

Implementation

1. Backend Setup (.NET 8.0 Web API)

- ! Initialize new .NET 8.0 Web API project.
- ! Install necessary NuGet packages:
 - ! `Microsoft.AspNetCore.Authentication.JwtBearer`
 - ! `Microsoft.AspNetCore.Identity`
 - ! `Microsoft.EntityFrameworkCore`
 - ! `Npgsql.EntityFrameworkCore.PostgreSQL`
 - ! `AspNet.Security.OAuth.Providers` (Google, GitHub)
- ! Configure `Startup.cs`:
- ! Add authentication/authorization middleware.
- ! Add EF Core with PostgreSQL provider.
- ! Configure JWT and OAuth settings via `appsettings.json`.

2. Database Migration

- ! Create EF Core models and relationships as per the schema.
- ! Add `DbContext`, apply migrations, seed `roles` table (`Admin`, `User`).

3. JWT Token Service

- ! Generate `access_token` with 15-min expiry.
- ! Generate `refresh_token` with 7-day expiry and persist to DB.
- ! Include role claims in JWT.
- ! Build middleware to extract JWT from headers and validate.

4. OAuth Integration

- ! Setup Google and GitHub OAuth configuration.
- ! Build endpoint `/auth/oauth/callback` to handle token exchange.
- ! On first login, create a user record and assign `User` role by default.

5. Authorization Middleware

- ! Build custom attributes (e.g., `[Authorize(Roles = "Admin")]`) for role checks.
- ! Ensure all admin-level CRUD APIs are protected.

6. Frontend Setup (React + Bootstrap)

- ! Set up routing using React Router.
- ! Install Bootstrap for styling.
- ! Build:
 - ! `LoginPage.jsx` ð Login form + OAuth buttons
 - ! `SignupPage.jsx` ð Registration form

- ! `UsersPage.jsx` ⚡ Table view; conditionally render action buttons based on role
- ! `Navbar.jsx` ⚡ Show login/logout and current user
- ! Use Axios for API requests and store JWTs in memory (or HttpOnly cookie for refresh token).

7. Refresh Token Handling

- ! When API returns 401, auto-refresh token via `/auth/refresh-token`.
- ! If refresh fails, force logout and redirect to login.

8. Admin Functions

- ! Add forms/modals in `UsersPage` for create/edit/delete.
- ! Connect form actions to secure backend endpoints.

9. Security Hardening

- ! Use HTTPS only.
- ! Rate-limit login attempts.
- ! Validate OAuth token responses from providers.