DBMS Mini Project
Smart Bites: Online Food Delivery and Analytics Platform

Member – 1 SRN: PES1UG23AM260

Member – 1 Name: Samarth M Bharadwaj

Member – 2 SRN: PES1UG23AM279

Member – 2 Name: Shambhavi Prasanna Moudgalya

Description about the problem statement

Abstract

The **Online Food Delivery System** is a **web-based database application** that brings together three key stakeholders — **customers, restaurants, and delivery partners**.

- **Customers** can register and maintain profiles, browse a list of restaurants, view menus, select items, place orders, make payments, and provide reviews. They also receive notifications regarding order confirmation, preparation, dispatch, and delivery.

- **Restaurants** can create and manage their menu items, update food availability, set prices, and track active orders. They can also view customer feedback and ratings to improve their services.

- **Delivery Partners** are assigned orders by the system and are responsible for picking food from restaurants and delivering it to customers. They can update delivery status (on the way, delivered, etc.) in real time.

The project showcases full-stack database application development — integrating database modeling, SQL logic, and user interface design — to create a working prototype of a food delivery ecosystem that emphasizes both functionality and data integrity.

Objectives

1. Create a normalized relational schema representing customers, restaurants, cuisines, menu items, orders, order items, payments, delivery partners and reviews.

2. Implement DDL and DML scripts with constraints (PK, FK, UNIQUE, NOT NULL, AUTO_INCREMENT).

3. Implement triggers, stored procedures and functions to automate and encapsulate logic.

4. Build a simple, elegant frontend with animations and a Flask backend that connects to MySQL to perform CRUD operations.

5. Demonstrate analytics: average rating per restaurant, orders per restaurant.

User requirement specification in detail

The system allows users to register, log in, browse restaurants, place food orders, make payments, and submit reviews. It provides analytics for restaurants based on ratings and order data. The system ensures secure user authentication, smooth order management, and real-time partner assignment for an efficient food delivery experience.

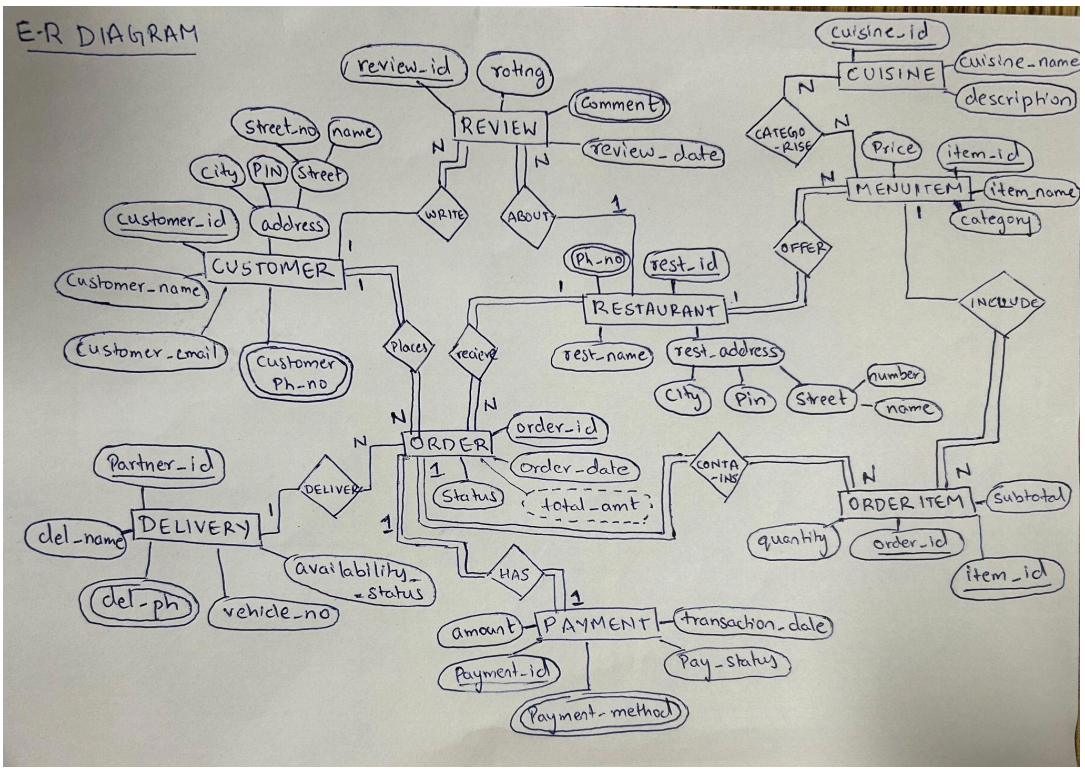
List of Softwares/Tools/Programming languages used

- MySQL for database management
- Flask (Python) for backend logic
- HTML, CSS
- Jinja templates for front-end development
- MySQL Workbench for schema creation
- The system also uses the Werkzeug library for password security and integrates stored procedures for optimized database operations.

ER Diagram

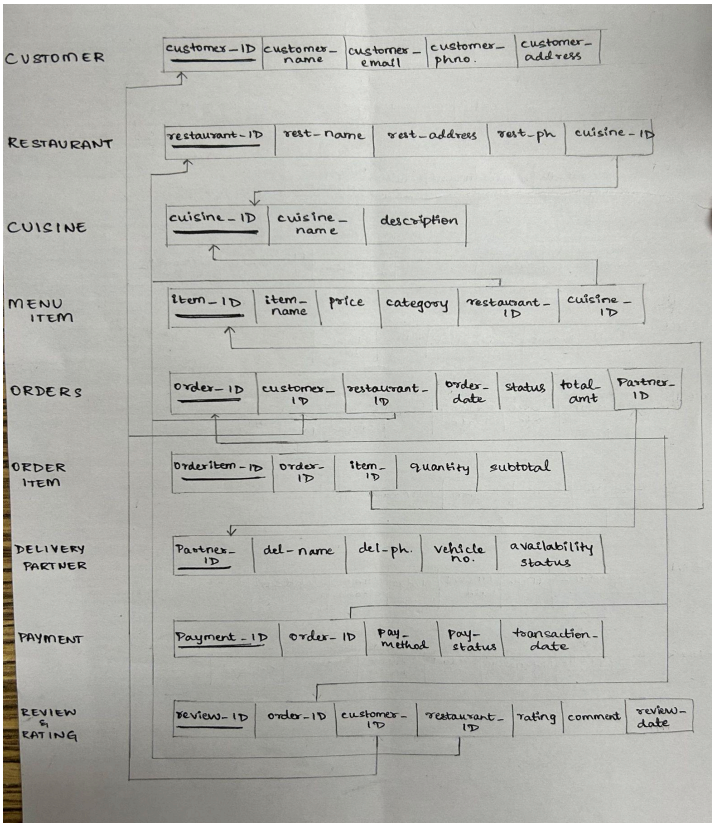
The ER diagram contains these main entities and relationships:

