Task Title: Build a Dynamic Dashboard with Authentication and API Integration

---

Objective:

The goal is to create a fully functional, dynamic **Dashboard** with:

- **Authentication (Login/Logout)**
- **API Data Fetching & Filtering**
- **Reusable Components**
- **Responsive Layout with Tailwind CSS**

Requirements:

1. Authentication Flow

- Create a **Login Page** with fields for:
  - Email
  - Password
- Add form validation:
  - Email format check
  - Password must be at least 6 characters
- On successful login:
  - Store a **JWT token** (mock one) in `localStorage`.
  - Redirect the user to the **Dashboard**.
- On logout:
  - Clear the token and redirect to the **Login Page**.
- Protect the Dashboard route:
  - If no token is present, redirect to the **Login Page**.

---

2. Dashboard Page

- Create a **Dashboard Layout** with:
  - **Header:** Display user info with a logout button.
  - **Sidebar:** Navigation links (Dashboard, Settings, Profile).
  - **Main Content:**
    - Display a table with **API data** (use `https://jsonplaceholder.typicode.com/posts`).
    - Add a **search and filter** feature (by title or ID).

        - Paginate the results (5 posts per page).
- **Error Handling:**
    - Display an error message if the API call fails.

---

3. API Integration

- Use `getServerSideProps` or `getStaticProps` to **fetch initial data** on server-side.
- Use `useEffect` and `useState` hooks for **client-side filtering and pagination**.

---

4. Styling

- Use **Tailwind CSS** for styling.
- Ensure the dashboard is **responsive** on all devices.
- Add hover effects and transitions for a polished UI.

Bonus (Optional)

- Add a **dark mode toggle**.
- Use **Shadcn/ui** components (e.g., table, button, input).
- Add **loading spinners** while fetching API data.

Tech Stack:

- **Framework:** Next.js (App Router)
- **Styling:** Tailwind CSS
- **Authentication:** Mock JWT Token
- **API:** JSONPlaceholder

Acceptance Criteria:

- **Authentication flow** works correctly with redirection.
- Dashboard displays API data with filtering and pagination.
- Proper error handling and loading states.
- Clean and modular code with reusable components.

Deadline:

- **Estimated Time:** 6-8 hours
- Submit the code via GitHub with a `README.md` including setup and running instructions.
- Vercel for deployment

Tips for the Intern:

- Use `localStorage` for token management.
- Split the code into **reusable components** (e.g., `Header`, `Sidebar`, `Table`).
- Use `map()` efficiently to render dynamic data.
