Sprint 2 Evaluation Project -

Spring Boot RESTful API with React JS

Guidelines:

Demonstrate a working Full Stack application built with a Spring Boot back-end and a React Js front-end.

The application must feature the following:-

1. Display the list of records in the database
2. Display a single record from the database
3. Add a record to the database
4. Update a record in the database
5. Delete a record from the database

There are 7 teams in all with 5 members each.

The teams will be composed as follows from the excel doc in this folder :-

1. Team 1 :- Participants with serial numbers 1,8,15,22,29.
2. Team 2 :- Participants with serial numbers 2,9,16,23,30.
3. Team 3 :- Participants with serial numbers 3,10,17,24,31.
4. Team 4 :- Participants with serial numbers 4,11,18,25,32.
5. Team 5 :- Participants with serial numbers 5,12,19,26,33.
6. Team 6 :- Participants with serial numbers 6,13,20,27,34.
7. Team 7:- Participants with serial numbers 7,14,21,28. (with 4 members)

Each participant will present one aspect of the project ex: Presentation, Postman demonstration etc.

Therefore, it is important that every participant should be well versed with all aspects of the project.

**Deliverables:**

1. **Project Presentation :-**
   1. Less than 10 slides
   2. Slide 1 - Title of the project
   3. Slide 2 - Names of the team members
   4. Main Features of the project
   5. Process Flow diagrams of each feature

*Example :-* Add Customer,List Customers, Display a single Customer’s details, Update a Customer etc.

1. **Spring Boot Back-end Project**

**Technologies:**

* Spring Boot
* Spring MVC
* Spring Data JPA

Build the REST APIs for the following HTTP methods:-

* **GET**
* **POST**
* **PUT**
* **DELETE**

**The GET endpoints must be at least 2.**

1. Retrieve a list of entity instances
2. Retrieve a single entity by appending a parameter to the URI

**Database:**

You may use your choice of database - H2, MySQL or PostgreSQL

**Back-end Project demonstration:**

1. Demonstrate all API endpoints in Postman
2. Updates and new records must reflect in the database.
3. Explain implementation in the code
4. Make a change to the API as requested and implement it end to end .

Ex: add a field to a form and have the input reflected in the database.

1. Answer technical questions.
2. **React JS Back-end Project**

**Technologies:**

* HTML5
* CSS3
* Bootstrap 4 or 5
* React JS > version 16
* Redux (optional)

Build the complete user interface using aforementioned presentation technologies.

Utilize React Context API or Redux to store state.

Consume the back-end REST API endpoints using AXIOS.

Data to be retrieved and displayed in the UI must be retrieved from the database.

**Back-end Project demonstration:**

1. Demonstrate a working front end application
2. All the back-end REST APIs must be accessible from the front-end
3. Updates and new records must reflect in the database.
4. Explain implementation in the code
5. Display an alert to the user when a record is successfully added, updated or deleted.
6. Introduce a 3 second delay into the application when the list of records is being retrieved, or when a record is being updated or added.
7. The page displaying the list of records should show a Spinner or Progress Bar only while the records are being retrieved. The Spinner or Progress Bar should disappear when the page is rendered.
8. The page(s) where a record is added or updated should have the Save/Update button disabled while a new record is being saved or updated.
9. On successful save or update, the user should only then be navigated to the list of records.
10. On Delete, the user must immediately be navigated to the list of records. (Optimistic Deletes)
11. Unsuccessful Creates, Updates or Deletes must be followed by a user friendly error message on the screen.
12. Use **Toast Notifications** to display alerts and messages to a user. It can be a success message, warning message, or custom message.
13. Answer technical questions.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*