- Customer (composite Address split into street, city, pin)
- Cuisine
- Restaurant (address split)
- MenuItem (belongs to Restaurant and Cuisine)
- DeliveryPartner
- Orders (references Customer, Restaurant, DeliveryPartner)
- OrderItem (references Orders, MenuItem)
- Payment (references Orders)
- Review (references Orders, Customer, Restaurant)



Relational Schema

- Customer(customer_id PK, customer_name, customer_email UNIQUE, customer_phno, customer_street, customer_city, customer_pin)
- Cuisine(cuisine_id PK, cuisine_name, description)
- Restaurant(restaurant_id PK, rest_name, rest_street, rest_city, rest_pin, rest_ph, cuisine_id FK)
- MenuItem(item_id PK, item_name, price, category, restaurant_id FK, cuisine_id FK)
- DeliveryPartner(partner_id PK, del_name, del_ph, vehicle_no, availability_status)
- Orders(order_id PK, customer_id FK, restaurant_id FK, order_date, status, total_amt, partner_id FK)
- OrderItem(orderitem_id PK, order_id FK, item_id FK, quantity, subtotal)
- Payment(payment_id PK, order_id FK, pay_method, pay_status, transaction_date)
- Review(review_id PK, order_id FK, customer_id FK, restaurant_id FK, rating, comment, review_date)



DDL Commands and example screenshots:

```
-- Create Database
CREATE DATABASE IF NOT EXISTS FoodDeliveryDB;
USE FoodDeliveryDB;


-- ========================
--  CUSTOMER
-- ========================
CREATE TABLE Customer (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_name VARCHAR(100),
    customer_email VARCHAR(100) UNIQUE,
    customer_phno VARCHAR(15),
    customer_address VARCHAR(255)
);


-- ========================
--  CUISINE
-- ========================
CREATE TABLE Cuisine (
    cuisine_id INT AUTO_INCREMENT PRIMARY KEY,
    cuisine_name VARCHAR(100),
```

```
    description VARCHAR(255)
);


-- ========================
--  RESTAURANT
-- ========================
CREATE TABLE Restaurant (
    restaurant_id INT AUTO_INCREMENT PRIMARY KEY,
    rest_name VARCHAR(100),
    rest_address VARCHAR(255),
    rest_ph VARCHAR(15),
    cuisine_id INT,
    FOREIGN KEY (cuisine_id) REFERENCES
Cuisine(cuisine_id)
);


-- ========================
--  MENU ITEM
-- ========================
CREATE TABLE MenuItem (
    item_id INT AUTO_INCREMENT PRIMARY KEY,
    item_name VARCHAR(100),
```

```
    price DECIMAL(8,2),
    category VARCHAR(100),
    restaurant_id INT,
    cuisine_id INT,
    FOREIGN KEY (restaurant_id) REFERENCES Restaurant(restaurant_id),
    FOREIGN KEY (cuisine_id) REFERENCES Cuisine(cuisine_id)
);

-- ========================
--  DELIVERY PARTNER
-- ========================
CREATE TABLE DeliveryPartner (
    partner_id INT AUTO_INCREMENT PRIMARY KEY,
    del_name VARCHAR(100),
    del_ph VARCHAR(15),
    vehicle_no VARCHAR(20),
    availability_status VARCHAR(50)
);

-- ========================
--  ORDERS
```

```
-- ========================

CREATE TABLE Orders (

   order_id INT AUTO_INCREMENT PRIMARY KEY,

   customer_id INT,

   restaurant_id INT,

   order_date DATE,

   status VARCHAR(50),

   total_amt DECIMAL(10,2),

   partner_id INT,

   FOREIGN KEY (customer_id) REFERENCES
Customer(customer_id),

   FOREIGN KEY (restaurant_id) REFERENCES
Restaurant(restaurant_id),

   FOREIGN KEY (partner_id) REFERENCES
DeliveryPartner(partner_id)

);


-- ========================

--  ORDER ITEM

-- ========================

CREATE TABLE OrderItem (

   orderitem_id INT AUTO_INCREMENT PRIMARY KEY,

   order_id INT,
```

```
    item_id INT,
    quantity INT,
    subtotal DECIMAL(8,2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (item_id) REFERENCES MenuItem(item_id)
);

-- ========================
--  PAYMENT
-- ========================
CREATE TABLE Payment (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    pay_method VARCHAR(50),
    pay_status VARCHAR(50),
    transaction_date DATE,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);

-- ========================
--  REVIEW & RATING
-- ========================
```

```
CREATE TABLE Review (
   review_id INT AUTO_INCREMENT PRIMARY KEY,
   order_id INT,
   customer_id INT,
   restaurant_id INT,
   rating INT,
   comment VARCHAR(255),
   review_date DATE,
   FOREIGN KEY (order_id) REFERENCES Orders(order_id),
   FOREIGN KEY (customer_id) REFERENCES
Customer(customer_id),
   FOREIGN KEY (restaurant_id) REFERENCES
Restaurant(restaurant_id)
);


--
================================================================
==========
--  SAMPLE DATA INSERTION (for demo)
--
================================================================
==========


-- Customers
```

INSERT INTO Customer (customer_name, customer_email, customer_phno, customer_address)

VALUES

('Samarth', 'samarth@gmail.com', '9876543210', 'Bangalore'),

('Riya', 'riya@gmail.com', '9988776655', 'Mysore');

-- Cuisine

INSERT INTO Cuisine (cuisine_name, description)

VALUES

('Indian', 'Traditional Indian dishes'),

('Italian', 'Pasta, Pizza, and more');

-- Restaurants

INSERT INTO Restaurant (rest_name, rest_address, rest_ph, cuisine_id)

VALUES

('SpiceHub', 'Koramangala, Bangalore', '9876501234', 1),

('LaPasta', 'Indiranagar, Bangalore', '9865432109', 2);

-- Menu Items

INSERT INTO MenuItem (item_name, price, category, restaurant_id, cuisine_id)

