Online Food Ordering and Delivery Management System

Course:DBMS-UE23CS351A

Team:

Mevin Jose PES1UG23AM170

Prateek Meher K PES1UG23AM169

Section:C

Github Repo Link:

https://github.com/MJenius/DBMS-Project

Abstract

This project implements a comprehensive Database Management System (DBMS) for a food delivery platform, utilizing MySQL for data persistence and Python Flask for web application development. The system manages core entities including customers, restaurants, menu items, orders, deliveries, and drivers, with enforced referential integrity through foreign key constraints.

The application features a modern web interface with a vibrant yet formal design, including a left-sidebar navigation for improved usability. Key functionalities encompass order management, delivery assignment, menu administration, and comprehensive reporting capabilities. The system supports complex database queries including joins, nested queries, and aggregate functions to provide business intelligence and operational insights.

The DBMS architecture ensures data consistency through proper normalization, constraint validation, and error handling mechanisms. This project demonstrates practical implementation of relational database design principles combined with full-stack web development, creating an efficient solution for managing multi-restaurant food delivery operations.

User Requirement Specifications

Purpose of the Project

The purpose of this project is to design and develop a comprehensive Database Management System (DBMS) for a multi-restaurant food delivery platform. The system aims to streamline operations by managing complex relationships between customers, restaurants, drivers, menu items, and orders in a centralized database. By implementing a robust relational database with proper data integrity constraints and a user-friendly web interface, the system enables efficient order processing, delivery management, and business analytics. The primary goal is to provide stakeholders—including administrators, restaurant managers, and delivery personnel—with a unified platform to manage daily operations while maintaining data consistency and accessibility.

Scope of the Project

The scope of this DBMS project encompasses the design and implementation of a complete food delivery ecosystem. The system will manage multiple restaurants, each with their own menu items, customers placing orders, delivery drivers, and delivery assignments. The project includes database design with proper normalization, implementation of referential integrity through foreign key constraints, and a web-based interface for user interaction. The system will support core operations such as customer registration, order placement, menu management, delivery assignment, and generate reports on customer activity and order trends. The project also incorporates advanced SQL queries including joins, nested queries, and aggregate functions for reporting purposes. However, the scope excludes payment gateway integration, real-time GPS tracking, and external API integrations.

Detailed Description

The Food Delivery System is a multi-tiered application that connects customers with restaurants and delivery services. The system maintains detailed information about all stakeholders and their interactions throughout the order lifecycle.

Core Entities and Relationships:

- Users & Authentication: The system maintains a Users table that stores user credentials and roles (Customer, Restaurant Manager, Delivery Driver, Admin). User Privileges table defines role-based access control, determining which users can perform specific operations within the system.

- Customers: The Customers table stores customer details including contact information, addresses, and phone numbers. Customers can place multiple orders and have a one-to-many relationship with the Orders table.

- Restaurants & Menu Management: The Restaurants table maintains information about partner restaurants. Each restaurant has a Menu table containing menu items with details such as item name, description, price, and availability status. The relationship between Restaurants and Menu is one-to-many.

- Orders & Order Items: When customers place orders, records are created in the Orders table containing order details, timestamps, total amount, and delivery address. The Order_Items junction table maintains many-to-many relationship between Orders and Menu items, storing quantity and item-specific pricing for each order.

- Drivers & Deliveries: The Drivers table stores information about delivery personnel including their contact details and current status. The Deliveries table records delivery assignments, linking orders to drivers and tracking delivery status (pending, in-transit, delivered).

- Data Integrity: Foreign key constraints are implemented to ensure referential integrity between all tables, preventing orphaned records and maintaining data consistency.

Functional Requirements:

- Create, read, update, and delete operations for all entities (CRUD)
- Place orders and automatically update menu item availability
- Assign deliveries to available drivers
- Generate reports on customer order frequency, restaurant performance, and delivery metrics
- Execute complex queries including multi-table joins and aggregate functions
- Role-based access control for different user types

Technical Implementation:

The backend uses MySQL for data storage with proper normalization up to 3NF. Python Flask framework provides the web application layer with a modern, vibrant UI featuring a left-sidebar navigation. The system implements error handling for constraint violations and provides meaningful feedback to users during database operations.

