Cloud Ready Application Development - IT 4118

Assignment 02
102 Batch – Year 4 Semester 01

Objectives:

1. To understand and implement RESTful APIs using Spring Boot and Data JPA.
2. To design and connect APIs to a UI built using HTML, CSS, and JavaScript.

Requirements:

Database Setup

- Create a **Customer** table in a database (use MySQL, PostgreSQL, or any RDBMS of your choice) or Use JPA.
- The **Customer** table should have the following fields:
  - `id` (Primary Key, Auto-increment)
  - `first_name` (String, 50 characters)
  - `last_name` (String, 50 characters)
  - `email` (String, unique, 100 characters)
  - `phone` (String, 15 characters)
  - `address` (String, 255 characters)
  - `created_at` (Timestamp, auto-generated)

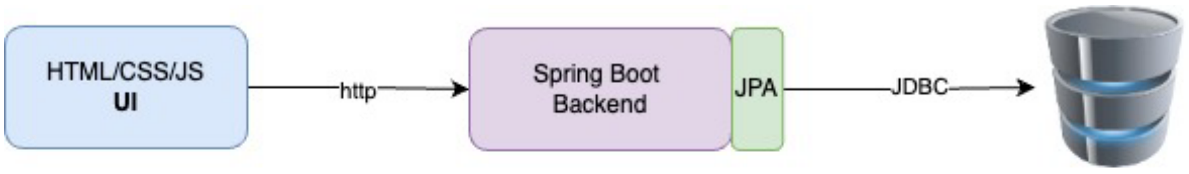
REST API Development

- Create a Spring Boot application with the following endpoints:
  1. **GET /customers**: Fetch all customers.
  2. **GET /customers/{id}**: Fetch a customer by ID.
  3. **POST /customers**: Add a new customer.
  4. **PUT /customers/{id}**: Update an existing customer's details.
  5. **DELETE /customers/{id}**: Delete a customer by ID.

UI Development

- Develop a simple UI using **HTML**, **CSS**, and **JavaScript** (no frameworks like React or Angular).
- The UI should have the following features:
  1. A table to display all customers.
  2. A form to add or edit customer details.
  3. Buttons to delete a customer and submit the form.
- Use JavaScript to make API calls and dynamically update the UI.

Expected Architecture



Submission Requirements:

1. **Codebase**:
   - Submit the complete source code for the Spring Boot application.
   - Include the HTML, CSS, and JavaScript files for the UI.

     UI + Backend in ZIP format

2. **Database Script**:
   - Provide an SQL script to create and populate the **Customer** table with sample data if applicable.

3. **Documentation**:
   - Include a README file explaining how to set up and run the project locally.

Evaluation Criteria:

1. **Functionality**:
   - Does the API adhere to RESTful principles?
   - Are all endpoints functional?
   - Does the UI connect seamlessly with the API?

2. **Code Quality**:
   - Is the code modular and well-structured?
   - Are naming conventions and best practices followed?

3. **UI Design**:
   - Is the UI user-friendly and responsive?
   - Are the interactions smooth and error-free?

4. **Documentation**:
   - Is the README file comprehensive?
   - Are setup instructions easy to follow?

Deadline:

The assignment must be submitted by **Jan 31st 2025**.