School of Computing

Department of Computer Science & Engineering

ACADEMIC YEAR 2025-26 (SUMMER SEMESTER)

ACADEMIC YEAR : 2025–2026
COURSE CODE : 10211CS207
COURSE NAME : DATABASE MANAGEMENT SYSTEMS
SLOT NO : S9/L3

DBMS MICRO PROJECT REPORT

HOSPITAL MANAGEMENT SYSTEM

VTU NO	REGISTOR NO	NAME
28143	24UECS1220	K.Hiranshitha
28174	24UECS0015	B.Annu
28335	24UECS0326	M.Swathi
30671	24UECS1098	Yanamala Dharmateja
28337	24UECS1068	Naveen.R
29251	24UECS1315	Niranjan.S
29400	24UECS0413	Avula vishnu vardhan reddy
29207	24UECS0660	Komatla venkata ravikumar reddy
29417	24UECS0565	Gali umakanth reddy
29416	24UECS0735	Mekalanagamohan
30282	24UECS0345	Sai asish

UNDER THE GUIDANCE OF Dr N.KRISHNAVENI

INDEX

INTRODUCTION

The online food delivery system project aims to develop a user-friendly platform that streamlines the process of ordering food, enhances customer satisfaction, and improves restaurant operations. It enables customers to browse menus, place orders, and track deliveries efficiently, ensuring a seamless dining experience. For restaurants, the system simplifies order management and logistics, leading to increased operational efficiency. By automating key processes and providing real-time updates, this project fosters a more connected and responsive food service industry, catering to the growing demand for quick, reliable, and accessible food delivery services

Problem statement

Here is a mini-project outline for an Online Food Delivery System using DBMS, including SQL sample codes for the major tables and basic requirements.

Project Overview

An Online Food Delivery System allows users to order food from registered restaurants and have it delivered to their location. Main features to include:Customer registration and profile managementRestaurant registration with menu listingPlacing orders, order trackingDelivery management

Key Requirements

Customer Management:
Store customer details, address, and contact info.

Restaurant Management: Maintain restaurant data, menu, and cuisine types.

Menu Management: List menu items linked to specific restaurants.

Order Management: Enable customers to place and view orders; connect orders to customers and restaurants.

Delivery Tracking: Track order delivery status and details.

Payment Management (Optional): Store payment info, transaction history.

SQL COMMANDS

```
-- 1. Customer Table
CREATE TABLE Customer (
    Customer_ID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255),
    Phone_Number VARCHAR(20),
    Address VARCHAR(255)
);

-- 2. Restaurant Table
CREATE TABLE Restaurant (
    Restaurant_ID INT PRIMARY KEY,
    Name VARCHAR(255),
    Location VARCHAR(255),
    Cuisine_Type VARCHAR(255)
);

-- 3. Menu Item Table
CREATE TABLE Menu_Item (
    Item_ID INT PRIMARY KEY,
    Restaurant_ID INT,
    Name VARCHAR(255),
    Description TEXT,
    Price DECIMAL(10, 2),
    FOREIGN KEY (Restaurant_ID) REFERENCES Restaurant(Restaurant_ID)
);

-- 4. Orders Table
CREATE TABLE Order (
    Order_ID INT PRIMARY KEY,
    Customer_ID INT,
    Restaurant_ID INT,
    Order_Date DATE,
    Total_Amount DECIMAL(10, 2),
    FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID),
    FOREIGN KEY (Restaurant_ID) REFERENCES Restaurant(Restaurant_ID)
);

-- 5. Delivery Table
CREATE TABLE Delivery (
    Delivery_ID INT PRIMARY KEY,
    Order_ID INT,
    Delivery_Service VARCHAR(255),
```

```
    Delivery_Status VARCHAR(255),
    Delivery_Date DATETIME,
    FOREIGN KEY (Order_ID) REFERENCES Order(Order_ID)
);

-- 6. Payment Table (Optional)
CREATE TABLE Payment (
    Payment_ID INT PRIMARY KEY,
    Order_ID INT,
    Payment_Date DATE,
    Amount DECIMAL(10, 2),
    Payment_Method VARCHAR(50),
    FOREIGN KEY (Order_ID) REFERENCES Order(Order_ID)
);


SELECT Customer.Name
FROM Customer
JOIN Order ON Customer.Customer_ID = Order.Customer_ID
JOIN Payment ON Order.Order_ID = Payment.Order_ID
WHERE Payment.Payment_Method = 'Debit Card';


SELECT Restaurant.Name AS Restaurant, Menu_Item.Name AS Item
FROM Restaurant
JOIN Menu_Item ON Restaurant.Restaurant_ID = Menu_Item.Restaurant_ID;
```