Softwares Used

Backend Development

- Python 3.x – Primary programming language for server-side logic
- Flask – Web framework for building the application and routing HTTP requests
- MySQL – Relational Database Management System for data storage and queries

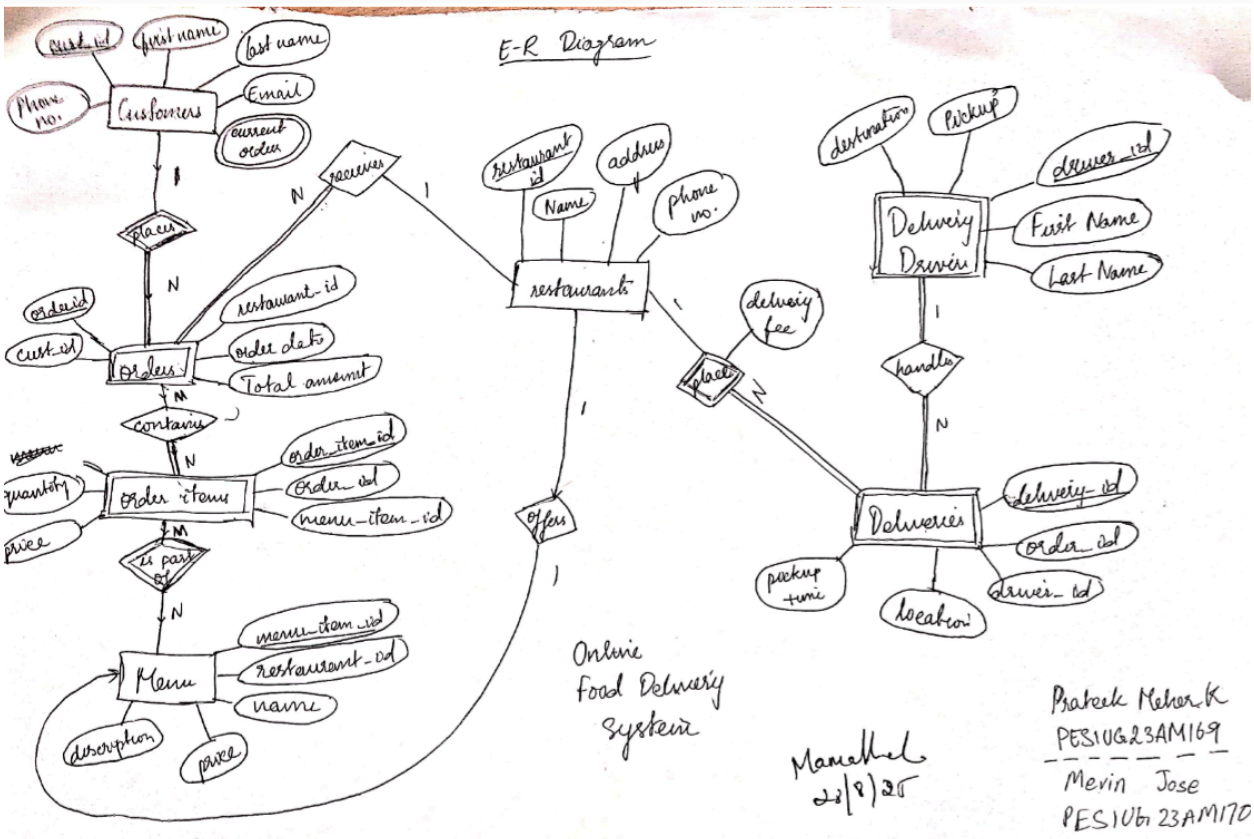
Frontend Development

- HTML5 – Markup language for web page structure
- CSS3 – Styling with gradients, animations, and responsive design
- JavaScript – Client-side interactivity and form validation