VALUES

('Butter Chicken', 250.00, 'Main Course', 1, 1),

('Paneer Tikka', 180.00, 'Starter', 1, 1),

('Margherita Pizza', 350.00, 'Main Course', 2, 2),

('Pasta Alfredo', 300.00, 'Main Course', 2, 2);

-- Delivery Partners

INSERT INTO DeliveryPartner (del_name, del_ph, vehicle_no, availability_status)

VALUES

('Rahul', '9898989898', 'KA05AB1234', 'Available'),

('Asha', '9797979797', 'KA03XY4567', 'Available');

-- Orders

INSERT INTO Orders (customer_id, restaurant_id, order_date, status, total_amt, partner_id)

VALUES

(1, 1, '2025-10-10', 'Delivered', 430.00, 1),

(2, 2, '2025-10-09', 'Delivered', 650.00, 2);

-- Order Items

INSERT INTO OrderItem (order_id, item_id, quantity, subtotal)

VALUES

(1, 1, 1, 250.00),

(1, 2, 1, 180.00),

(2, 3, 1, 350.00),

(2, 4, 1, 300.00);

-- Payment

INSERT INTO Payment (order_id, pay_method, pay_status, transaction_date)

VALUES

(1, 'UPI', 'Success', '2025-10-10'),

(2, 'Credit Card', 'Success', '2025-10-09');

-- Review

INSERT INTO Review (order_id, customer_id, restaurant_id, rating, comment, review_date)

VALUES

(1, 1, 1, 5, 'Excellent food and delivery!', '2025-10-10'),

(2, 2, 2, 4, 'Tasty food but slightly late.', '2025-10-09');

--
==========

-- QUICK CHECKS FOR REVIEW

--
==========

SELECT * FROM Customer;

SELECT * FROM Restaurant;

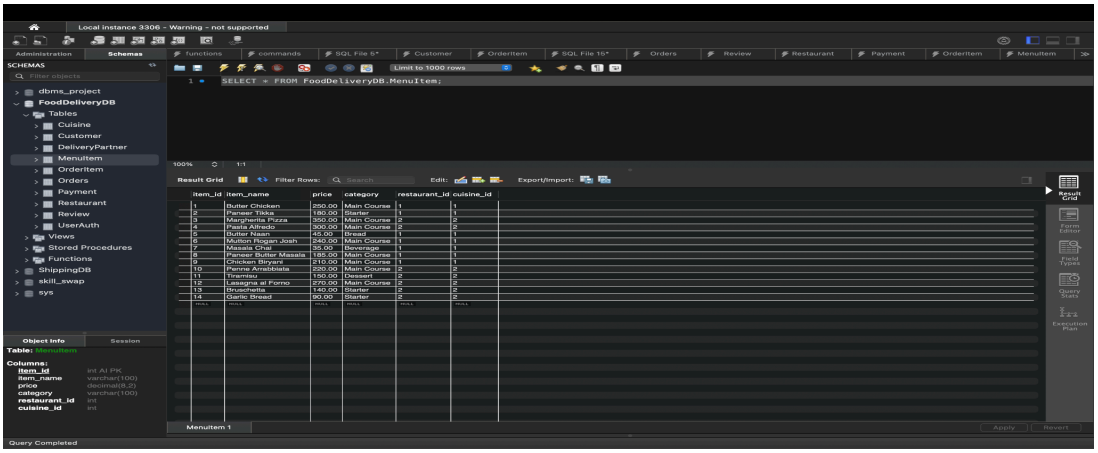
SELECT * FROM MenuItem;

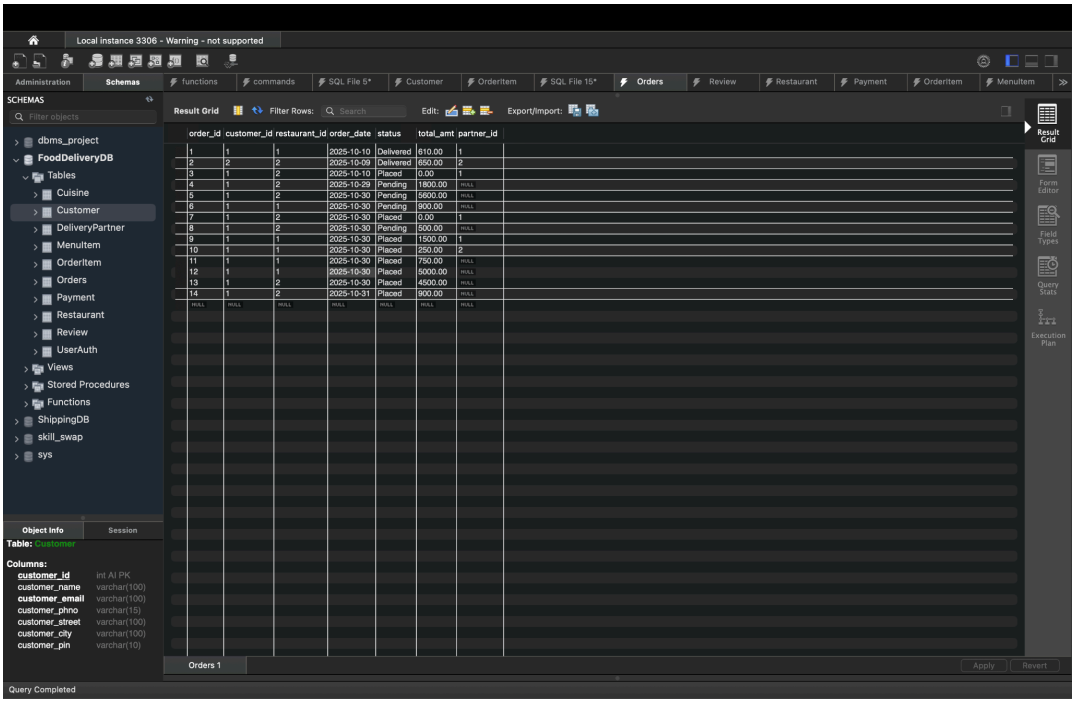
SELECT * FROM Orders;

SELECT * FROM OrderItem;

SELECT * FROM Payment;

SELECT * FROM Review;


