Assignment 1 Self-Evaluation (Authentication, Authorization, Layout & Dashboard)

Context

- Frontend: React + Vite + TypeScript with Tailwind/shadcn in `frontend/src`. Routing guarded via `ProtectedRoute` / `PublicRoute`, session hydration via `SessionSync` / `useSession`, and persisted access token store in `frontend/src/stores/auth.ts`.
- Backend: NestJS + TypeORM/Postgres (`backend/src`). Global `MyAuthGuard` defaults to bearer JWT; refresh token is HttpOnly cookie via `setRefreshCookie`. Dashboard endpoints are role-protected.
- Design mockups: `/design/admin-dashboard.excalidraw` and `/design/user-dashboard.excalidraw`.

Requirement-by-requirement check

- **Authentication — Met**
  - Email/password signup & signin with DTO validation and bcrypt hashing (`backend/src/modules/auth/auth.controller.ts`, `backend/src/modules/auth/providers/sign-in.provider.ts`, `backend/src/modules/users/providers/users.repository.ts`).
  - Google login wired on client and verified server-side (`frontend/src/components/auth/*`, `backend/src/modules/auth/social/google-authentication.service.ts`).
  - Token model: AT in persisted Zustand store; RT in HttpOnly cookie with rotation on `/auth/refresh` and Axios retry (`frontend/src/lib/http.ts`, `backend/src/modules/auth/auth.controller.ts`). Session hydration via `SessionSync` / `useSession`.
  - Private routes redirect unauthenticated users to `/signin` (`frontend/src/routes/ProtectedRoute.tsx`); signed-in users blocked from auth pages via `PublicRoute`.
- **Authorization — Partial**
  - Role field on users and guard infrastructure present (`backend/src/modules/auth/guard/role-based.guard.ts`, `backend/src/modules/auth/providers/access-control.provider.ts`).
  - Dashboard routes now enforce roles: `/dashboard/admin` requires `ADMIN`; `/dashboard/user` allows `USER|ADMIN` (`backend/src/modules/dashboard/dashboard.controller.ts`).
  - UI switches dashboards/sidebar based on `me.role` (`frontend/src/pages/home/AdminDashboard.tsx`, `frontend/src/pages/home/UserDashboard.tsx`, `frontend/src/components/app-sidebar.tsx`); client-side enforcement is limited to view selection/403 handling—no admin-only actions yet.
- **Layout & Design System — Met**
  - Theme tokens for light/dark, spacing, gradients (`frontend/src/index.css`, `frontend/tailwind.config.ts`), persisted theme toggle (`frontend/src/providers/ThemeProvider.tsx`, `frontend/src/components/theme/ThemeToggle.tsx`).
  - Reusable layout primitives (`MainLayout`, grid/stack/section) and shadcn UI kit (`frontend/src/components/ui/*`, `frontend/src/components/ui/field.tsx`).
- **Dashboard UI & Implementation — Partial**
  - Admin/User dashboards with summary cards, trend chart, tables/lists, navigation sidebar (`frontend/src/pages/home/*`), fetching via React Query from `/dashboard/*`.
  - Backend serves DB-backed data when tables exist with safe fallbacks (`backend/src/modules/dashboard/dashboard.service.ts`); user endpoint filters by active user id. Frontend handles 401/403 with friendly states.
  - No admin-only action buttons or privileged mutations.
- **Developer Tooling — Partial**
  - ESLint configured frontend/backend; backend has Prettier config. Tests: frontend Vitest for `ProtectedRoute`, backend e2e covers dashboard auth 401.
  - Prettier/lint-staged/Husky not wired; broader test coverage pending.
- **Deployment & Documentation — Missing**
  - Local/dev setup documented; no live deployment URL or CI checks yet. `NEXT_STEPS.md` present.

Overall assessment

- Auth flows (email/password + Google) with AT/RT rotation and guarded routing are working. Dashboards call role-protected APIs and handle 401/403 with fallbacks to mock data.
- Authorization depth is limited to dashboard views; admin-only actions and broader `@Roles` coverage are not implemented. Tooling (Prettier/Husky/test breadth) and deployment remain gaps.

Next steps to reach rubric parity

1. Deploy FE/BE (e.g., Vercel/Netlify + Railway/Render) and document live URLs, cookie domain/env tweaks, and how authorization is enforced in prod.