Technical Assessment (Intern - Fullstack Developer)

Objective: Using the provided Figma file as your design reference, create a simple React application that highlights your full-stack skills. Your stack should include React, TailwindCSS, and optional Shadcn for UI, Supabase for your database, Prisma for ORM, and Redux or Zustand for state management.

Figma file link:
https://www.figma.com/design/2raZwGOfHhhXvhsuoJyg4P/Full-Stack-Developer-Technical-Assessment?node-id=1-19&t=dPNaunn8dVX2Niu2-1

Part 1: UI and State Management

A. Design & Layout

- **Figma Reference:** Re-create the primary screen from the Figma file as closely as possible, including layout, color schemes, and typography.
- **Component Structure:**
    - **Header & Navigation**: Implement a responsive header with basic navigation.
    - **Cards/Sections**: Build cards or sections as indicated in the Figma, focusing on reusability.
- **Styling:**
    - **TailwindCSS**: Use utility classes to achieve the desired spacing, fonts, and color scheme.
    - **Shadcn (Optional)**: Integrate Shadcn to add depth (shadows, subtle UI effects) for extra polish.

B. State Management

- **Choose Redux or Zustand:**
    - Set up a global state to store and manage data from your Supabase database (e.g., items, user profiles).
    - **Redux**: Create a store, slices/actions, and dispatch CRUD operations to your reducer.
    - **Zustand**: Create a global store with actions for retrieving and updating data.
- **Data Flow:**

  - **Fetching Data**: On mount or user interaction, fetch items/users from your API or directly from Supabase.
  - **Updating Data**: When users create or modify entries, update the global state and persist changes to the database.

Part 2: Database and ORM

A. Supabase Setup

- **Project Creation:** Create a new project on Supabase.
- **Database Configuration:** Define tables/models (e.g. users, posts or items) relevant to your Figma screen.
- **API Keys and Environment:**
  - Retrieve your Supabase project URL and API keys
  - Store then in a .env file

B. Prisma ORM

- Create a simple Node/Next.js API route or server function to handle create, read, update, and delete operations via Prisma.
- Integrate these endpoints/actions with your state management in Part 1.

Deployment and Submission

- **GitHub Repository:** Push your full project (frontend, Prisma config, .env template) to a public GitHub repository.
- **Live Deployment:** Deploy to Vercel or Netlify. Ensure your environment variables (Supabase URL, API keys) are set in the hosting platform.
- **Deliverables:**
  - Repo Link: Public GitHub URL.
  - Live Demo URL: A publicly accessible link to your deployed application.

Assessment Criteria

- **Functionality:**
  - Does the UI mirror the Figma design?
  - Are CRUD operations fully functional (create, read, update, delete)?

- **Code Quality:**
  - Is your React structure logical and maintainable?
  - Are your Redux/Zustand implementations or custom hooks readable and well-documented?
- **Design & Responsiveness:**
  - Does your application adapt to different screen sizes cleanly?
  - Are TailwindCSS and Shadcn (if used) properly implemented and consistent?
- **Deployment:**
  - Is your project publicly accessible and connected to Supabase?