Local instance 3306 - Warning - not supported
SELECT * FROM FoodDeliveryDB.MenuItem;
Result Grid
item_id item_name price category restaurant_id cuisine_id
1 Butter Chicken 250.00 Main Course 1 1
2 Paneer Tikka 180.00 Starter 1 1
3 Margherita Pizza 350.00 Main Course 2 2
4 Pasta Alfredo 300.00 Main Course 2 2
5 Butter Naan 45.00 Bread 1 1
6 Mutton Rogan Josh 240.00 Main Course 1 1
7 Masala Chai 35.00 Beverage 1 1
8 Paneer Butter Masala 185.00 Main Course 1 1
9 Chicken Biryani 210.00 Main Course 1 1
10 Penne Arrabbiata 220.00 Main Course 2 2
11 Tiramisu 150.00 Dessert 2 2
12 Lasagna al Forno 270.00 Main Course 2 2
13 Bruschetta 140.00 Starter 2 2
14 Garlic Bread 90.00 Starter 2 2
Table: MenuItem
Columns:
item_id int AI PK
item_name varchar(100)
price decimal(8,2)
category varchar(100)
restaurant_id int
cuisine_id int
Query Completed


Local instance 3306 - Warning - not supported
Result Grid
order_id customer_id restaurant_id order_date status total_amt partner_id
1 1 1 2025-10-10 Delivered 610.00 1
2 2 2 2025-10-09 Delivered 650.00 2
3 1 2 2025-10-10 Placed 0.00 1
4 1 2 2025-10-29 Pending 1800.00 NULL
5 1 2 2025-10-30 Pending 5600.00 NULL
6 1 1 2025-10-30 Pending 900.00 NULL
7 1 2 2025-10-30 Placed 0.00 1
8 1 2 2025-10-30 Pending 500.00 NULL
9 1 1 2025-10-30 Placed 1500.00 1
10 1 1 2025-10-30 Placed 250.00 2
11 1 1 2025-10-30 Placed 750.00 NULL
12 1 1 2025-10-30 Placed 5000.00 NULL
13 1 2 2025-10-30 Placed 4500.00 NULL
14 1 2 2025-10-31 Placed 900.00 NULL
Table: Customer
Columns:
customer_id int AI PK
customer_name varchar(100)
customer_email varchar(100)
customer_phno varchar(15)
customer_street varchar(100)
customer_city varchar(100)
customer_pin varchar(10)
Query Completed

CRUD Operation Screenshots

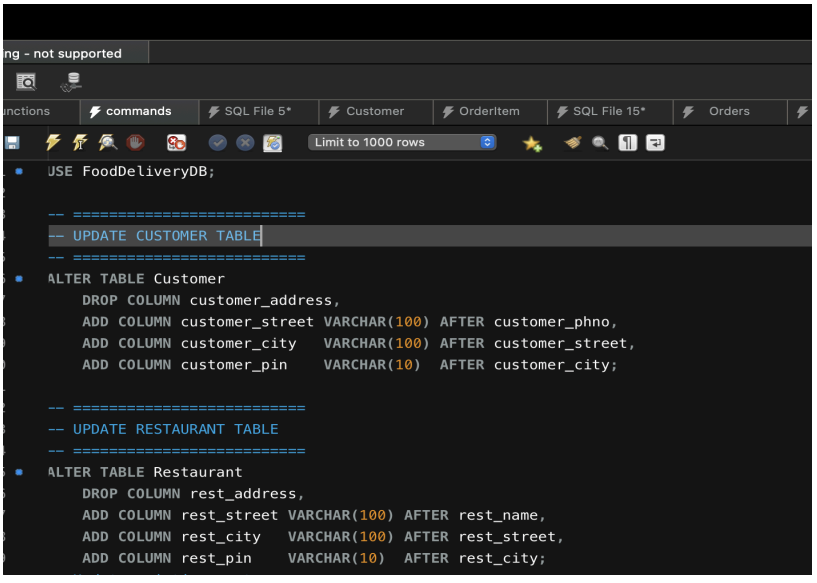
Operation	SQL Example	Description
Create (Insert)	INSERT INTO Customer VALUES (...);	Customer registration via frontend form.
Read (Select)	SELECT * FROM MenuItem WHERE restaurant_id=1;	Fetch menu items when user selects a restaurant.
Update	UPDATE DeliveryPartner SET availability_status='Unavailable' WHERE partner_id=1;	Automatically triggered after order placed.
Delete	DELETE FROM Review WHERE review_id=3;	Admin deletes inappropriate review (if implemented).

```sql
-- Update existing customers
UPDATE Customer
SET customer_street='5th Main', customer_city='Bangalore', customer_pin='560034'
WHERE customer_id=1;

UPDATE Customer
SET customer_street='Church Road', customer_city='Mysore', customer_pin='570001'
WHERE customer_id=2;

-- Update restaurants
UPDATE Restaurant
SET rest_street='Koramangala 6th Block', rest_city='Bangalore', rest_pin='560095'
WHERE restaurant_id=1;

UPDATE Restaurant
SET rest_street='Indiranagar 1st Stage', rest_city='Bangalore', rest_pin='560038'
WHERE restaurant_id=2;
```



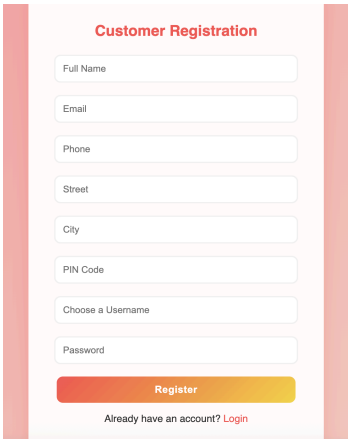
List of Functionalities / Features

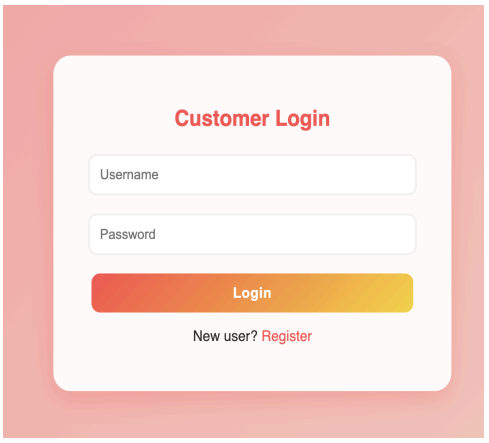
Feature	Description	SQL Involved
Customer Registration & Login	Allows user to create account and sign in	INSERT INTO Customer, SELECT * FROM Customer
Place Order Workflow	Choose restaurant → cuisine → menu → order → payment	SELECT, INSERT INTO Orders, INSERT INTO OrderItem, trigger on OrderItem
Review Restaurant	Customer submits star rating and comments.	INSERT INTO Review
Analytics Dashboard	Displays average ratings and total orders per restaurant**.**	SELECT AvgRestaurantRating(restaurant_id) + aggregate queries
Delivery Partner Assignment	Auto-updates partner availability after order placed.	UPDATE DeliveryPartner SET availability_status='Unavailable'

