■ Full-Stack Interview Assignment: Task Manager App

Overview: Build a Task Manager Web App where users can sign up, log in, and manage tasks (CRUD). Use Next.js for frontend/backend API, Supabase Auth for authentication, and Supabase Database for storage.

Requirements

1. Authentication (Supabase Auth)

- Users must be able to sign up, log in, and log out. - Only authenticated users can access the task management page. - Use Supabase's built-in email/password authentication. ■ Evaluation: Supabase Auth API usage, session handling, protected routes.

2. Database (Supabase)

Create a `tasks` table: - id (UUID, PK) - user_id (UUID, FK → users) - title (text) - description (text) - status (enum: "todo" | "in-progress" | "done") - created_at (timestamp) Each user can only see their own tasks. ■ Evaluation: Schema design, RLS usage, Supabase queries.

3. Task CRUD (Next.js API + React UI)

- List tasks for the logged-in user. - Create a task (title, description, status). - Edit task fields. - Delete a task. - Filter tasks by status. ■ Evaluation: SSR/CSR usage, API routes, secure CRUD.

4. UI/UX (React + Next.js)

- Use Next.js Pages or App Router. - Clean, simple UI (TailwindCSS optional). - Display loading and error states. ■ Evaluation: React practices, component structure, UX.

■ Bonus (Optional)

- Add task search or pagination. - Add "Mark all done" bulk action. - Deploy to Vercel + Supabase and share live URL.

■ Submission Guidelines

- Submit a GitHub repository with the code. - Include README with setup, usage, and (optional) demo link.

■ Evaluation Criteria

Skill	What We Check
React.js	Component structure, hooks, state management
Next.js	Routing, API routes, SSR/CSR usage
Supabase Auth	Auth flow, session handling

Supabase DB	Schema design, CRUD, RLS usage
Code Quality	Clean structure, reusable components
UX	Validation, usability, loading/error states