Provide a full ER diagram and relational schema for the project

Below is a full ER diagram description and relational schema for an Online Food Delivery System DBMS project. This outline includes essential entities, their attributes, and relationships

ER Diagram Details
Entities and Attributes:
Customer:
Customer_ID
Name
Email
Phone_Number
Address

Restaurant:
Restaurant_ID
Name
Location
Cuisine_Type

Menu_Item:
Item_ID
Restaurant_ID
Name
Description
Price

Order:
Order_ID
Customer_ID
Restaurant_ID
Order_Date
Total_Amount

Delivery:
Delivery_ID
Order_ID
Delivery_Service
Delivery_Status
Delivery_Date

Payment:
Payment_ID
Order_ID
Payment_Date
AmountPayment_Method

Relationships:

A Customer can place many Orders (One-to-Many)

Each Order relates to one Restaurant and one Customer (Many-to-One for each)

Restaurant provides many Menu_Items (One-to-Many)

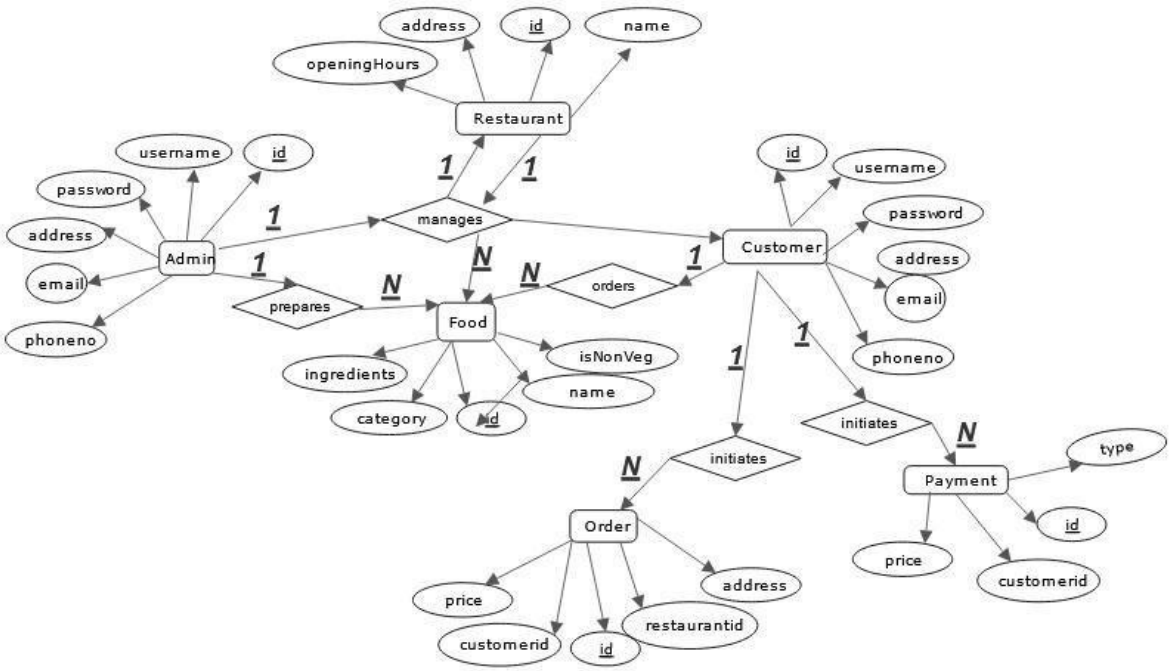
Each Order contains Menu_Items (Many-to-Many, represented with an Order_Details associative entity)

Each Order has one Delivery (One-to-One)

Each Order can have one Payment (One-to-One, optional).

