NotesApp T3 Stack
Application Architecture Overview

1) Database – `prisma/schema.prisma`

This file defines the structure of my database: the tables, fields, and relationships (like `User`, `Note`, `Post`). Prisma uses it to generate a type-safe client so my app can query the database with clean, predictable code.

2) Authentication

- `server/auth/config.ts` → Configures NextAuth with providers (like Discord) and Prisma as the adapter, ensuring user accounts and sessions are stored in the database.
- `server/auth/index.ts` → Initializes NextAuth with my config and exports helpers (`auth`, `signIn`, `signOut`) for use across the app.
- `src/app/api/auth/[...nextauth]/route.ts` → Exposes the NextAuth handlers to Next.js, so the browser can call `/api/auth/*` endpoints for login, logout, and session management.

3) tRPC Backend

- `server/api/routers/notes.ts` → Defines all the backend logic for notes (create, read, update, delete), with input validation and session checks.
- `src/app/api/trpc/[trpc]/route.ts` → Connects my `appRouter` to Next.js, exposing the tRPC API at `/api/trpc` so the frontend can call these procedures.

4) Frontend

- `src/app/notes/page.tsx` → The notes feature page. It uses tRPC hooks to fetch, create, edit, and delete notes, and shows different UI depending on whether the user is logged in.
- `src/app/layout.tsx` → The global wrapper for your app. It applies styles, fonts, and sets up providers (`SessionProvider` for auth and `TRPCReactProvider` for API access) so every page has authentication and tRPC available.