**Task Overview: Create a responsive, interactive dashboard web application using Next.js, TypeScript, and Tailwind CSS. The dashboard will manage and visualize user data fetched from a provided API endpoint. Complete the entire task within 4 days and deploy the application to a serverless hosting provider such as Vercel.**

**Requirements:**

1. **API Integration:**
   * **Fetch user data from https://jsonplaceholder.typicode.com/users.**
   * **Fetch posts data from https://jsonplaceholder.typicode.com/posts.**
   * **Handle loading, error states, and implement data fetching using Next.js server-side rendering (SSR) and API Routes.**
   * **Integrate fetched data effectively, associating posts with corresponding users based on user ID.**
   * **Implement caching or data prefetching techniques to optimize performance.**
2. **Dashboard UI:**
   * **Create a responsive dashboard layout using Tailwind CSS.**
   * **Dashboard should include:**
     + **Sidebar navigation (collapsible on mobile)**
     + **User data table with sorting and filtering functionality**
     + **Search bar to filter users dynamically by name, email, and username**
3. **Dynamic Routing:**
   * **Implement a detailed user page accessible via dynamic routing (/users/[id]), displaying detailed information of the selected user along with their associated posts.**
4. **Interactive Components:**
   * **Develop reusable components with clear separation of concerns:**
     + **Sidebar, Header, UserTable, SearchBar, Pagination, and PostList**
   * **Implement client-side sorting and pagination for the user data table (10 items per page).**
5. **Form and Validation:**
   * **Create an editable form within the detailed user page allowing user details to be edited and validated using a library of your choice (e.g., React Hook Form, Yup).**
6. **JWT Authentication:**
   * **Implement JWT-based authentication using Next.js middleware.**
   * **Create a login page with mock authentication credentials:**
     + **Username: testuser**
     + **Password: testpass**
   * **Generate JWT tokens upon successful login.**
   * **Restrict dashboard and user detail pages to authenticated users only.**
   * **Implement logout functionality by clearing JWT tokens.**
7. **Advanced TypeScript Usage:**
   * **Utilize strict and clear TypeScript interfaces/types throughout the project.**
   * **Avoid using any; implement proper types for components, props, and API responses.**
8. **Performance Optimization:**
   * **Optimize components for performance (memoization, avoiding unnecessary re-renders).**
9. **Serverless Deployment:**
   * **Deploy the application to a serverless hosting provider such as Vercel.**
10. **Extra Challenge (Optional but Recommended):**
    * **Add a dark mode toggle feature with persistence (Local Storage or Cookie).**

**Evaluation Criteria:**

* **Clean, modular, and maintainable code structure.**
* **Effective use of Next.js features (SSR, dynamic routes, API routes, middleware).**
* **Robust TypeScript integration and accurate typing.**
* **Tailwind CSS implementation for a responsive and attractive UI.**
* **Handling edge cases and errors gracefully.**
* **Secure and functional JWT-based authentication flow.**
* **Code readability and comments/documentation.**

**Deliverables:**

* **GitHub repository link containing the complete codebase.**
* **A brief README.md describing the setup, dependencies, running instructions, and deployment details**