Frontend Design

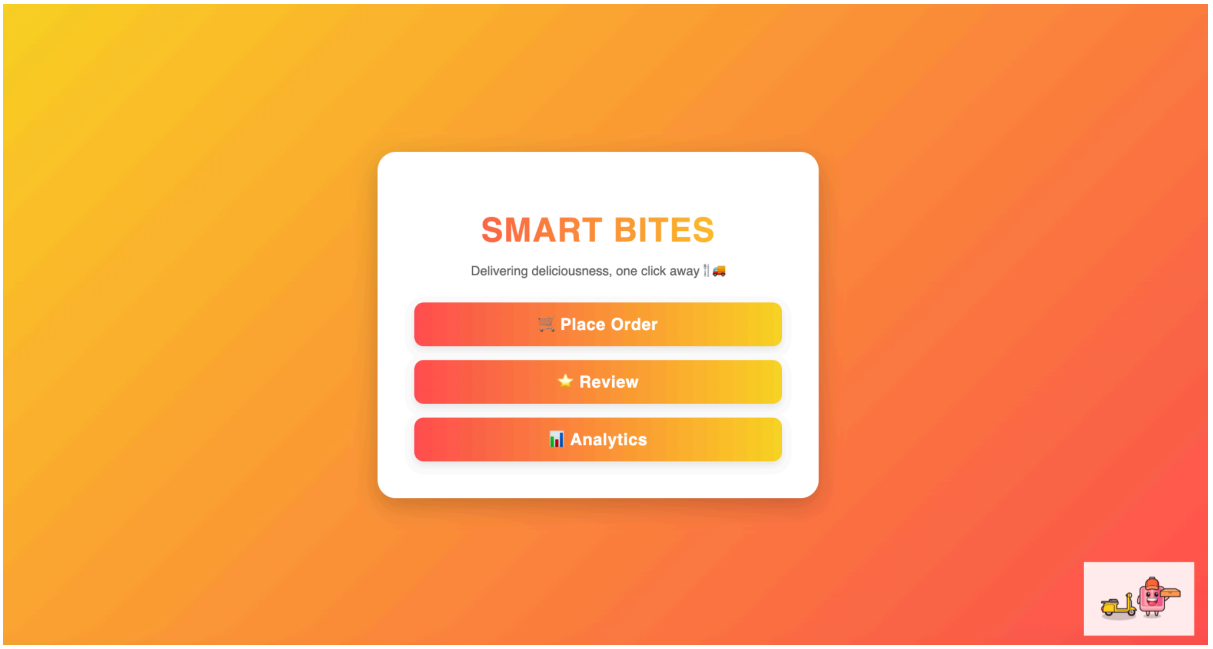
Pages & Components

1. **Login / Register**
    - Clean card layout, fade-in animation
    - Fields: name (register), email, password

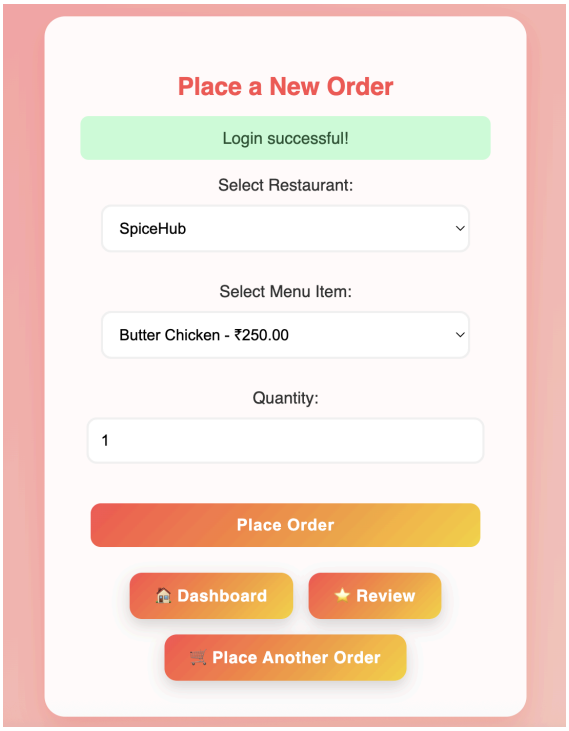


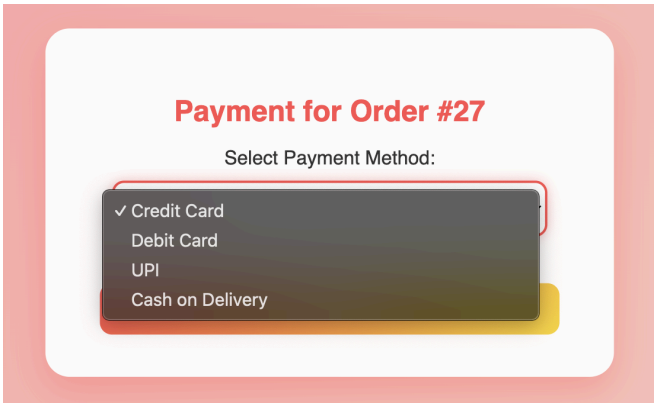


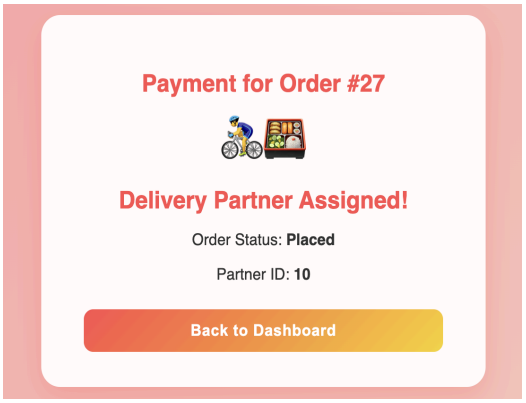
2. **Dashboard**
    - Large action buttons: Place Order, Review Restaurant, Analytics (slide-in)



3. **Place Order Flow**
    - Step 1: Select Restaurant (list/grid)
    - Step 2: Select Cuisine (filters)
    - Step 3: Select Menu Items (Add to Cart)
    - Step 4: Review Cart & Place Order (confirm)
    - Step 5: Payment (dummy flow) and confirmation
    - On confirmation: backend calls to stored procedure + insert order items + payment; assign delivery partner

