BLG317E
Database Systems Project Guidelines

Research Assistant: Erhan Biçer - bicer21@itu.edu.tr
Research Assistant: Sümeyye Öztürk ozturks20@itu.edu.tr

Project Overview

In this project, students are expected to develop a database system based on an idea of their choice, focusing on designing a suitable data model, implementing CRUD operations, and demonstrating the system through a RESTful API. The project will be completed in stages, with specific milestones at each step.

By following these guidelines, you will develop a complete and well-documented database system, showcasing your understanding of database design, implementation, and API integration. Ensure each phase is completed thoroughly to achieve success in the final presentation.

Good luck!

Project Milestones:

Milestone 1: Project Proposal and Group Formation

Due Date:

- Submit your proposal form by **04.11.2024**.

Requirements:

1. **Group Formation:**
    - Groups must consist of **2 to 4 members**.
    - Clearly state the group members' names and roles (if defined).

Project Idea:

- Write a document in the proposal form (max 1 page) describing the idea for the project.
- The idea must be complex enough to require the design of a database with at least **8 tables with proper relationships**.
- Think of all the operations your project would involve (data insertion, updates, deletions, and queries).
- Explain why a database is necessary for your chosen idea and provide an overview of the data it will store and manage.

Note:

- The project idea will be reviewed and approved by the instructor before you proceed to the next stage.
- Ensure that your idea allows for **CRUD operations** and **complex queries**, as these will be required in the final project.

Milestone 2: ER Diagram and Data Model Design

Due Date:

- Submit your detailed ER diagram by the specified date after your proposal is approved.

Requirements:

1. **ER Diagram:**
   - Design an **Entity-Relationship (ER) diagram** that accurately represents your data model.
   - Your data model should consist of **at least 8 tables**, with all necessary **non-key columns** clearly defined.
   - Identify all relationships between tables, including **One-to-One** and **One-to-Many** relationships.
2. **Data Population:**
   - You can use **dummy data in** your tables to simulate real-world scenarios.
3. **CRUD Operations:**
   - Plan how CRUD (Create, Read, Update, Delete) operations will be performed on your data.
   - Consider how users will interact with your data model through queries and updates.

Milestone 3: Database Queries and RESTful API Development

Requirements for the Final Project:

1. **Complex Queries:**
    - Implement **complex queries**, including **nested queries**, using your data model.
    - Be prepared to demonstrate how these queries interact with your database.

RESTful API:

For this project, students are required to develop a **RESTful API** that interacts with their database system, along with implementing authentication for secure access. The API must allow authorized users to perform CRUD operations on the database.

Requirements:

1. **Database Choice:**
    - You are free to choose any database management system (DBMS) you are comfortable with, such as **MySQL**, **PostgreSQL**, **MongoDB**, or any other relational or non-relational database.
    - You should avoid manual data manipulation. All the rows in all the tables must be inserted, deleted and updated through this RESTful API.
2. **Programming Language and Framework:**
    - You can implement your API using any platform or language you're familiar with. Some popular choices include:
        - **Node.js with Express** (JavaScript)
        - **Flask or Django** (Python)
        - **Spring Boot** (Java)
    - Ensure that your API endpoints are designed in a RESTful manner, adhering to the best practices for REST API development.
3. **API Endpoints:**
    - The API must cover the **CRUD operations** for the tables you have designed. Each endpoint should correspond to one or more operations on the database. Examples:
        - `GET /items` to retrieve data from a table.
        - `POST /items` to add new data.
        - `PUT /items/:id` to update existing data.
        - `DELETE /items/:id` to remove data from the database.
    - You are also expected to implement **complex queries** through the API. This could involve nested queries or advanced filtering options.

4. **Authentication Requirement:**
   Implement authentication to restrict access to the API. You can use either:
   - Token-based authentication (e.g., JWT- JSON Web Tokens) to allow users to securely log in and perform operations.

User Roles: If applicable, define different user roles (e.g., Admin, Regular User) to control what operations different users can perform.

5. **API Documentation:**
   - To ensure the API is well-documented, you are required to use **Swagger** (https://swagger.io) for creating interactive API documentation.
   - Swagger allows you to clearly define the endpoints, request types, response formats, and parameters. This will make it easier for users (or evaluators) to understand how to interact with your API.

Example Structure:

- **Database Operations:** Define how each CRUD operation will work on the chosen tables.
- **Endpoints:** Clearly list the API endpoints, explaining the HTTP methods (GET, POST, PUT, DELETE) and the data expected at each endpoint.
- **Authentication :**
  - **Login endpoint (POST /login)** where users submit their credentials.
  - **Register endpoint (POST /register)** for new users to sign up.
  - Protected routes: Ensure all CRUD and query operations are restricted to authenticated users only. Unauthorized users should receive an appropriate error response (e.g., HTTP 401 Unauthorized).

Platforms & Tools Suggestions:

- You can deploy your API locally or use cloud platforms such as **Heroku**, **AWS**, or **Azure** to showcase it in a live environment during your demo.
- You can use **Postman** or **curl** to test and demonstrate the functionality of your API endpoints.
- Use libraries such as Passport.js (Node.js) , Flask-JWT-Extended(Flask), Django REST Framework with JWT(Django), or similar tools for implementing secure authentication.

Milestone 4: Demo, Presentation, and Documentation

Final Project Deliverables:

1. **ER Diagram** with relationships and non-key columns clearly defined.
2. **Fully implemented database system** capable of handling CRUD operations and complex queries.
3. **RESTful API** with Swagger documentation.
4. **Demo Presentation:**
    - Prepare and deliver a **live demo** of your project.
    - The demo should highlight key functionalities, including CRUD operations, complex queries, and how the API works.
    - Be ready to explain design decisions and answer questions about your database model and implementation.
5. **Project Documentation:**
    - Prepare detailed project documentation that includes:
        - **Overview of your project idea**.
        - **ER diagram and data model explanations**.
        - **Descriptions of CRUD operations** and their implementations.
        - **API design** and how each endpoint works (Swagger documentation).
        - **Examples of complex queries** with explanations.
        - **Challenges and solutions** encountered during the project development.
    - Documentation should be professional and clear, summarizing the entire development process.