CS 202: Project - Online Food Ordering System

Spring 2025

Important Information

Project Groups Sheet: C202 - Spring 2025 - Project Group

Deadlines
Part 1: **26th of April 2024, 11:59 PM**
Part 2: **20th of May 2024, 11:59 PM**

Note 1: The **deadline** is very **strict** and it will **NOT** be changed under any conditions. You can submit your assignment/project with **ONLY 3 DAYS** delay, but be aware for each day (**EVEN 1 MINUTES PASS THE DEADLINE**) you will receive a **PENALTY (-10)** for each day.

Note 2: There will be a **demo session** (Day, Time and Location will be announced), which you are required to present your work in about 15 minutes and answer some questions in-person (NO ONLINE DEMO SESSION).

Friendly Reminder: On the Internet there exists similar projects, copying and submitting any of this work will result in a Penalty (-100) without any further discussion or considerations. Your codes will be checked by *Stanford University's* **MOSS (Measure Of Software Similarity) - Plagiarism Detection** system.

TAs' Email Address:

- Amin Alamdari amin.alamdari@ozu.edu.tr
- Anıl Doğru: anil.dogru@ozu.edu.tr
- Tuğçe Özgirgin: tugce.ozgirgin@ozu.edu.tr

Honor Code

As part of our commitment to academic integrity and ethical conduct, all students undertaking the *Database Management System* assignment/project are expected to adhere to the following honor code:

1. **Original Work:** All assignment/project submissions must be the original work of the individual student or group. Plagiarism or any form of unauthorized collaboration is strictly prohibited.
2. **Citations and References:** Properly cite and reference any external sources, including books, articles, websites, or any other materials used in the assignment/project. Failure to acknowledge sources is considered a violation of academic honesty.
3. **Independent Effort:** Each student or group is expected to complete the assignment/project independently without seeking unauthorized help from other students, online sources, or any other external parties unless explicitly permitted by the instructor.
4. **Honesty and Authenticity:** The assignment/project submission must accurately represent the student's or group's own understanding and effort, reflecting the knowledge and skills gained throughout the course.
5. **Respect for Academic Integrity:** Uphold the principles of academic integrity and abide by the rules and guidelines provided by the instructor and the institution.
6. **Adherence to Course Policies:** Follow all guidelines, deadlines, and instructions specified in the assignment/project description and course syllabus. Non-compliance may result in penalties.
7. **Report Violations:** Students are encouraged to report any suspected violations of the honor code to the instructor for appropriate investigation and action.

Consequences of Violating the Honor Code: Violations of the honor code will result in disciplinary action as per the course or institutional policies. Penalties may include but are not limited to failing grades on the assignment/project, failing the course, or academic probation. *By submitting their assignment/project, each student or group acknowledges their understanding and commitment to upholding this honor code.*

1 Description

You will design and implement an Online Food Ordering System, similar to real-world platforms like Yemeksepeti, GetirYemek, Uber Eats, or DoorDash. The goal is to create a system that handles food ordering, menu browsing, order tracking, and basic user/restaurant management using database technologies and web programming. Users should be able to register, browse restaurants and menus, place orders, and track them. Restaurant managers should be able to manage their menu items, discounts and order flow. Also, Restaurant managers should be able to display a summary of the last month (e.g., total profit, item-wise profit, the most expensive order, the most frequently ordering customer). The project is divided into **two main parts**.

2 Part 1 – Database Design

You are required to:

1. Design the Entity-Relationship (ER) Diagram using proper notation (as shown in class). Hand-drawn diagrams or auto-generated MySQL ERs will not be accepted.
2. Create DDL SQL File to define the schema.
3. Create DML SQL File to populate the schema with meaningful sample data.
4. Write a Report including:
   - ER diagram
   - List of functional dependencies
   - **Explanation** of normalization (up to 3NF)
   - Key design decisions and constraints explanations

2.1 Entities and Relations (Minimum Required)

You are required to create an ER diagram representing (**at least**) the following entities:

- **User:** A customer or restaurant manager who can browse restaurants and place or manage orders. Each user must have a username and password to log into the platform. Additionally, a user may have multiple addresses and phone numbers.
- **Restaurant:** Each restaurant has a name, cuisine type, address (including city), and a menu. It is managed by a user. A restaurant can be associated with multiple keywords, defined by the manager, which are used for searching and browsing.
- **MenuItem:** Items offered by a restaurant. Each item must have a name, description, price, and image. The restaurant manager may also define time-limited discounts for menu items.
- **Cart:** Represents an order placed by a customer and fulfilled by a restaurant. A cart can contain multiple items with specified quantities. It has a status representing its progress (e.g., preparing, sent, accepted).
- **Ratings:** After an order is accepted, customers can leave a rating (1 to 5) and an optional comment. Average rating values directly influence restaurant visibility in search results.

