AlgoBrave Tech Assignment

Intern - Engineering Challenge

Welcome to the **AlgoBrave Tech Assignment** 👋

This assignment is designed to evaluate your ability to build a **clean, well-structured full-stack application** using modern technologies and best practices. You will work on both **Frontend (FE)** and **Backend (BE)** layers, implement **full CRUD functionality**, and demonstrate your approach to **state management, API design, and testing**.

TASK 1: SETUP NEW FRONTEND APPLICATION (NEXT.JS + REACT)

Create a **Next.js** application that integrates with the backend API built in Task 2.

REQUIREMENTS:

- Set up a Next.js application using React.
- Use Redux Toolkit for state management.
- (Optional but preferred) Use Tailwind CSS for styling.
- Configure API communication with the backend (e.g., environment-based API base URL).
- Initialize the Redux store and implement slices/thunks for managing user data.
- Make the application mobile responsive

CRUD UI REQUIREMENTS (Frontend)

Implement full CRUD functionality in the UI using the backend APIs:

1) List Users (Read)

- Create a **User List page** that:
    - Fetches user data from the backend
    - Displays users as cards or rows
    - Shows **name, email, and phone**
    - Includes loading, empty, and error states

2) View User Details (Read by ID)

- Provide a way to view a single user (page or modal) by calling the backend “get by id” endpoint.

3) Create User (Create)

- Provide a **Create User** form (page or modal) with fields:
    - name, email, phone
- On submit, call the backend, create endpoint and update the UI accordingly.

4) Edit User (Update)

- Provide an **Edit User** form (page or modal) that:
    - Pre-fills existing user data
    - Updates the user via the backend update endpoint
    - Reflects changes in the UI

5) Delete User (Delete)

- Add a **Delete** action per user with a confirmation step.
- On confirm, call the backend delete endpoint and remove the user from the UI.

TESTING REQUIREMENTS (Frontend) - Nice to have 🙂

- Use **Jest** and **React Testing Library**.
- Write unit tests covering at minimum:
    - User list renders correctly.
    - Loading and error states
    - Creating a user updates the UI state
    - Editing a user updates the UI state
    - Deleting a user triggers the correct action and updates the UI
- Include at least one test for Redux logic (slice reducer and/or async thunk).

TASK 2: SETUP NEW BACKEND API (JAVA SPRING BOOT)

Build a Java Spring Boot backend application to support and integrate with the frontend built in Task 1.

The backend must expose RESTful APIs to manage users and must be fully connected with the frontend so that all CRUD operations work end-to-end.

Technical Scope

- Use **Java with Spring Boot**
- Build a **RESTful API**
- Persist data using a **relational database** (e.g., PostgreSQL, MySQL, H2, etc.)
- The backend must be usable by the frontend application from Task 1
- The application must run locally

Functional Requirements

Implement backend APIs to support full **CRUD operations** for users:

- Create a user
- Retrieve all users
- Retrieve a user by ID
- Update a user
- Delete a user

The backend must manage user data including at least:

- `name`
- `email`
- `phone`

You are free to add more user data, design the endpoints, request/response models, database schema, and overall structure.

Design & Engineering Expectations

Your backend solution should demonstrate good backend engineering practices, including:

- A clear and maintainable overall project structure
- Well-designed and intuitive APIs
- Proper handling of invalid input and failure scenarios
- Visible use of logging for application behavior and/or error tracking
- Code that is clean, readable, and easy to follow

The overall design approach and structure are left to you. Make reasonable engineering decisions and document them where appropriate.

Testing (Nice to have)

- Unit tests and/or integration tests for backend logic and/or REST endpoints

---

SUBMISSION INSTRUCTIONS:

1. Create a new repository to house the complete fullstack project.
2. Commit your code to the repository, ensuring all necessary files and folders are included.
3. Provide a README file with clear instructions on how to set up and run both the frontend and backend applications.
4. Include details on how to run the tests for both the frontend and backend.
5. If any additional documentation or explanations are necessary, include them in the repository.

NOTE:

- By completing these tasks, you'll showcase your ability to build a cohesive fullstack application using Java Spring boot, Redux Toolkit, and React.
- You're encouraged to seek clarification if you have any questions about the tasks.

Happy Coding!!!