Frontend Developer Lab

Objective:

The goal of this task is to assess your ability to create a simple full-stack web application that uses a React front-end and a Node.js back-end to fetch data from an external API and display it to the user.

Task:

Develop a small web application where:

1. The front-end allows the user to click a button to retrieve data.
2. The Node.js server makes a request to a mock API (provided below) and returns the result to the front-end.
3. The fetched data should be displayed in a simple list on the front-end.

Requirements:

1. Front-End (React)
    a. Create a simple web page with a button that triggers the data fetch.
    b. Display the returned data (from the back-end) in a list or table format.
    c. Use basic styling with CSS.

2. Back-End (Node.js)
    a. Use Node.js and Express to create a server that handles the API request.
    b. When the button is clicked, the front-end sends a request to the Node.js server.
    c. The Node.js server then calls the mock API and returns the result to the front-end.
    NOTE: Ensure that the API call to the mock API is made from the back-end (Node.js server) and not directly from the front-end.
    HINT: You can use axios

Mock API:

Here's a simple mock API endpoint to simulate fetching data:

- API Endpoint: https://jsonplaceholder.typicode.com/posts
- The API returns a list of posts in JSON format. Each post contains fields like:
    - `userId`: User ID
    - `id`: Post ID
    - `title`: Post Title
    - `body`: Post Body

Example Response:

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "Sample Post Title 1",
    "body": "This is the body of the first sample post."
  },
  {
    "userId": 1,
    "id": 2,
    "title": "Sample Post Title 2",
```

```
    "body": "This is the body of the second sample post."
  }
]
```

Steps:

1. Front-End:
    a. Create a button labeled “Fetch Posts.”
    b. When clicked, it sends a request to the back-end.
2. Back-End:
    a. Set up a route (e.g., `/api/posts`) in Node.js that makes a request to the mock API (https://jsonplaceholder.typicode.com/posts).
    b. Return the response data from the mock API to the front-end.
3. Display Data:
    a. Display the fetched data (titles of posts) in a list format on the front-end.

Submission:

1. Push your code to a public GitHub repository.
2. Include instructions in the README for running the application locally.
3. Once completed, share the link to your GitHub repository.

Bonus Points:

- Implement error handling for the API call.
- Use hooks (like useEffect) to fetch data after the button is clicked.
- Display additional post details like the body or userId.
- Implement local authentication

END LAB