React Dashboard Project Guide

Overview

You'll build a complete React + TypeScript dashboard app with **React Router**, **React Query (or Redux/Context API)**, and **Tailwind or Sass**.
The app will contain a **Login page** and a **Dashboard** with three feature cards.

---

1. Login Page

- Accept username and password.
- Use any dummy credentials (e.g., `username: admin`, `password: admin123`).
- On success, navigate to the **Dashboard** route using React Router.
- Use state management (Context/Redux) to store the login state.

---

2. Dashboard Overview

After login, the user is routed to `/dashboard`.
The dashboard contains **three cards** — each representing a separate feature.

Card 1: User & Posts Manager

Tasks

1. Fetch users from `https://jsonplaceholder.typicode.com/users` using React Query or Redux thunk.
2. Render a list of users — each clickable.
3. Clicking a user routes to `/users/:id` page:
    - Display user info (name, email, etc.).
    - Two sections:
        - **Posts:** List all posts for that user.

- **To-dos:** List all to-dos for that user, toggle done/undone (change style: green & line-through) preserve to-do done/not-done state through the app life cycle.

Card 2: Note Manager

Tasks

1. Input field + dropdown for priority (`important` , `normal` , `delayed`).
2. Add note button — stores note in state.
3. Three categorized sections showing notes by priority.
4. Ability to:
    - Delete a note.
    - Change note priority (drag/drop or select change).

Card 3: Simple Analytics

Goal: Summarize statistics from User data.

Tasks

1. Show total number of users.
2. Show which user has (username and number of todos/posts):
    - The most posts.
    - The fewest posts.
    - The most completed to-dos.
    - The fewest completed to-dos.
3. Display results in simple styled boxes (no charts).

Card 4: Weather Widget

Goal: Display real-time weather information for any city using a public weather API.

Tasks

1. Create a Weather Card that fetches weather data from the **OpenWeatherMap API** using the following endpoint:
   https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}&units=metric
2. Add an **input field** for the user to enter a city name and a **Search** button to trigger the fetch.
3. Display:

- City name
- Temperature (°C)
- Weather description (e.g., "clear sky")
- Humidity
- Weather icon via the link:
  https://openweathermap.org/img/wn/10d@2x.png
  Docs: https://openweathermap.org/weather-conditions

4. Add **loading** and **error** states:

- "Fetching weather..." during loading
- "City not found" or "Error fetching data" on failure

Bonus

- Detect and display the **current location's weather** using `navigator.geolocation` via this link:
  https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={api_key}

Note Replace each {} in the urls with real data

Bonus Challenges (optional)

- Persist login and notes using `localStorage` .
- Add loader and error states for fetch requests.