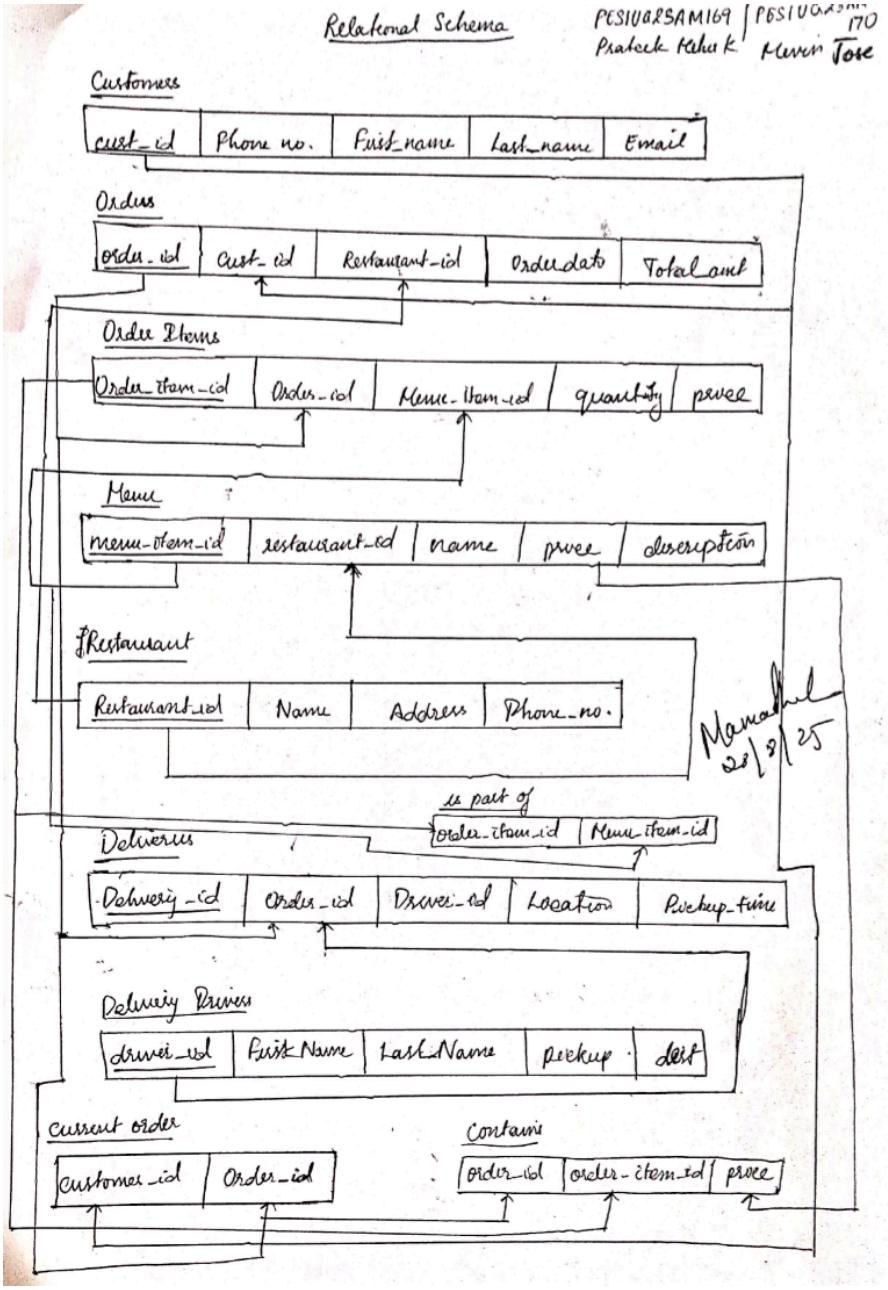
Libraries & Tools

- Flask-MySQL – Python connector for MySQL database integration
- Jinja2 – Template engine for dynamic HTML rendering
- Font Awesome – Icon library for UI elements

ER Diagram


E-R Diagram
Customers
cust_id
first name
last name
Email
Phone no.
current order
places
receives
Orders
order_id
cust_id
restaurant_id
order date
Total amount
contains
Order items
order_item_id
Order_id
menu_item_id
quantity
price
is part of
Menu
menu_item_id
restaurant_id
name
description
price
restaurants
restaurant id
Name
address
phone no.
offers
delivery fee
place
Delivery Driver
driver_id
First Name
Last Name
handle
Deliveries
delivery_id
order_id
driver_id
location
pickup time
destination
Pickup
Online Food Delivery System
Prateek Mehar K
PES1UG23AM169
Merin Jose
PES1UG23AM170

Relational Schema



DDL Commands

1. CREATE TABLE

Used to create database tables with their structure and constraints.

Examples from project:

CREATE TABLE customers – Stores customer information

CREATE TABLE restaurants – Stores restaurant details
CREATE TABLE menu – Stores menu items with foreign key to restaurants
CREATE TABLE orders – Stores order records with foreign keys to customers and restaurants
CREATE TABLE order_items – Junction table linking orders to menu items
CREATE TABLE deliveries – Stores delivery information
CREATE TABLE delivery_drivers – Stores driver details
CREATE TABLE customer_current_orders – Tracks active orders per customer

2. CREATE TRIGGER
Defines triggers to automatically execute actions on table events.
Examples in project:
after_order_insert – Automatically adds order to Customer_Current_Orders
after_order_item_insert – Automatically updates order total amount
after_delivery_insert – Automatically removes order from current orders

3. CREATE PROCEDURE
Defines stored procedures for complex database operations.
Examples in project:
PlaceOrder – Handles order placement logic
AssignDelivery – Assigns deliveries to drivers

4. CREATE FUNCTION
Defines reusable functions for calculations.
Examples in project:
GetActiveOrderCount() – Counts active orders for a customer
GetDriverEarnings() – Calculates total earnings for a driver
GetRestaurantRevenue() – Calculates total revenue for a restaurant

