecte d	P a s s
IT - 0 1	Ord er Stat us Upd ate	Test updati ng order status	Order must exist in the database	1. Con nect to the orde r data base 2. Run quer y to	UPDATE orders SET status = 'preparin g' WHERE order_id = 301;	Order status shoul d be updat ed to 'prepa ring'	As exp ecte d	P a s s

				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	<ol style="list-style-type: none"> 1. Connect to the delivery database 2. Run query to update location 3. Verify location update 	<pre>UPDATE deliveries SET current_location = 'GPS:40.7128,74.0060' WHERE delivery_id = 501;</pre>	Delivery location should be updated in the database	As expected
-------	-------------------	---------------------------------	------------------------------	---	--	---	-------------

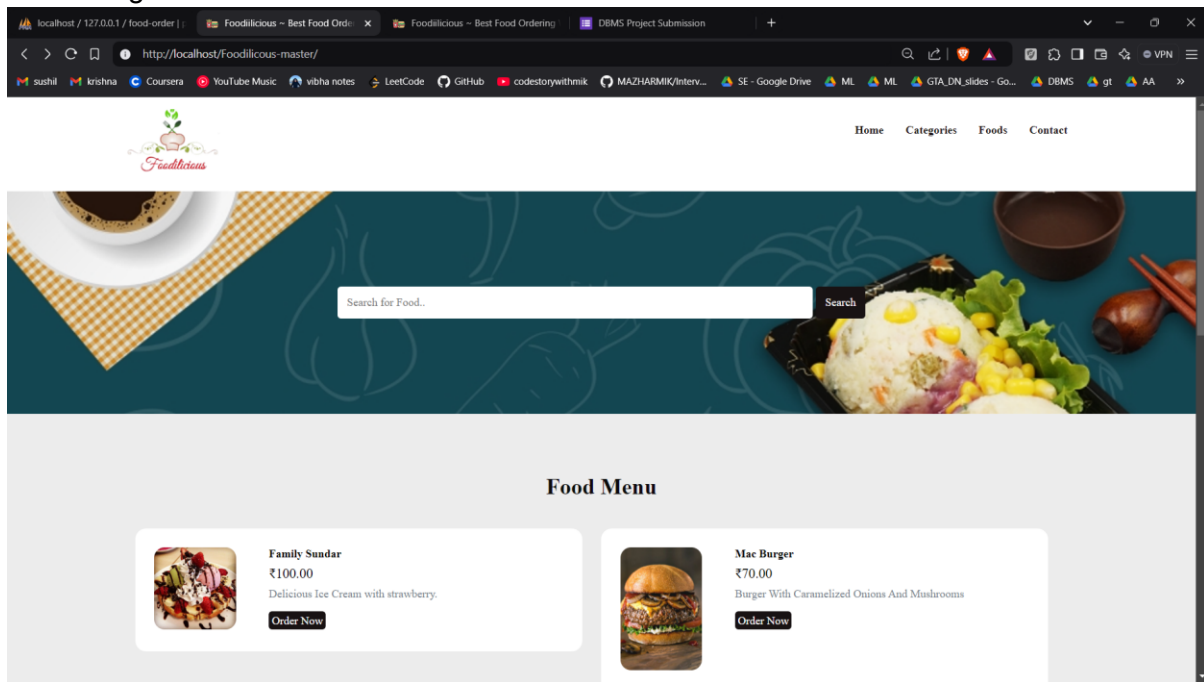
RESULTS

MySQL Database:

