Assignment

Copy

This assignment combines a full-stack development project using a variety of popular tools and practices. Here's a breakdown of the main tasks and guidelines for each part to help you get started:

Part I: Backend with NodeJS, Express, and MongoDB / Assignment 1

1. **API Development:** You will refer to **Assignment 1** to create a backend application using NodeJS, Express and MongoDB. This application should expose RESTful APIs for performing CRUD operations on user and employee data (for example, signup, login, adding, viewing, updating, and deleting employee records).

- Build a RESTful API using Node.js, Express, and MongoDB.
- Set up endpoints for CRUD operations on user and employee data, such as:
  - **User Endpoints**: `POST /signup`, `POST /login`
  - **Employee Endpoints**: `POST /employees`, `GET /employees`, `GET /employees/:id`, `PUT /employees/:id`, `DELETE /employees/:id`, `/search`
- Use validation to handle various data inputs and error scenarios.

2. **Axios for API Consumption**

- On the ReactJS frontend, make HTTP requests to your backend API using Axios. This ensures separation between frontend and backend and allows you to efficiently make data requests to the backend.

Important: Ensure the MongoDB connection is not established on the frontend; it should only reside in the backend.

3. **Docker Setup or Cloud Deployment**

- **Docker**: Create a `docker-compose.yml` file to orchestrate MongoDB, the backend, and the frontend. Each service (frontend, backend, MongoDB) should have its own Dockerfile if deploying with Docker Compose.

OR

- **Cloud Deployment**: Alternatively, deploy the backend and frontend on a cloud platform (e.g., Heroku ↗ / Vercel ↗ / Render ↗) for live access.

Update the backend (assignment 1) whenever required to accommodate the requirement of the frontend.

Part II: Frontend with ReactJS

👉🏻 Try to use features from **TanStack** ↗ to better prepare for future job.

1. **Application Setup and GitHub repository**
   - Initialize a ReactJS app named in the specified format (`studentID_comp3123_assignment2_reactjs`) using `create-react-app` Then, *set up your GitHub repository with the same name and commit/push your changes regularly with descriptive messages.*
   - *NOTE: if you are using docker-compose then make a single folder to keep the frontend and backend. For example,* Organize the project structure in a single folder if using Docker:

```
studentID_comp3123_assignment/
├── docker-compose.yml
├── frontend/
└── backend/
```

2. **Routing and Navigation**
   - Use `react-router-dom` for screen navigation:

- **Login**
- **Signup**
- **Employee components**

3. **Login and Signup Screens**
   - Implement login and signup screens with form validations.
   - Use controlled components to manage form input states.
4. **Session Management**
   - Store a user session token in `localStorage` to persist user authentication or Context API or Redux.

```
localStorage.setItem('token', response.data.token);
```

5. **Employee Management**
   - After login, navigate users to an **Employee List** screen displaying all employees in a table format.
   - Provide CRUD operations:
     - **Add Employee**: A form to add new employees.
     - **View Details**: A screen/modal displaying specific employee details.
     - **Update Information**: A form for editing employee information.
     - **Delete Employee**: A button to remove an employee record.
6. Search Functionality and d File upload Functionality

   a. Add employee search criteria for department or position and display them in the appropriate table format. This feature will allow users to search for employees based on department or position, showcasing your ability to handle dynamic queries and responsive UI. (***Add new REST API to backend***)

   b. Allow to upload employee's profile picture from frontend. To accommodate this requirement please update backend API to allow picture upload. for add, update and get employee.

7. **User Experience**
   - For a professional UI/UX design:
     - Use Material-UI, Bootstrap, or custom CSS for styling.

    - Make sure components are responsive.

8. **Logout and Redirect**
    - Implement a logout button to clear user sessions and redirect users to the login page.

Previous
Home

Next
Reference Sample Screen Designs

Last updated 1 month ago