CRUD Operations Screenshots

1. CREATE Operations

Add Customer

First Name

Last Name

Phone

Email

CANCEL

ADD

Add Restaurant

Name

Address

Phone

CANCEL

ADD

Add Menu Item

Name

Restaurant

leons

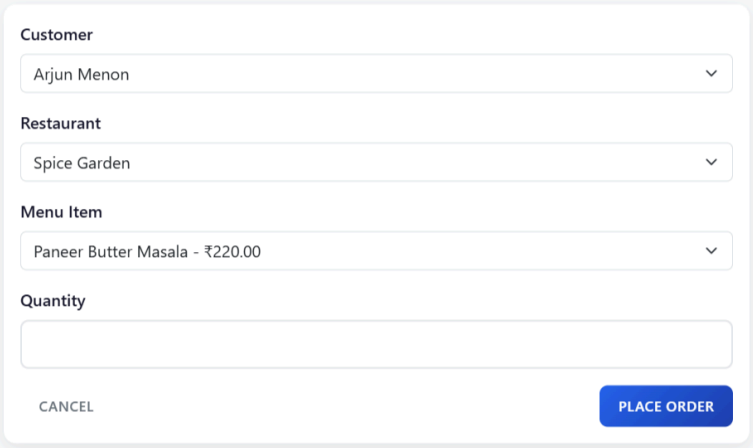
Description

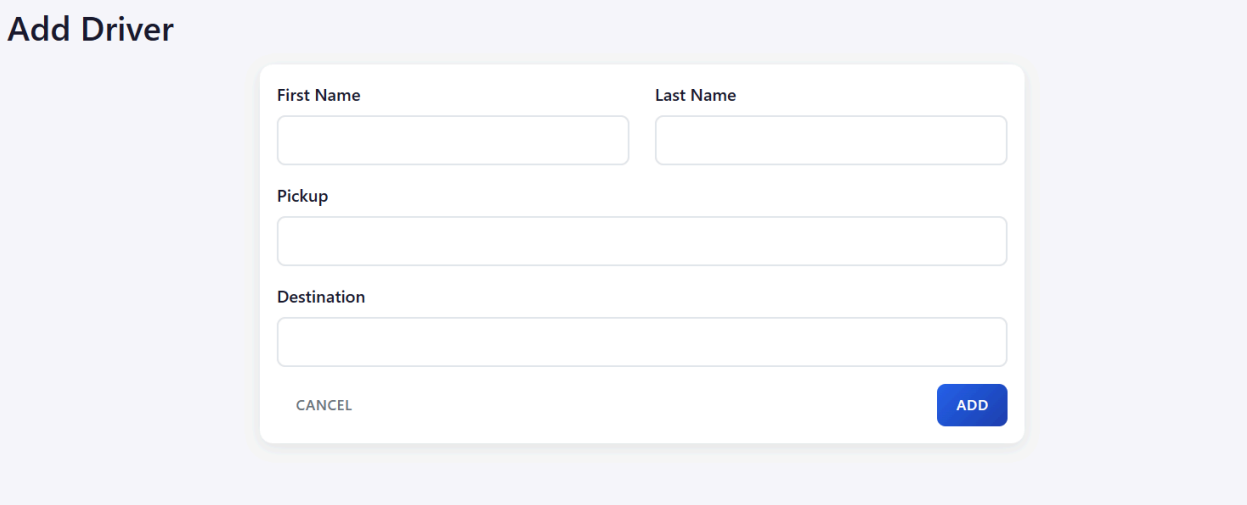
Price

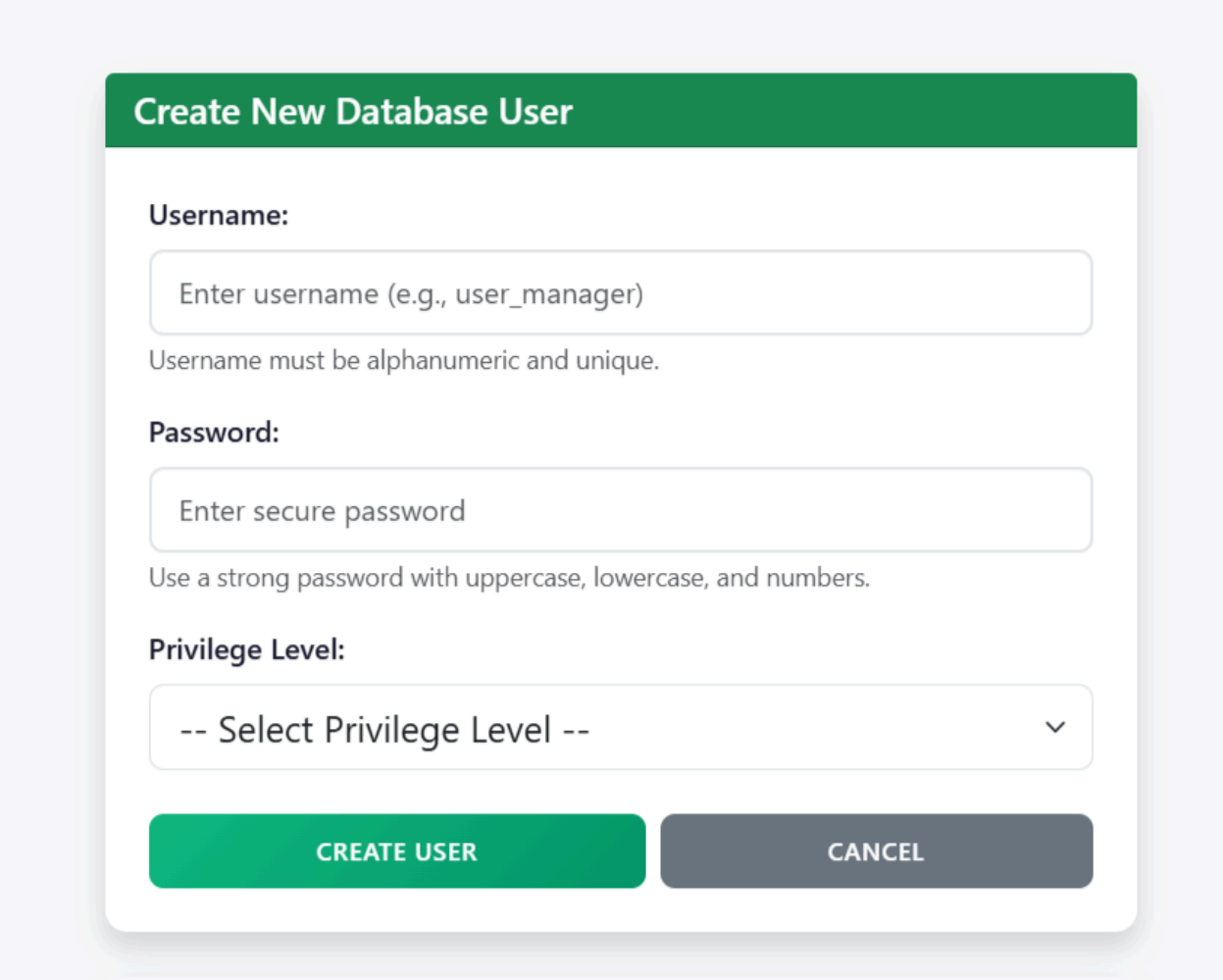
CANCEL

ADD

Place a New Order







2. READ Operations

Customers

+ ADD CUSTOMER

ID	NAME	PHONE	EMAIL	
8	Abhiram	1234567891	abhiram@gmail.com	EDIT DELETE
5	Rajeev	1234567890	rajeev@gmail.com	EDIT DELETE
4	Mevin Jose	9998887776	mevin.jose@email.com	EDIT DELETE
3	Rahul Patel	9123456789	rahul.patel@email.com	EDIT DELETE
2	Sneha Rao	9988776655	sneha.rao@email.com	EDIT DELETE
1	Arjun Menon	9876543210	arjun.menon@email.com	EDIT DELETE

Restaurants

+ ADD RESTAURANT

ID	NAME	ADDRESS	PHONE	
3	leons	banashankari	9876567891	EDIT DELETE