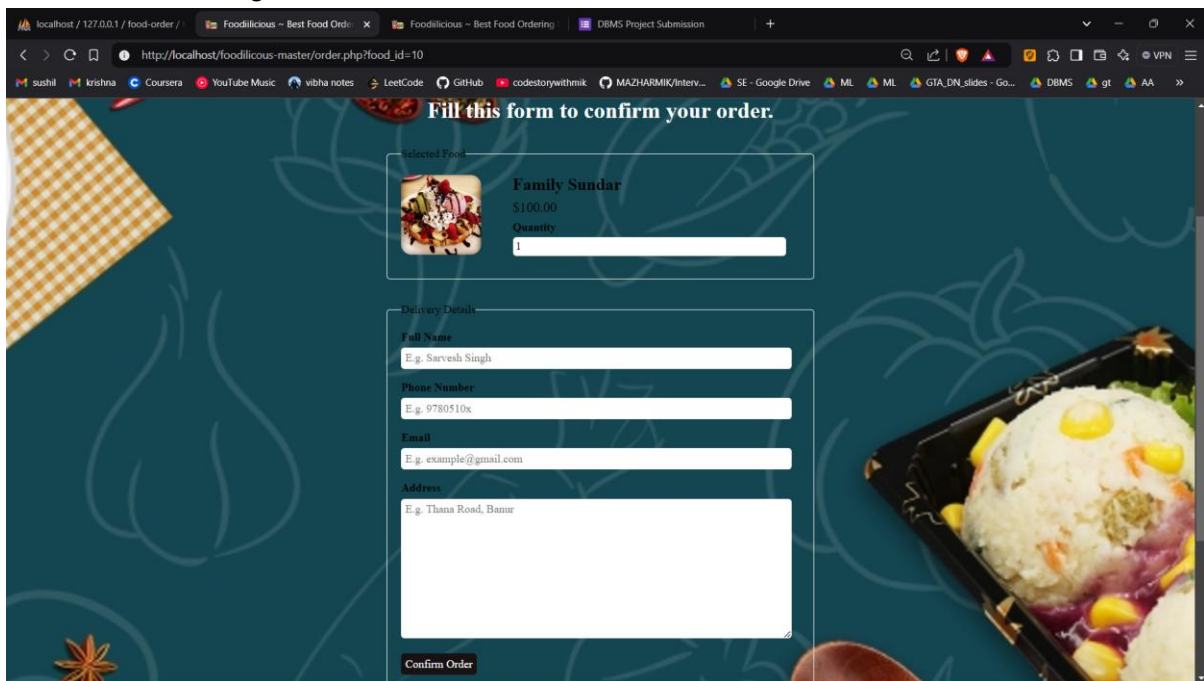
The screenshot displays the phpMyAdmin web interface for a MySQL database named 'food-order'. The left sidebar shows the database structure with a tree view. The main panel shows the 'Structure' tab for the 'food-order' database. A table list is displayed with columns: Table, Action, Rows, Type, Collation, Size, and Overhead. The tables listed are tbi_admin, tbi_admin_log, tbi_category, tbi_food, tbi_order, and users. Below the table list, there is a 'Create new table' section with fields for 'Table name' and 'Number of columns' (set to 4), and a 'Create' button. The bottom of the interface shows a 'Console' tab.

Table	Action	Rows	Type	Collation	Size	Overhead
tbi_admin	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8_general_ci	16.0 K	16.0 K
tbi_admin_log	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8_general_ci	16.0 K	16.0 K
tbi_category	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_general_ci	16.0 K	16.0 K
tbi_food	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8_general_ci	16.0 K	16.0 K
tbi_order	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8_general_ci	16.0 K	16.0 K
users	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	32.0 K	32.0 K
6 tables	Sum	30	InnoDB	utf8_general_ci	112.0 K	112.0 K

