SPEC-001: Scalable Authentication and Authorization System

Table of Contents

1. Background

This design proposes a scalable and extensible user authentication and authorization system using React (with Bootstrap for UI), .NET 8.0 Web API for the backend, and PostgreSQL for data persistence. The goal is to provide secure access control for a web application supporting both end-users and administrators.

The system must support both traditional username/password-based authentication as well as third-party logins via OAuth (Google, GitHub). The architecture should allow for future SSO provider integrations with minimal changes. Role-based access control will distinguish between admin users, who can perform all CRUD operations on user records, and regular users, who can only view user data.

2. Requirements

2.1. Must Have

- Support for user registration and login via email/password.
- Support for third-party authentication (Google, GitHub) via OAuth 2.0.
- Role-based access control:
- Admins: full CRUD access to user data.
- Users: read-only access.
- JWT-based stateless authentication.
- Secure password hashing and salting (e.g., using bcrypt).
- PostgreSQL database for user and role data.
- RESTful API with secure endpoint protection.

- Responsive React + Bootstrap UI for login, signup, and user management.
- Logging and basic monitoring (for audit and debugging).

2.2. Should Have

- Extensible architecture to easily plug in additional SSO providers.
- Token refresh mechanism using refresh tokens.
- Email verification during signup.

2.3. Could Have

- Account locking after multiple failed login attempts.
- Admin dashboard with user activity logs.

2.4. Won’t Have (for MVP)

- Multi-tenancy.
- Fine-grained permissions beyond Admin/User roles.