2	Tandoori Nights	Koramangala, Bengaluru	0807654321	EDIT DELETE
1	Spice Garden	MG Road, Bengaluru	0801234567	EDIT DELETE

Menu Items

+ ADD MENU ITEM

ID	NAME	RESTAURANT	PRICE	
5	chicken burger	3	₹150.00	EDIT DELETE
2	Chicken Biryani	2	₹250.00	EDIT DELETE
1	Paneer Butter Masala	1	₹220.00	EDIT DELETE

Orders

Search order id or custom | SEARCH

ORDER ID	CUSTOMER	RESTAURANT	ORDER DATE	
#18	Arjun Menon	Spice Garden	2025-10-10	VIEW
#17	Abhiram	Tandoori Nights	2025-10-10	VIEW
#16	Rajeev	Spice Garden	2025-10-10	VIEW
#15	Mevin Jose	Tandoori Nights	2025-10-10	VIEW
#14	Sneha Rao	Spice Garden	2025-10-10	VIEW
#13	Mevin Jose	leons	2025-10-10	VIEW

Delivery Drivers

+ ADD DRIVER

ID	NAME	PICKUP	DESTINATION	
4	Pratham	Tandoori Nights	HSR	EDIT DELETE
3	Vikram Das	Spice Garden	HSR Layout	EDIT DELETE
2	Aisha Khan	Tandoori Nights	Whitefield	EDIT DELETE
1	Karan Singh	Spice Garden	Indiranagar	EDIT DELETE

