**# Notes for the Full stack**

**## 1. Backend**

- Make sure the backend is up and running

- Test the backend CRUD before you move on

**## 2. Frontend setup**

- create vite project with the name "client", install dependencies inside the project

- install axios & react-router-dom

- Create all the neccessary views according to the wireframe

- Set the routes (BrowserRouter & Routes)

- Follow the platform or follow your code from React routing practice

**## 3. DashboardPage**

- Setup: Establish the routes in App.jsx & the Link to direct to the route

- Get the list of items from API (axios)

- Call the API when the component is loaded (useEffect)

- Variable change when API loads (useState)

- \*\* Refer to Pokemon API assignment \*\*

**## 4. DetailsPage**

- Setup: Establish the routes in App.jsx & the Link from the dashboard to the details route

- Get id from params (useParams)

- Use the id from params to get data from API when the component loads (axios & useEffect)

- Variable change when API loads (useState)

- \*\* Refer to Pokemon API assignment \*\*

**## 5. CreatePage**

- Setup: Establish the routes in App.jsx & the Link to direct to CreatePage

- Handle form input (useState)

- Handle form submit

- On submission, call the API in backend to post the new item in db (axios)

- Consider the logic after create (If redirecting to a different page, useNavigate?)

**## 6. UpdatePage (Details + Create)**

- Setup: Establish the routes in App.jsx & the Link from the dashboard to the UpdatePage route

- Get id from params (useParams) - Like Details

- Use the id from params to get data from API on load (axios & useEffect) - Like Details

- Handle form input (useState) - Like Create

- update form input state variables after getting the data from API

- On form submission, send the form to API (axios)

- Consider the logic after update (Where to redirect? useNavigate?)

**## 7.1 Delete on Edit/DetailsPage**

- Setup: No routes needed in App.jsx, but need to create the button to perfrom Delete

- Get id from params

- When the button is clicked, send a request to API with the ID to delete that item from db

- Consider the logic after delete - redirect? (useNavigate)

**## 7.2 Delete on Dashboard**

- Setup: No routes needed in App.jsx, but need to create the button in Dashboard

- Get the id in the button inside .map (Refers to TodoList or 5.10 from my demo )

- When the button is clicked, execute handleDelete with the ID, and send API request to delete

- Upon successful API request, update the list (in state) to remove that item (Refers to the platform/ TodoList / 5.10 from my demo)