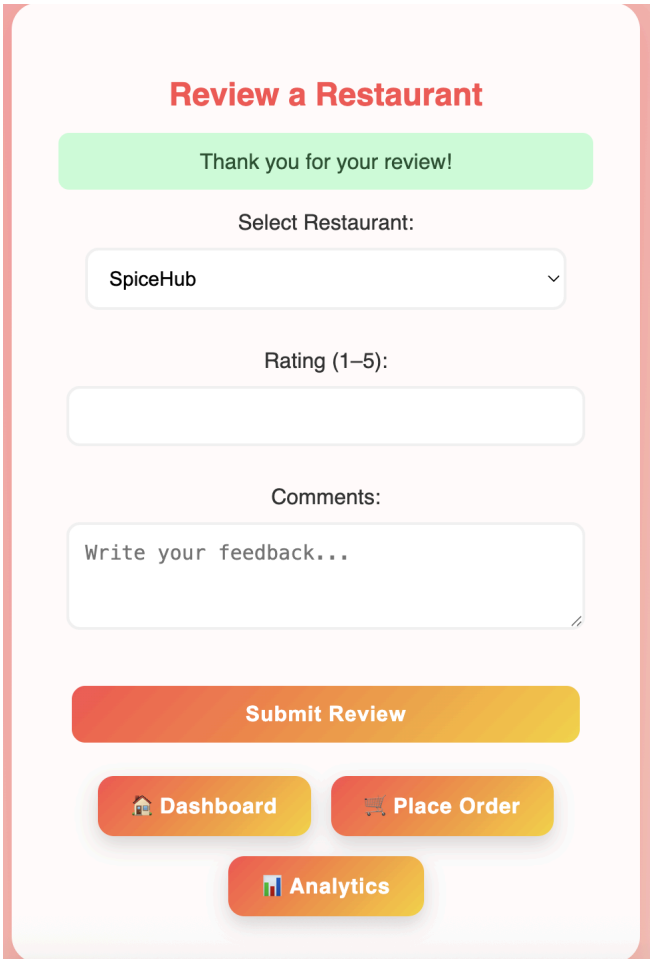






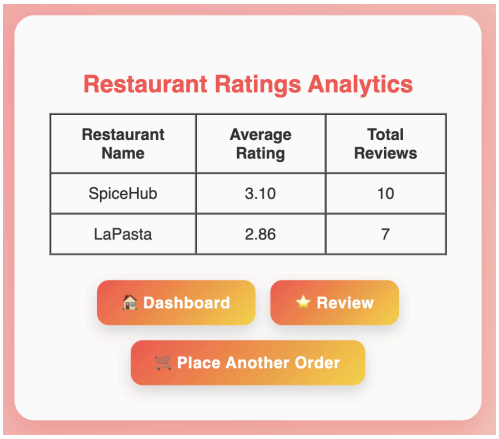
4. **Review Restaurant**

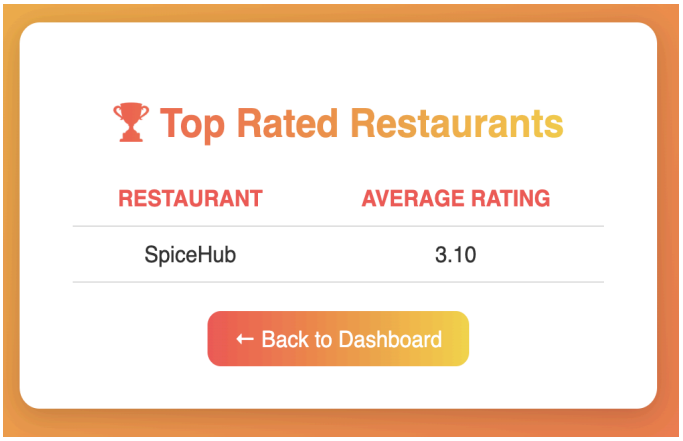
    - Select restaurant, star-rating control, comment box, submit



5. **Analytics**

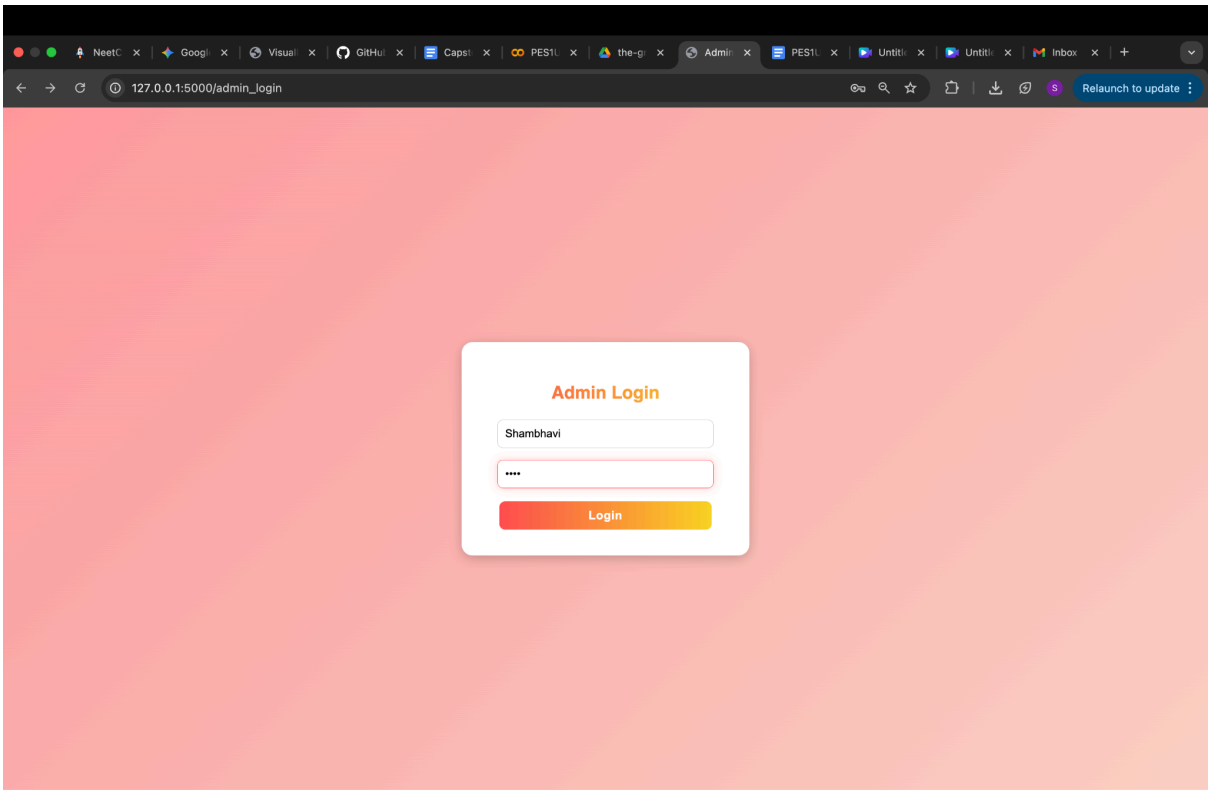
    - Card with average rating per restaurant (uses function AvgRestaurantRating)
    - Bar / table showing orders count per restaurant