All Deliveries

CUSTOMER	RESTAURANT	PICKUP TIME	LOCATION
Arjun Menon	Spice Garden	2025-10-08 19:30:00	Indiranagar
Sneha Rao	Tandoori Nights	2025-10-07 20:15:00	Whitefield
Rahul Patel	Spice Garden	2025-10-06 18:45:00	HSR Layout
Arjun Menon	Spice Garden	2025-10-08 09:27:35	Indiranagar
Arjun Menon	Spice Garden	2025-10-28 14:53:46	Indiranagar
Arjun Menon	Spice Garden	2025-10-28 18:49:38	banashankari
Arjun Menon	Spice Garden	2025-10-28 19:07:26	banashankari
Sneha Rao	Spice Garden	2025-11-04 14:31:36	jp nagar
Arjun Menon	Spice Garden	2025-11-04 14:42:59	jp nagar
Mevin Jose	leons	2025-11-04 14:54:50	home

Order Details

Order #18

Customer: Arjun Menon

Restaurant: Spice Garden

Placed on: 2025-10-10

Items

chicken burger
Qty: 2 ₹300.00

Deliveries

No deliveries recorded for this order.

3. UPDATE Operations

Edit Restaurant

Name

Spice Garden

Address

MG Road, Bengaluru

Phone

0801234567

CANCEL UPDATE

Edit Menu Item

Name

Chicken Biryani

Restaurant

Tandoori Nights

Description

Hyderabadi style biryani

Price

250.00

CANCEL

UPDATE

Edit Driver

First Name

Pratham

Last Name

Pickup

Tandoori Nights

Destination

HSR

CANCEL

UPDATE

GRANT STATEMENT

```
GRANT USAGE ON *.* TO `Rajeev`@`localhost`
```

```
GRANT SELECT ON `dbms_project`.* TO `Rajeev`@`localhost`
```

Database Access

This user has permissions on database: **dbms_project**

Scope

Connected from host: **localhost**

Edit User Privileges

Select Privilege Level:

