# Project: Online Restaurant Ordering System

**Course:** CS 301 Fundamentals of Database Systems

**Semester:** Fall 2025

## 1. Goal

The purpose of this project is to give students hands-on experience in the **conceptual design**, **logical design**, **implementation**, **operation**, and **maintenance** of a relational database system for an online restaurant ordering platform.

This project emphasizes individual understanding of full-stack database development rather than specialization in isolated tasks. Extensions and creative improvements are strongly encouraged and may evolve into senior design or research projects.

## 2. Enterprise Description

The enterprise to be modeled is an **online restaurant ordering system** that allows customers to browse menus, place orders (dine-in, takeaway, or delivery), and manage payments. The system also supports restaurant staff and managers who handle menus, orders, and basic reporting.

Each student should choose a **specific restaurant scenario** (e.g., single restaurant with branches, local food chain, pizzeria, multi-cuisine restaurant, etc.) to model in detail. The system should capture the following aspects:

- **Restaurants / Branches:** Each restaurant or branch has a name, cuisine type, location (address), opening hours, and contact information.
- **Menu Categories:** Menus are divided into categories such as appetizers, main courses, desserts, drinks, etc.
- **Menu Items:** Each item has a name, description, price, category, and availability status. Items may support options (e.g., size, spice level, extra toppings).
- **Customers:** Customers may register an account with contact details and one or more delivery addresses, or optionally order as guests.
- **Orders:** An order has a customer, restaurant/branch, order type (dine-in, takeaway, delivery), order time, status (pending, in progress, completed, cancelled), and total amount.
- **Order Items:** Each order consists of one or more menu items with quantities and selected options.

- **Delivery (optional):** For delivery orders, store delivery address, delivery fee, and optionally assigned delivery driver.
- **Payments:** Each payment record includes the order, payment method (cash, card, e-wallet, etc.), amount, timestamp, and result (success/failure).
- **Promotions / Discounts (optional):** Coupon codes, discounts, or special offers linked to orders or customers.

## 3. Data Generation

Realistic sample data is required to support meaningful queries and testing. You may generate random data programmatically (e.g., using Python scripts or SQL procedures).

At a minimum, aim for:

- 3–5 restaurants or branches,
- At least 15–20 menu categories and 80–100 menu items,
- 100–200 customers,
- Several hundred orders, with a mix of dine-in, takeaway, and delivery orders,
- Order item and payment records sufficient to support non-trivial queries.

Ensure that data is consistent (e.g., each order belongs to an existing customer and branch, totals match the sum of order items, capacities and opening hours are respected where modeled).

## 4. Client Requests

### 1. E-R Model

Construct an **E-R diagram** that represents your conceptual design. Clearly indicate:

- Main entities (e.g., Restaurant, Branch, Customer, MenuCategory, MenuItem, Order, OrderItem, Payment, DeliveryInfo),
- Relationships (e.g., customer *places* order, order *contains* items, restaurant *offers* menu items),
- **Primary keys**, **attributes**, **cardinalities**, and participation constraints,
- Any specialization/generalization (e.g., Order $\rightarrow$ DineInOrder / DeliveryOrder, Payment $\rightarrow$ OnlinePayment / CashPayment).

## 2. Relational Model

Convert your E-R model to a **relational schema**. Apply normalization up to **3NF or BCNF**. Define:

- Tables and attributes corresponding to your entities and relationships,
- **Primary keys** and **foreign keys**,
- **Constraints** (NOT NULL, UNIQUE, CHECK) and appropriate **indexes** (e.g., on order time, customer, restaurant).

Implement the schema in a relational DBMS such as Oracle, MySQL, or PostgreSQL.

## 3. Populate Relations

Populate all tables with enough data to test typical operations and queries:

- Use INSERT scripts or data-generation programs,
- Ensure that each table has multiple rows and that relationships are meaningful (e.g., customers with multiple orders, orders with multiple items),
- Validate referential integrity and keep data consistent.

## 4. Queries

Implement at least **8–10 queries** that are useful for restaurant managers and system administrators, such as:

1. List the top 10 best-selling menu items across all restaurants.
2. Find the top 5 customers by total spending.
3. Compute total revenue per restaurant or branch for a given month.
4. Show the number of orders and total revenue per day for a selected restaurant.
5. Find menu categories that generate the highest revenue.
6. List all active orders (pending or in progress) for a given branch.
7. Find delivery orders that took longer than a certain threshold (e.g., more than 45 minutes from order time to delivery time, if modeled).
8. Identify customers with a high cancellation rate (e.g., more than 3 cancelled orders).
9. (Optional) Show usage statistics for promotions or discount codes.
10. (Optional) List items that are frequently ordered together (basic association-style query).

**5. Interfaces**

You may implement one or more of the following:

- **Customer Interface:** For browsing menus, placing orders, and viewing order history.
- **Restaurant Staff Interface:** For viewing and updating order status, and managing menu item availability.
- **Manager Interface:** For generating summary reports (sales by day, by category, by branch).

Interfaces may be:

- Command-line based (menu-driven or prompt-based), or
- Simple web-based (optional) if you are comfortable with basic web development.

This is a database-focused project; a clean and functional command-line interface is sufficient. You may add triggers or scheduled jobs (e.g., to auto-expire unconfirmed orders) as optional enhancements.

## 5. What to Submit

1. **E-R Diagram** with explanatory notes.
2. **Relational Schema** consistent with your final E-R design.
3. **SQL Scripts** for table creation, constraints, and indexes.
4. **Data Loading Scripts** or programs used to populate the database.
5. **Sample Queries** (SQL files) with example outputs (you can demonstrate these during the lab).
6. **Interface Code** (Java, Python, or command-line scripts).
7. **README File** explaining the project structure, setup steps, and how to run your code.
8. **ZIP Archive** containing all files (.sql, .java, .py, .txt, etc.).

Avoid platform-specific or IDE-dependent submissions. Provide plain, compilable source files and clear instructions.

## 6. Grading Criteria

| **Component** | **Points** |
|---|---|
| E-R Design | 30 |
| Relational Design (constraints, normalization) | 30 |
| Data Creation (realism, integrity) | 10 |
| User Interfaces (functionality, usability) | 30 |
| Creativity / Exceptional Work | +10 bonus |
| **Total** | **100 points** |

## 7. Collaboration Policy

- Group integration work: each student should be able to defend and explain the project.
- A detailed explanation is required from each team member.
- All team conflicts should first be resolved internally, then brought to the instructor if unresolved.
- If any team member did not contribute during the project implementation, you may mention this when submitting the project files.

## 8. Notes

- Submit all deliverables electronically by the due date.
- Late submissions without prior request will incur penalties.
- Creativity is encouraged: recommendation features, dashboards, real-time order tracking, or triggers for automatic operations can earn bonus credit.