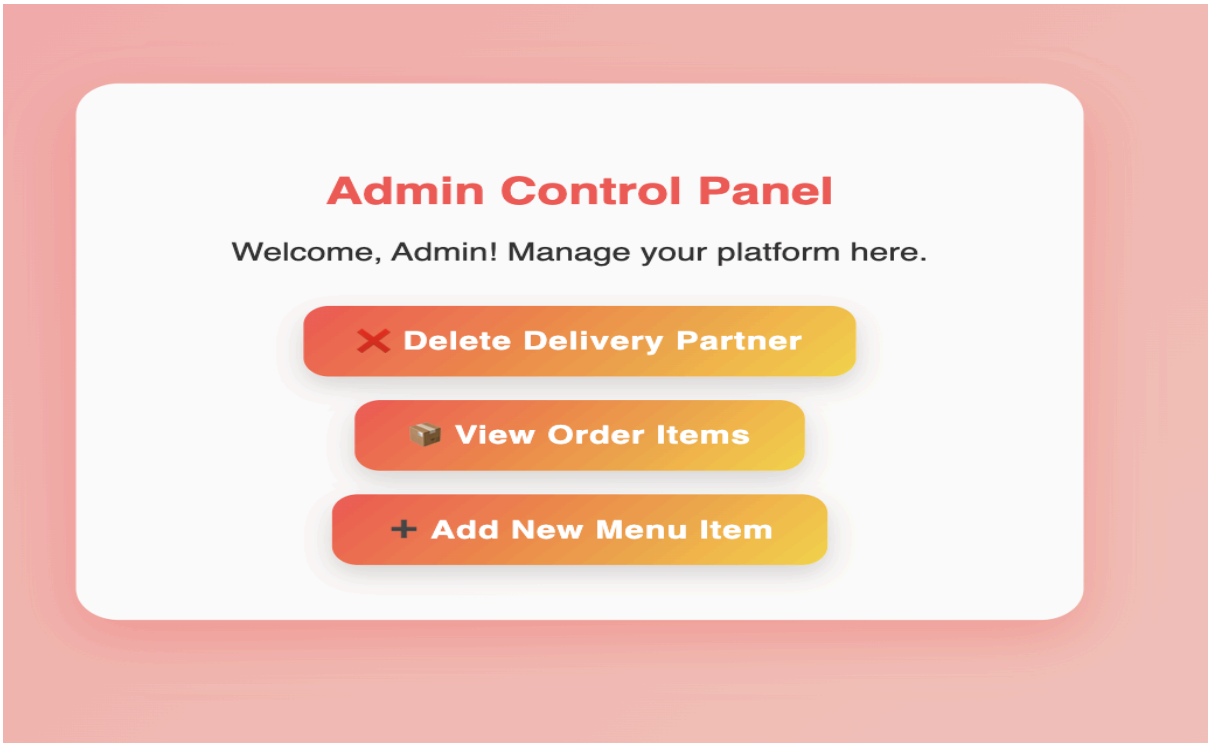


Admin control UI screenshots:

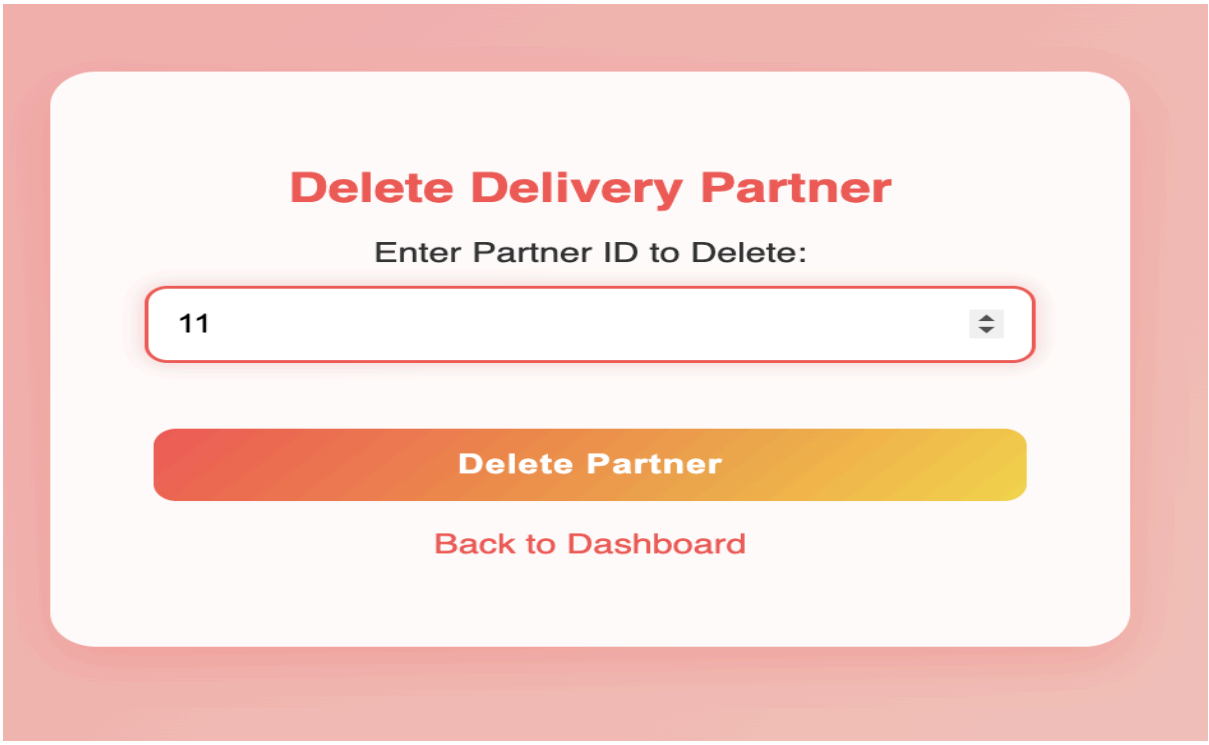
Login: (only with restricted credentials)