The relationships in your ER diagram should include (**at least**) the following:

- A **Customer** can place multiple **Orders (Carts)**.
- Each **Cart** includes one or more **MenuItems** from a single **Restaurant**.
- A **Restaurant** can have many **MenuItems** and receive many **Carts**.
- A **Restaurant** is managed by a **RestaurantManager** (a user with restaurant privileges).
- A **RestaurantManager** can manage one or more **Restaurants**.
- A **Customer** can leave reviews (ratings and optional comments) for the **Restaurants** they have ordered from.
- Each **Cart** is linked to delivery-related details such as status and timestamp (even if the delivery is not implemented).

⚠ **Important Note 1**: For drawing ER Diagram, you can use tools such *Microsoft Excel*, Draw.io, LucidChart, SmartDraw, or any other application or software do you now. Hand-drawn ER will **NOT** be accepted at all.
⚠ **Important Note 2**: Your **ER Diagram**'s style SHOULD (**IS REQUIRED**) to be based on what the professor taught you during lectures. Other styles of ER Drawing styles are **NOT** accepted.
⚠ **Important Note 3**: Automatic ER diagram generation through MySQL is **prohibited**.

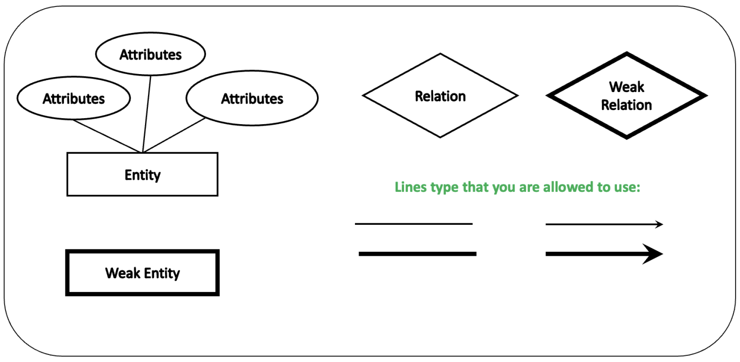

Figure 1: Allowed Symbol Types in the ER-Diagram.

2.2 Data Definition Language (DDL.sql) SQL Code

Write SQL scripts to create all tables defined in your ER diagram:

- Include Primary Keys (PK), Foreign Keys (FK), Unique constraints, NOT NULL, CHECK conditions.
- Establish foreign keys (FK) to maintain relational integrity.
- Implement necessary data types, constraints, and indexes.
- Ensure referential integrity among tables.

2.3 Data Modification Language (DML.sql) SQL Code

You must write DML SQL scripts to insert sample data into the tables, ensuring your database has at least:

- 5 Restaurants with some keywords
- 3 Restaurant Managers
- 15 Menu Items
- 5 Customers
- 10 Carts (with multiple items from different customers)
- 10 Review for each restaurant

Tip: Use real restaurant/menu data for inspiration, but do not copy real platforms.

3 Part 2 – Full Stack Web Application

You will develop a web application that uses your database from Part 1 to provide an interactive interface.

3.1 Requirements

- **Backend:** Python with Flask.
- **Database Connection:** Only `MySQL Connector/Python` is allowed. ORMs such as `SQLAlchemy` are strictly **prohibited**.
- **Frontend:** Students are free to choose the technologies for the frontend. It must be fully functional and compatible with the backend logic. You may use technologies such as HTML5, CSS3, JavaScript, Bootstrap, jQuery. You may also use Flask's built-in templating engine (Jinja2) to render dynamic content. The visual aesthetics and UI design will **not** be directly graded. However, all functionalities must be accessible and usable through your web interface.
- **SQL Queries:** All SQL operations (INSERT, DELETE, UPDATE, SELECT) must be manually written. Code tricks and non-standard shortcuts are not allowed.
- ❗ **Note:** Utilizing any other libraries except `Flask` and `mysql-connector-python` is **prohibited**.

