Intern Assignment – Full-stack Practice Task

🎓 Intern Assignment – Full-stack Practice Task

🔍 Overview

This exercise is designed to assess your ability to build a small full-stack web application using modern technologies in both frontend and backend. Your submission will demonstrate your proficiency in React, Tailwind CSS, ShadCN UI, form handling, API integration, and database design.

🧾 Requirements

You are expected to build a simple **authentication flow** and a **basic dashboard page**, based on the following example links:

- **Sign-in/Sign-up Page:**
  👉 https://demo.achromatic.dev/auth/login
- **Dashboard Page:**
  👉 https://demo.achromatic.dev/dashboard/home

These pages are meant for reference in terms of layout, behavior, and minimal styling.

💻 Frontend

Build the frontend using the following technologies:

- **React** (preferably with Vite)
- **Tailwind CSS**
- **ShadCN UI** components
- **React Hook Form** + **Zod** for form validation
- **Redux Toolkit** (for state management)
  ↪️ *Or any other popular state management library you're comfortable with (e.g., Zustand, Jotai, React Context)*
- Optional but nice-to-have: **React Router**, **TanStack Query**

✨ Features to implement:

1. **Sign Up / Sign In Flow**

    - Form validation (e.g. email format, required fields)
    - Proper error handling from backend responses
    - Form UI using ShadCN components
2. **Authenticated Layout**
    - Use a token-based auth approach (e.g., JWT)
    - Protect dashboard routes
    - Show basic user profile info in the navbar/sidebar
3. **Dashboard Home Page**
    - Static layout that mimics the reference UI
    - Optional: fetch user-specific data from backend
    - Use modular components and reusable UI patterns

🛠️ Backend

You can use either **ExpressJS** or **NestJS** (NestJS is preferred) with:

- **PostgreSQL** (preferred) or **MongoDB**
- Any ORM of your choice (e.g., Prisma, Mongoose, TypeORM)

🔒 Auth Flow

- Use JWT for authentication (access + refresh tokens if possible)
- Store users in the database with encrypted passwords (e.g., bcrypt)
- Provide REST API endpoints for:
    - `POST /auth/signup`
    - `POST /auth/login`
    - `GET /me` (authenticated route)

🗃️ Database

You'll need a basic user table/schema with:

- `id`
- `name`
- `email`
- `passwordHash`
- `createdAt`

- `updatedAt`

If you're using PostgreSQL + Prisma, bonus points for using migrations and seed scripts.

✅ Optional Enhancements

- Use TanStack Query for backend data fetching
- Deploy the app to Vercel / Render / Railway / Supabase
- Add basic unit tests or integration tests

📦 Submission

- GitHub repository with clear commit history
- README.md that includes:
  - Setup instructions
  - Assumptions or trade-offs
  - Screenshots or demo link (if deployed)
- Bonus: Add `data-testid` attributes for key buttons (e.g., `create-btn`, `login-btn`) to show test-readiness.

🙋‍♂️ Questions?

If any requirement is unclear or if you want to propose alternatives (e.g., using Firebase auth or Next.js), feel free to ask during the assignment.