CLAUDE.md

This file provides guidance to Claude Code (claude.ai/code) when working with code in this repository.

Repository Layout

The actual Next.js application lives in `vishlabs-client/`. All commands below should be run from that directory.

```
Personal-Dashoard/
└── vishlabs-client/    ← Next.js app (work here)
    ├── src/
    │   ├── app/        ← App Router pages
    │   ├── components/ ← Shared + page-specific components
    │   ├── data/       ← Static data files (projects, experience, etc.)
    │   ├── lib/        ← Prisma singleton
    │   ├── styles/     ← globals.scss
    │   └── types/      ← next-auth session type augmentation
    ├── prisma/
    │   └── schema.prisma
    ├── src/auth.ts      ← NextAuth config
    └── src/middleware.ts ← Edge route guard
```

Commands

```
# From vishlabs-client/
npm run dev       # Start dev server
npm run build     # Production build
npm run lint      # ESLint

# Database
npx prisma studio       # GUI for editing DB records (e.g., promoting users to
ADMIN)
npx prisma db push      # Push schema changes to Supabase
npx prisma generate     # Regenerate Prisma client (runs automatically via
postinstall)
```

Required Environment Variables

Both `.env` (CLI/Prisma) and `.env.local` (Next.js runtime) must be present in `vishlabs-client/`:

```
GOOGLE_CLIENT_ID=
GOOGLE_CLIENT_SECRET=
AUTH_SECRET=
```

```
DATABASE_URL=        # Supabase pooler, port 6543, append ?pgbouncer=true
DIRECT_URL=          # Supabase direct, port 5432 (used by Prisma migrations)
```

Architecture

Auth Flow

- **Provider**: Google OAuth only. `src/auth.ts` exports `{ handlers, signIn, signOut, auth }`.
- **Strategy**: JWT (required for edge middleware — Prisma can't run in the edge runtime).
- **Session augmentation**: `user.id` and `user.role` are injected into the JWT in the `jwt` callback and exposed via `session` callback. Types declared in `src/types/next-auth.d.ts`.
- **AuthModal**: Triggered via `useAuthModal()` context (from `AuthProvider`). `AuthProvider` wraps the entire app in `layout.tsx` and also provides `SessionProvider`.
- **Auth logging**: Every sign-in writes to the `AuthLog` table (non-fatal; won't block login on failure).

Route Protection

`src/middleware.ts` runs at the edge and reads the JWT role:

- `/admin`, `/anu` — ADMIN only; redirects to `/` if not ADMIN.
- `/projects` — any authenticated user; redirects to `/` if not authenticated.
- Pages also do a server-side `auth()` check as a second layer (see `app/admin/page.tsx`).

Page → Component Pattern

Each route in `src/app/` is a thin page file that composes section components from `src/components/`. Data is sourced from static files in `src/data/` rather than fetched at runtime (except admin and DB-backed pages).

Visual / UI Conventions

- **`Beams`**: Three.js WebGL animated beam background used on `/` and `/admin`. It uses a custom GLSL shader via `extendMaterial`.
- **`SmoothScrolling`**: Wraps the layout with `ReactLenis` for smooth scroll.
- **SCSS modules**: Every component has a co-located `.module.scss`. Global styles in `src/styles/globals.scss`. SASS `includePaths` is set to `./src/styles`.
- **Styling**: Black background (`#000`), white text, accent `rgb(33, 119, 109)` / `rgb(80, 220, 200)`.
- **Font**: "Jersey 10" (Google Fonts).
- Remote images: only `*.googleusercontent.com`, `avatars.githubusercontent.com`, and `github.com` are whitelisted in `next.config.mjs`.

Database Schema (Prisma)

Standard NextAuth tables (`User`, `Account`, `Session`, `VerificationToken`, `Authenticator`) plus:

- `User.role` — `String`, default `"USER"`. To promote to ADMIN: `npx prisma studio`.
- `AuthLog` — records every sign-in with provider and email in a JSON `metadata` field.

Prisma Client

Singleton at `src/lib/prisma.ts`. The `postinstall` script runs `prisma generate` automatically (needed for Vercel deployments).