3.2 Functional Requirements

The system will support two types of users: **Restaurants** and **Customers**. Restaurant users will be able to define their restaurant profile (including city, name, and keywords), create and manage their menu, assign discounts to specific items for limited periods, and track sales performance. Customer users will be able to search for restaurants based on keywords, place orders by adding items to a cart, and rate their experience after the restaurant has accepted the order.

The following sections outline the detailed functionalities available to each user type:

User Types

1. Restaurant Users:

- Can set specific keywords to improve search visibility.
- Must create a menu. Each item includes name, description, image, and price.
- Can define time-limited discounts for certain products.
- Can accept incoming order requests manually.
- Average rating (out of 5) is shown only after receiving at least 10 ratings. Restaurants with fewer than 10 ratings will appear in search results with a rating of 0 and a “New” label.
- Can view sales statistics for the past month, including:
  - Total revenue and total number of orders,
  - Total quantity sold and revenue per menu item,
  - The customer who placed the most orders in the last month,
  - The customer with the highest-value cart in the last month (including cart details).

2. Customer Users:

- Can search for restaurants based on keywords. Restaurant and customer must be located in the same city.
- Search results should be ranked by keyword match and average rating (if available). If a restaurant has fewer than 10 ratings, its average rating will not be displayed, and it will be marked as “New”.
- Can add multiple items and quantities to a cart.
- Finalizes the cart and sends an order request to the restaurant.
- After the restaurant accepts the order, they can rate and optionally comment within 24 hours.

Order Process Flow

- **Cart Preparation:** Customer builds the cart with desired items.
- **Order Sent:** Customer confirms and sends the order to the restaurant.
- **Restaurant Accepted:** Restaurant reviews and accepts the order.
- Each stage must be represented and tracked in the database.

3.3 Excluded Features

- Payment and delivery logistics are outside the scope of this project.

3.4 Non-Functional Requirements

- All SQL queries must be clearly written and documented.
- No use of ORM libraries or any automated query builders.
- A final demo must be presented showing how the application works and how each query is executed.

4 Required Technologies and Related Tutorials

For implementing the Online Food Ordering System, the following technologies and tools are required:

- Programming Language and Web Framework:
  - `Python` → Used for backend development and application logic. You can find some tutorials in the following links:
    1. Python Official Tutorial
    2. Tutorials Point: Python Tutorial
    3. Programming with Mosh: Python Full Course for Beginners [2025]
    4. Bro Code: Python Full Course for free (2024)
    5. Tech With Tim: Python MySQL Tutorial - Setup & Basic Queries (w/ MySQL Connector)
    6. Telusko: Python Database Connection — MySQL
    7. TheCodex: Python and MySQL - Getting Started with MySQL ⇒ **Macbook**
  - `Flask` → A lightweight Python web framework used to build the web interface and manage routes. You can find some tutorials in the following links:
    1. Flask Official Tutorial (Blog App Walkthrough)
    2. freeCodeCamp.org: Learn Flask for Python - Full Tutorial
    3. freeCodeCamp.org: Flask Course - Python Web Application Development
    4. Corey Schafer: Python Flask Tutorial: Full-Featured Web App
    5. Tech With Tim: Python Website Full Tutorial - Flask, Authentication, Databases & More
  - **Note:** You can directly setup the **Flask library** by "`pip install Flask`".
- Database System:
  - `MySQL` → Relational database management system (RDBMS) for storing and managing banking data. You can find some tutorials in the following links:
    1. MySQL Official Tutorial
    2. www.mysqltutorial.org Website
    3. Installing MySQL on Windows 10
    4. Bro Code's MySQL Full Course for free
    5. Programming with Mosh's SQL Course for Beginners [Full Course]
- Database Connectivity:
  - `MySQL Connector/Python` → A pure Python MySQL client used to connect Python applications with MySQL databases. You can find some tutorials in the following links:
    1. MySQL Connector/Python Developer Guide (Official Website)
    2. Tech With Tim: Python MySQL Tutorial - Setup & Basic Queries (w/ MySQL Connector)
  - **Note:** You can directly setup the **mysql-connector library** by "`pip install mysql-connector-python`".
- Development Environment and Tools:
  - IDE (Integrated Development Environment):
    1. Visual Studio Code
    2. PyCharm – You can get the professional edition free with your edu.tr email.
  - MySQL Workbench → GUI tool for database design and query management.