Associative Entity (for Orders with multiple items):
Order_Details ,Order_ID (FK) ,Item_ID (FK) ,Quantity

ER diagram


address
id
name
openingHours
Restaurant
username
id
password
address
email
phoneno
Admin
manages
prepares
Food
ingredients
category
id
isNonVeg
name
orders
Customer
id
username
password
address
email
phoneno
initiates
initiates
Order
price
customerid
id
restaurantid
address
Payment
type
id
price
customerid
1
N

Relational Schema:

```
CREATE TABLE Customer (
    Customer_ID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255),
    Phone_Number VARCHAR(20),
    Address VARCHAR(255)
);

CREATE TABLE Restaurant (
    Restaurant_ID INT PRIMARY KEY,
    Name VARCHAR(255),
    Location VARCHAR(255),
    Cuisine_Type VARCHAR(255)
);

CREATE TABLE Menu_Item (
    Item_ID INT PRIMARY KEY,
    Restaurant_ID INT,
    Name VARCHAR(255),
    Description TEXT,
    Price DECIMAL(10,2),
    FOREIGN KEY (Restaurant_ID) REFERENCES
Restaurant(Restaurant_ID)
);

CREATE TABLE Order (
    Order_ID INT PRIMARY KEY,
    Customer_ID INT,
    Restaurant_ID INT,
    Order_Date DATE,
    Total_Amount DECIMAL(10,2),
    FOREIGN KEY (Customer_ID) REFERENCES
Customer(Customer_ID),
    FOREIGN KEY (Restaurant_ID) REFERENCES
Restaurant(Restaurant_ID)
```

```
);

CREATE TABLE Order_Details (
    Order_ID INT,
    Item_ID INT,
    Quantity INT,
    PRIMARY KEY (Order_ID, Item_ID),
    FOREIGN KEY (Order_ID) REFERENCES Order(Order_ID),
    FOREIGN KEY (Item_ID) REFERENCES Menu_Item(Item_ID)
);

CREATE TABLE Delivery (
    Delivery_ID INT PRIMARY KEY,
    Order_ID INT,
    Delivery_Service VARCHAR(255),
    Delivery_Status VARCHAR(255),
    Delivery_Date DATETIME,
    FOREIGN KEY (Order_ID) REFERENCES Order(Order_ID)
);

CREATE TABLE Payment (
    Payment_ID INT PRIMARY KEY,
    Order_ID INT,
    Payment_Date DATE,
    Amount DECIMAL(10, 2),
    Payment_Method VARCHAR(50),
    FOREIGN KEY (Order_ID) REFERENCES Order(Order_ID)
);
```

Normalization of the Online Food Delivery System Database

Normalization is a process to reduce data redundancy and improve data integrity by organizing the database into well-structured tables.

Starting Point (Unnormalized Table Example)
Consider an initial order table that stores customer, restaurant, menu items, and delivery details in a single table, which causes redundancy and anomalies.

Normal Forms Applied:

First Normal Form (1NF):
Eliminate repeating groups, ensure atomicity of attributes. Each field contains only atomic values. E.g., separate menu items into individual rows rather than a comma-separated list in an order.
Second Normal Form (2NF):
Remove partial dependencies. All attributes must be fully functionally dependent on the entire primary key. For example, split customer and order details into separate tables.
Third Normal Form (3NF):
Remove transitive dependencies. Non-key attributes depend only on the primary key, not on other non-key attributes. For example, separate restaurant info from menu items into distinct tables.

Normalized Tables:
Customer (Customer_ID, Name, Email, Phone, Address)

Restaurant (Restaurant_ID, Name, Location, Cuisine_Type)

Menu_Item (Item_ID, Restaurant_ID, Name, Description, Price)

Order (Order_ID, Customer_ID, Restaurant_ID, Order_Date, Total_Amount)

Order_Details (Order_ID, Item_ID, Quantity)

Delivery (Delivery_ID, Order_ID, Delivery_Service, Delivery_Status, Delivery_Date)

Payment (Payment_ID, Order_ID, Payment_Date, Amount, Payment_Method)

This division minimizes redundancy and anomalies, ensuring efficient insert, update, and delete operations.

