Online Food Ordering System - Database Report

1. Introduction

The Online Food Ordering System allows users to order food from various restaurants via a web or mobile platform. This report presents the design and implementation of the database system used to manage users, restaurants, menus, orders, and payments efficiently and securely.

2. Objectives

- To design a robust and scalable database for managing food orders.
- To support multiple users and restaurants.
- To ensure fast and reliable order processing and tracking.
- To maintain data integrity and security.

3. Database Requirements

The database must:

- Handle multiple restaurants and menu categories.
- Allow customers to browse menus and place orders.
- Track order status (e.g., pending, preparing, delivered).
- Manage user accounts and authentication.
- Support payment processing.

4. Entity-Relationship Diagram (ERD)

Main Entities:

- User: Customer or Admin
- Restaurant
- MenuItem
- Order

- OrderDetails

- Category

- Payment

(Insert ERD image here or describe relationships)

5. Entity Descriptions and Schema

Sample schema for key tables is provided below:

User Table: user_id, name, email, password, role

Restaurant Table: restaurant_id, name, address, phone

Category Table: category_id, name

MenuItem Table: item_id, name, price, restaurant_id, category_id, description

Order Table: order_id, user_id, order_date, status, total_amount

OrderDetails Table: detail_id, order_id, item_id, quantity

Payment Table: payment_id, order_id, amount, payment_method, payment_date

6. Normalization

The database is normalized to 3rd Normal Form (3NF):

- No redundant data.

- All non-key attributes depend only on the primary key.

- Eliminated transitive dependencies.

7. Sample SQL Queries

a. Get all menu items from a restaurant:

SELECT name, price FROM MenuItem WHERE restaurant_id = 1;

b. Get all orders of a user:

```
SELECT * FROM `Order` WHERE user_id = 5;
```

c. View order details:

```
SELECT o.order_id, m.name, od.quantity, m.price
FROM OrderDetails od
JOIN MenuItem m ON od.item_id = m.item_id
JOIN `Order` o ON od.order_id = o.order_id
WHERE o.user_id = 5;
```

8. Security Considerations

- Passwords are hashed before storing.
- SQL Injection is prevented using parameterized queries.
- Role-based access is implemented to restrict admin features.

9. Conclusion

The designed database effectively supports the core functionalities of an online food ordering system. It ensures scalability, performance, and data integrity, laying a solid foundation for a web or mobile-based application.