5 Submission Guidelines

5.1 General Guidelines

Guidelines to ensure a successful project completion:

- Ensure clarity and correctness in the ER diagram, representing all necessary entities and relationships. You are required to use the symbols as presented in Figure 1, otherwise you get penalty.
- DDL SQL code **should** accurately create the database schema based on the ER diagram.
- Use appropriate data types, primary and foreign keys, constraints, and naming conventions.
- You **should** write your own SQL queries manually. You **are NOT allowed** to use a framework that automatically generates SQL statements and not allowed to use frameworks that might abstract the database interaction such as ORM frameworks.
- Your program is **required** to handle all the exceptions and errors.

5.2 Submission Guideline and Requirement for Part 1

First Part's Deadline: 26th of April 2024, 11:59 PM (Tuesday)

1. Please compress all your files and submit a single **zip** file.
2. The name of your **zip** file should look like the following:
   CS202_Group[id of your group]_Project_Part1.zip for example *CS202_Group61_Project_Part1.zip*
3. Inside of the **zip** file you should provide the followings:
   - (PDF file for Report and ER-Diagram) Group[id of your group]_Project_Report.pdf
   - Inside the your report write your Group ID and members of the group and their student ID.
   - DDL.sql
   - DML.sql
4. Failing to comply with these guidelines will result in a **penalty**.
5. ⚠ Automatic ER diagram generation through MySQL is **prohibited**.
6. ❗ **Note:** You can add ER-Diagram separately in `PDF`, `JPEG`, or `PNG` format in you `.zip` file.
7. Inside of the **zip** file you should provide the working version of the program. Otherwise, you get **zero** from this part of the project.

5.3 Submission Guideline and Requirement for Part 2

Second Part's Deadline: 20th of May 2024, 11:59 PM (Tuesday)

1. Please compress all your files and submit a single **zip** file.
2. The name of your **zip** file should look like the following:
   CS202_Group[id of your group]_Project_Part2.zip for example *CS202_Group61_Project_Part2.zip*
3. Inside of the **zip** file you should provide the followings:
   - (PDF file for Report and ER-Diagram) Group[id of your group]_Project_Report.pdf
   - Inside the your report write your Group ID and members of the group and their student ID.
   - DDL.sql
   - DML.sql
4. Failing to comply with these guidelines will result in a **penalty**.
5. ⚠ Automatic ER diagram generation through MySQL is **prohibited**.
6. ❗ **Note:** You can add ER-Diagram separately in `PDF`, `JPEG`, or `PNG` format in you `.zip` file.
7. Inside of the **zip** file you should provide the working version of the program. Otherwise, you get **zero** from this part of the project.
8. ❗ **Note:** If you revised your ER-Diagram, you are required to provide both versions of your ER-Diagram, old and updated version, and in your report clearly describe why you need to change your design.

6 Final Notes

This project is designed to give you hands-on experience in modern web development with databases by integrating **Python, Flask, and MySQL**. You will apply both theoretical and practical knowledge in building a real-world system.

- Focus on the **correctness and clarity** of your ER diagram, DDL, and DML SQL code.
- Automatic ER diagram generation through MySQL is **prohibited**.
- Follow **best practices in database design**, including normalization and referential integrity.
- Ensure that your Flask application is **modular, readable, and testable**.
- Write all SQL operations manually and securely using `MySQL Connector/Python`.
- You can directly setup the **mysql-connector library** by "`pip install mysql-connector-python`" and **Flask library** by "`pip install Flask`".
- The use of any libraries other than `Flask` and `mysql-connector-python` is **strictly prohibited**.
- Design a web interface that provides all functionalities clearly, even if the aesthetics are simple.

Make sure your final submission includes **a working version** of your application, and be prepared to present your project during the in-person demo session.

For any questions, reach out to the course TAs or the instructor.

Be Successful!