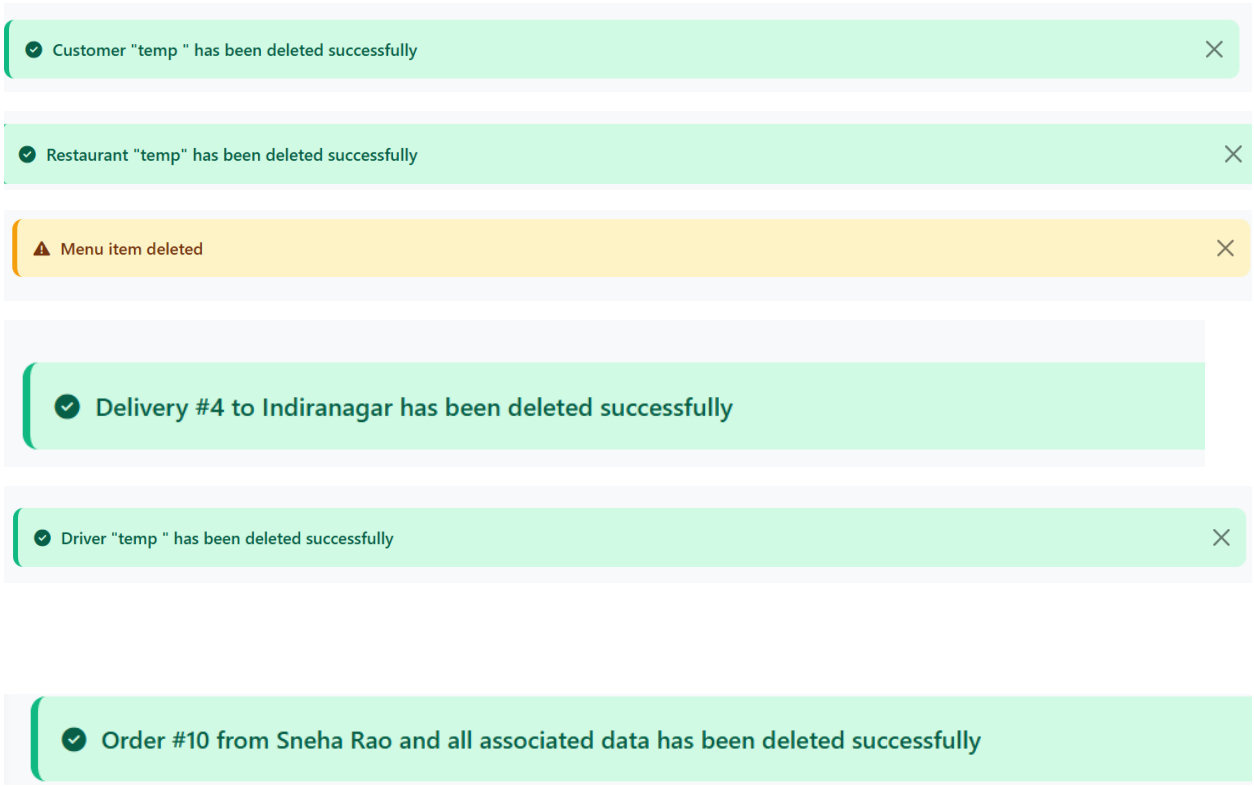
-- Choose Privilege Level --

Note: Changing the privilege level will revoke all current privileges and grant only the new ones.

UPDATE PRIVILEGES

CANCEL

4. DELETE Operations



Features Of The Application

1. Dashboard
    - Central hub displaying system overview
    - Quick access to all major functions
    - Real-time status indicators
2. Customer Management
    - Add, view, edit, and delete customers
    - Store customer contact information and delivery addresses
    - Track customer order history
    - Cascade delete customers with all related orders
3. Restaurant Management
    - Register and manage multiple restaurants
    - View restaurant details and performance metrics
    - Edit restaurant information
    - Delete restaurants (with constraint validation)
4. Menu Management
    - Add menu items to restaurants with prices and descriptions
    - Update menu item details and availability
    - Delete menu items with integrity protection
    - Organize items by restaurant
5. Order Management
    - Place orders from specific restaurants
    - View all orders with detailed information
    - Track order status (pending, processing, completed)
    - View complete order details including items and quantities
    - Delete orders with cascading deletion of related deliveries and items
    - Automatic order total calculation
6. Delivery Management
    - Assign deliveries to available drivers
    - View all active deliveries with pickup times and locations
    - Track delivery status and assignments

- Delete delivery records with proper error handling
- Display delivery location and customer information

7. Driver Management
    - Register delivery drivers
    - Track driver availability and status
    - View driver information and contact details
    - Delete drivers with constraint validation
    - Monitor driver assignments
8. User & Security Management
    - Create and manage system users
    - Role-based access control (Admin, Manager, Operator, Viewer)
    - Edit user privileges with granular permission levels
    - Protect system users (MySQL, root, admin)
    - Prevent unauthorized deletions of critical database users
9. Advanced Queries
    - JOIN Queries – Multi-table joins showing related data (Customers-Orders-Restaurants)
    - Nested Queries – Complex queries with subqueries for advanced filtering
    - Aggregate Queries – Statistical analysis using COUNT, SUM, AVG functions
10. Business Reports
    - Customer Order Summary – Track active orders per customer
    - Restaurant Performance Metrics – Revenue and order analytics
    - Driver Statistics – Earnings and delivery counts
    - Order Analytics – Trends and patterns
11. User Interface
    - Left Sidebar Navigation – Always-accessible menu with hierarchical organization
    - Responsive Layout – Works on desktop and mobile devices
    - Intuitive Forms – Clean, organized data entry interfaces
    - Data Tables – Sortable, searchable tables with action buttons
    - Visual Feedback – Color-coded alerts (success, error, warning)
12. Data Integrity & Error Handling
    - Foreign key constraint validation

- Cascade delete protection with helpful error messages
- Transaction rollback on failures
- Graceful error handling with user-friendly messages
- Prevention of orphaned records

13. Database Features
    - Relational database design normalized to 3NF
    - Automated triggers for business logic (order updates, delivery assignments)
    - Stored procedures for complex operations
    - Referential integrity constraints
    - Proper indexing for performance

Triggers, Procedures/Functions, Nested query, Join, Aggregate queries

Triggers

- after_order_insert – Automatically adds new orders to Customer_Current_Orders table
- after_order_item_insert – Automatically updates order total amount when items are added
- after_delivery_insert – Automatically removes completed orders from Customer_Current_Orders
- before_order_delete – Handles cleanup before order deletion

Stored Procedures

- PlaceOrder – Handles complete order placement logic with validation
- AssignDelivery – Assigns deliveries to available drivers with status updates
- CancelOrder – Cancels orders and updates related records

Functions

- GetActiveOrderCount(Customer_ID) – Returns count of active orders for a customer
- GetDriverEarnings(Driver_ID) – Calculates total earnings for a driver
- GetRestaurantRevenue(Restaurant_ID) – Calculates total revenue for a restaurant
- GetOrderTotal(Order_ID) – Computes order total from order items

JOIN Queries

- Customer-Orders-Restaurants JOIN – Displays customers, their orders, and associated restaurants
- Orders-Menu-Items JOIN – Shows order details with menu item names and prices
- Deliveries-Orders-Drivers JOIN – Links delivery information with orders and driver details
- Restaurants-Menu-Items JOIN – Associates restaurants with their complete menu