Admin Dashboard:

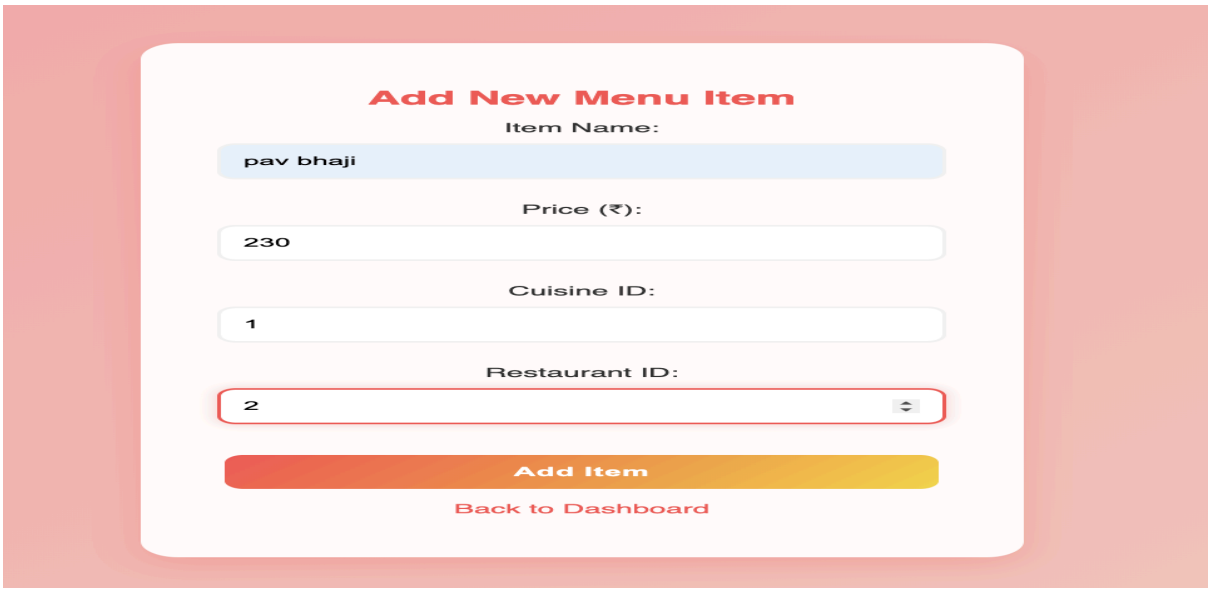


Delete delivery partner page:



View order items page:

Add new menuitems page:



Triggers, Procedures/Functions and Joins

```
DELIMITER $$

CREATE TRIGGER update_order_total
AFTER INSERT ON OrderItem
FOR EACH ROW
BEGIN
    UPDATE Orders
    SET total_amt = (
        SELECT SUM(subtotal)
        FROM OrderItem
        WHERE OrderItem.order_id = NEW.order_id
    )
    WHERE Orders.order_id = NEW.order_id;
END $$

DELIMITER ;
```

```
DELIMITER $$

CREATE TRIGGER update_order_status_after_payment
AFTER INSERT ON Payment
FOR EACH ROW
BEGIN
    UPDATE Orders
    SET status = NEW.pay_status
    WHERE order_id = NEW.order_id;
END $$

DELIMITER ;
```

```sql
DELIMITER $$

CREATE TRIGGER assign_delivery_after_payment
AFTER INSERT ON Payment
FOR EACH ROW
BEGIN
    DECLARE random_partner INT;

    -- Pick a random available delivery partner
    SELECT partner_id
    INTO random_partner
    FROM DeliveryPartner
    WHERE availability_status = 'Available'
    ORDER BY RAND()
    LIMIT 1;

    -- Assign partner & mark order as placed
    UPDATE Orders
    SET partner_id = random_partner,
        status = 'Placed'
    WHERE order_id = NEW.order_id;

    -- Mark that partner as busy (optional)
    UPDATE DeliveryPartner
    SET availability_status = 'Busy'
    WHERE partner_id = random_partner;
END $$

DELIMITER ;
```

```sql
DELIMITER $$

CREATE PROCEDURE PlaceNewOrder (
    IN p_order_id INT
)
BEGIN
    DECLARE random_partner INT;

    -- pick any available partner
    SELECT partner_id
    INTO random_partner
    FROM DeliveryPartner
    WHERE availability_status = 'Available'
    ORDER BY RAND()
    LIMIT 1;

    -- update the existing order
    UPDATE Orders
    SET partner_id = random_partner,
        status = 'Placed'
    WHERE order_id = p_order_id;

    -- mark that partner as busy
    UPDATE DeliveryPartner
    SET availability_status = 'Busy'
    WHERE partner_id = random_partner;

    -- return confirmation
    SELECT p_order_id AS order_id, random_partner AS assigned_partner;
END $$

DELIMITER ;
```

```
DELIMITER $$

CREATE FUNCTION AvgRestaurantRating(p_restaurant_id INT)
RETURNS DECIMAL(3,2)
DETERMINISTIC
BEGIN
    DECLARE avg_rating DECIMAL(3,2);

    SELECT AVG(rating)
    INTO avg_rating
    FROM Review
    WHERE restaurant_id = p_restaurant_id;

    RETURN IFNULL(avg_rating, 0.00);
END $$

DELIMITER ;
```

```
CREATE OR REPLACE VIEW OrderSummary AS
SELECT
    o.order_id,
    c.customer_name,
    r.rest_name,
    o.order_date,
    o.status,
    o.total_amt
FROM Orders o
JOIN Customer c ON o.customer_id = c.customer_id
JOIN Restaurant r ON o.restaurant_id = r.restaurant_id;

SELECT * FROM OrderSummary;
```

Code snippets for invoking the Procedures and Functions

```
CALL PlaceNewOrder(1, 2, 1, 'Placed');
SELECT * FROM Orders ORDER BY order_id DESC LIMIT 1;
```

```
SELECT AvgRestaurantRating(1) AS SpiceHub_Rating;
SELECT AvgRestaurantRating(2) AS LaPasta_Rating;
```

Github Link

https://github.com/ShambhaviPM/Smart-Bites-Online-Food-Delivery-and-Analytics-Platform

http://127.0.0.1:5000/register