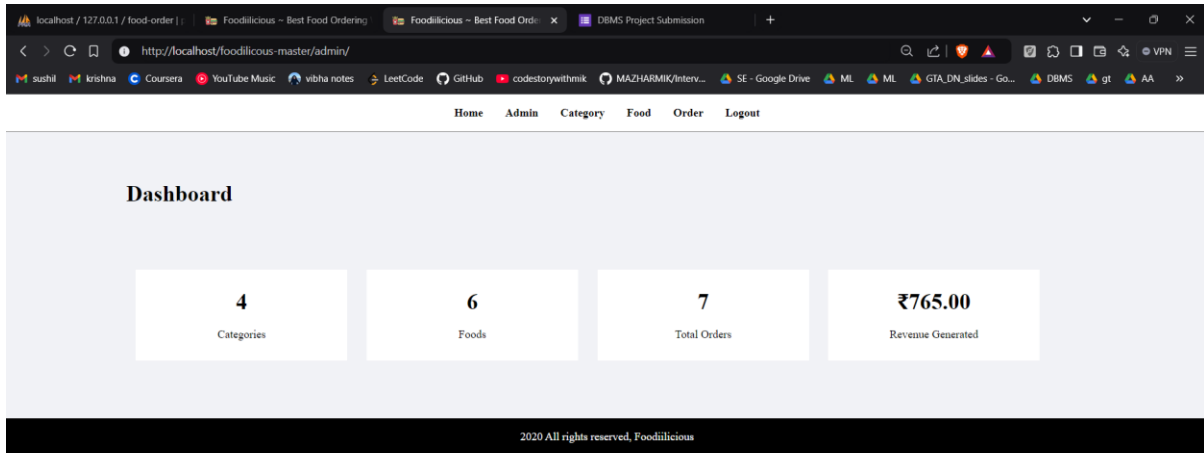
Home Page



User Food Ordering



Admin DashBoard



Manage Order Page

Manage Order

S.N.	Food	Price	Qty	Total	Order Date	Status	Customer Name	Contact	Email	Address	Actions
1.	Family Sundar	100.00	1	100.00	2024-11-14 06:14:19	Ordered	xyz	123	as@gmail.com	456	Update Order
2.	Paneer Rice Bowl	200.00	1	200.00	2024-11-14 05:59:08	Ordered	kka	60872	hbb@gmail.com	kiasu	Update Order
3.	Family Sundar	100.00	3	300.00	2024-11-12 07:15:25	Ordered	animesh	1234	Sushil33845agraval@gmail.com	blr	Update Order
4.	Mac Burger	70.00	3	210.00	2024-11-10 08:49:20	Ordered	parv	12345	parv@gmail.com	kldevkoja	Update Order
5.	Mac Burger	70.00	2	140.00	2024-11-10 08:44:29	Ordered	dfsag	8888	Sushil33845agraval@gmail.com	wkjdac	Update Order
6.	Sweet tops.	100.00	1	100.00	2024-11-10 05:20:46	Ordered	dfsag	sdfb	radhe@gmail.com	sdfb	Update Order
7.	Dumplings Specials	255.00	3	765.00	2021-11-14 07:42:07	Delivered	Sarvesh Singh	9780435103	sarveshshingh1322@gmail.com	House No 112, Thana Road, Banur	Update Order

Use of Aggregate Functions of MySQL in Admin DashBoard

```
<?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

The screenshot shows the phpMyAdmin interface with the 'tbl_admin' table selected. Three trigger export windows are open, showing the following SQL code:

- Export of trigger 'after_admin_delete':**

```
1 CREATE TRIGGER `after_admin_delete` AFTER DELETE
  ON `tbl_admin`
  FOR EACH ROW BEGIN
  2 INSERT INTO tbl_admin_log (admin_id,
  3 full_name, username, password, action, log_time)
  4 VALUES (OLD.id, OLD.full_name, OLD.username,
  5 OLD.password, "DELETE", NOW());
  6 END
```
- Export of trigger 'after_admin_insert':**

```
1 CREATE TRIGGER `after_admin_insert` AFTER INSERT
  ON `tbl_admin`
  FOR EACH ROW BEGIN
  2 INSERT INTO tbl_admin_log (admin_id,
  3 full_name, username, password, action, log_time)
  4 VALUES (NEW.id, NEW.full_name, NEW.username,
  5 NEW.password, "INSERT", NOW());
  6 END
```
- Export of trigger 'after_admin_update':**

```
1 CREATE TRIGGER `after_admin_update` AFTER UPDATE
  ON `tbl_admin`
  FOR EACH ROW BEGIN
  2 INSERT INTO tbl_admin_log (admin_id,
  3 full_name, username, password, action, log_time)
  4 VALUES (NEW.id, NEW.full_name, NEW.username,
  5 NEW.password, "UPDATE", NOW());
  6 END
```

Code Snippet for Delete Admin

```
// 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

```
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

```
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);
}
```