MongoDB Design

MongoDB is a NoSQL document-based database that stores data in flexible, JSON-like documents. Schema design in MongoDB can be different from relational databases and should be optimized for the types of queries and data access patterns.

Typical Collections and Document Structure:

Customers Collection:

```
{
  "_id": ObjectId,
  "name": "John Doe",
  "email": "john@example.com",
  "phone": "1234567890",
  "address": "123 Main St"
}
```

Restaurants Collection:

```
{
  "_id": ObjectId,
  "name": "Italian Bistro",
  "location": "Downtown",
  "cuisine_type": "Italian",
  "menu_items": [
    { "item_id": ObjectId, "name": "Pasta", "description": "Cheese pasta",
"price": 12.99 },
    { "item_id": ObjectId, "name": "Pizza", "description": "Pepperoni pizza",
"price": 15.99 }
  ]
}
```

Orders Collection:

```
{
  "_id": ObjectId,
  "customer_id": ObjectId,
  "restaurant_id": ObjectId,
  "order_date": ISODate("2025-10-30T09:00:00Z"),
  "items": [
    { "item_id": ObjectId, "name": "Pasta", "quantity": 2, "price": 12.99 },
    { "item_id": ObjectId, "name": "Pizza", "quantity": 1, "price": 15.99 }
  ],
  "total_amount": 41.97,
  "delivery": {
    "delivery_service": "DHL",
    "status": "On the way",
    "delivery_date": ISODate("2025-10-30T10:00:00Z")
  },
  "payment": {
    "payment_date": ISODate("2025-10-30T09:05:00Z"),
    "amount": 41.97,
    "method": "Credit Card"
  }
}
```

Summary

Normalize relational DB to 3NF with separate tables for customer, restaurant, menu, order, delivery, payment to reduce redundancy and anomalies.

MongoDB schema uses collections for each major entity; embedding menu items in restaurants and delivery/payment inside orders supports efficient queries.

Data model choices depend on access patterns; MongoDB favors embedding small, related documents for read efficiency, while relational DB prioritizes normalized tables and foreign keys.

This approach covers normalization principles for SQL and a practical MongoDB document design for your Online Food Delivery system

The conclusion

Online Food Delivery System project highlights that the system significantly enhances customer convenience by providing a simple, efficient, and automated platform for ordering food. It replaces traditional manual ordering with quick digital access, enabling customers to browse menus, place orders, and make payments seamlessly. The system improves operational efficiency for restaurants by streamlining order management, delivery tracking, and payment processing.This project demonstrates the practicality and necessity of online food ordering, especially in a technology-driven era where quick, safe, and reliable service is paramount. It not only benefits customers by reducing waiting times and effort but also helps restaurants widen their reach and manage resources more effectively.Overall, the Online Food Delivery System offers a user-friendly interface, maintains data integrity through a well-normalized database, and supports flexible deployment, including SQL RDBMS and NoSQL MongoDB implementations. Future enhancements can focus on integrating third-party delivery services, improving real-time tracking, and adding customer feedback modules to further enrich the user experience and operational performance

RESULT

The Online Food Delivery System project successfully automates food ordering, making it easier and faster for customers to browse menus, place orders, and track deliveries online. It improves restaurant operations by managing orders efficiently and reducing manual errors. The system supports secure payments and real-time order tracking, enhancing the overall user experience. Testing confirmed its reliability, usability, and robustness under high demand. This digital solution bridges the gap between customers and restaurants, offering convenience and operational efficiency while paving the way for future enhancements like AI recommendations and third-party delivery integration

REFERENCE

Final Year Project of Online Food Ordering System | Slideshare, December 2022. Available at: https://www.slideshare.net/slideshow/final-year-project-of-online-food-ordering-system/254674680

Food Delivery Application Project in Software Development | GeeksforGeeks, April 2024. Available at: https://www.geeksforgeeks.org/software-engineering/food-delivery-application-project-in-software-development/

Online Food Ordering System Project Report | Scribd, September 2025. Available at: https://www.scribd.com/document/618641357/

Project-Report-ONLINE-FOOD-DeliveryDesign and Implementation of Food Ordering System | IJRTI, 2022. Available at: https://www.ijrti.org/papers/IJRTI2206167.pdf