Nested Queries

- Find customers with orders from specific restaurants
- Retrieve orders with total amounts exceeding average
- List drivers assigned to deliveries from top restaurants
- Find menu items ordered by specific customers
- Identify restaurants with above-average order counts

Aggregate Queries

- COUNT(Order_ID) – Total orders per customer (Customer Order Summary Report)
- SUM(Order_Total) – Total revenue per restaurant
- AVG(Order_Total) – Average order value
- COUNT(Delivery_ID) – Delivery count per driver
- SUM(Driver_Earnings) – Total earnings by driver
- MAX/MIN(Order_Total) – Highest and lowest order values

Code snippets for Procedures Functions and Triggers

TRIGGERS

after_order_insert

```
CREATE TRIGGER after_order_insert
AFTER INSERT ON orders
FOR EACH ROW
BEGIN
   INSERT INTO customer_current_orders (Customer_ID, Order_ID)
   VALUES (NEW.Customer_ID, NEW.Order_ID);
END;
```

after_order_item_insert

```
CREATE TRIGGER after_order_item_insert
AFTER INSERT ON order_items
FOR EACH ROW
BEGIN
   UPDATE orders
   SET Order_Total = Order_Total + (NEW.Quantity * NEW.Item_Price)
   WHERE Order_ID = NEW.Order_ID;
END;
```

after_delivery_insert

```
CREATE TRIGGER after_delivery_insert
AFTER INSERT ON deliveries
FOR EACH ROW
BEGIN
   DELETE FROM customer_current_orders
   WHERE Order_ID = NEW.Order_ID;
END;
```

STORED PROCEDURES

PlaceOrder

```
DELIMITER //
CREATE PROCEDURE PlaceOrder(
    IN p_customer_id INT,
    IN p_restaurant_id INT,
    IN p_delivery_address VARCHAR(255),
    OUT p_order_id INT
)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;
    START TRANSACTION;
    INSERT INTO orders (Customer_ID, Restaurant_ID, Order_Date, Delivery_Address, Order_Total)
    VALUES (p_customer_id, p_restaurant_id, NOW(), p_delivery_address, 0);
    SET p_order_id = LAST_INSERT_ID();
    COMMIT;
END //
DELIMITER ;
```

AssignDelivery

```
DELIMITER //
CREATE PROCEDURE AssignDelivery(
    IN p_order_id INT,
    IN p_driver_id INT,
    IN p_pickup_time DATETIME,
    OUT p_delivery_id INT
)
BEGIN
```

```
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;
    START TRANSACTION;
    INSERT INTO deliveries (Order_ID, Driver_ID, Pickup_Time, Delivery_Status)
    VALUES (p_order_id, p_driver_id, p_pickup_time, 'Pending');
    SET p_delivery_id = LAST_INSERT_ID();
    UPDATE delivery_drivers
    SET Driver_Status = 'Busy'
    WHERE Driver_ID = p_driver_id;
    COMMIT;
END //
DELIMITER ;
```

FUNCTIONS

GetActiveOrderCount

```
DELIMITER //
CREATE FUNCTION GetActiveOrderCount(p_customer_id INT)
RETURNS INT
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE order_count INT;

    SELECT COUNT(*)
    INTO order_count
    FROM customer_current_orders
    WHERE Customer_ID = p_customer_id;

    RETURN order_count;
END //
```

```
DELIMITER ;
```

GetDriverEarnings

```
DELIMITER //
CREATE FUNCTION GetDriverEarnings(p_driver_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE total_earnings DECIMAL(10,2);

    SELECT SUM(o.Order_Total)
    INTO total_earnings
    FROM deliveries d
    JOIN orders o ON d.Order_ID = o.Order_ID
    WHERE d.Driver_ID = p_driver_id;

    RETURN COALESCE(total_earnings, 0);
END //
DELIMITER ;
```

GetRestaurantRevenue

```
DELIMITER //
CREATE FUNCTION GetRestaurantRevenue(p_restaurant_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE total_revenue DECIMAL(10,2);

    SELECT SUM(Order_Total)
    INTO total_revenue
```

```
    FROM orders
    WHERE Restaurant_ID = p_restaurant_id;

    RETURN COALESCE(total_revenue, 0);
END //
DELIMITER ;
```

GetOrderTotal

```
DELIMITER //
CREATE FUNCTION GetOrderTotal(p_order_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE order_total DECIMAL(10,2);

    SELECT SUM(Quantity * Item_Price)
    INTO order_total
    FROM order_items
    WHERE Order_ID = p_order_id;

    RETURN COALESCE(order_total, 0);
END //
DELIMITER ;
```