**Project Documentation (EG-Group-WebApp)**

**Design Decisions**

* **Separation of Concerns:**  
  The project is divided into a **Next.js frontend** and a **.NET Core Web API backend**, enabling independent development and scaling of client and server logic.
* **Token-Based Authentication:**  
  JWT is used for authenticating users. Tokens are stored in local Storage on the frontend to persist sessions across refreshes.
* **User Roles:**  
  Each user has a Role property (Admin, Manager, User etc.), allowing role-based access control in both UI and API.
* **Activity Logging:**  
  Implemented using *ILogger<T>* in the backend for logging key actions like login attempts, API requests, and exceptions.
* **Modular Database Initialization:**  
  The database is initialized using a SQL script (*init.sql*) to simplify onboarding and local setup.
* **.env Configuration in Frontend:**  
  API base URL is injected via *.env.local* using *NEXT\_PUBLIC\_API\_BASE\_URL*, avoiding hardcoded URLs in code.

**Technologies Used**

**Front-End (Next.js)**

* **React (Next.js App Router)** for modern server/client hybrid rendering
* **Tailwind CSS** for UI styling
* **React Toastify** for non-blocking toast notifications
* **Environment Variables (.env.local)** for flexible API integration

**Back-End (ASP.NET Core)**

* **.NET 8 Web API** for RESTful endpoints
* **C# with Entity Framework Core (optional)** for ORM-based database access
* **ILogger<T>** for structured logging
* **JWT Authentication** for secure, stateless login
* **SQL Server** for relational database management

**Key Features**

* ✅ User Registration and Login
* ✅ Role-based User Access
* ✅ Secure API Integration via JWT
* ✅ Activity and Error Logging
* ✅ Toast notifications for better UX
* ✅ .env.local based config for flexibility
* ✅ Manual SQL script (init.sql) to initialize database

**Challenges Faced**

* **Submodule Misconfiguration:**  
  Git initially treated the front-end folder as a Git submodule, preventing it from opening in the GitHub UI. This was resolved by removing the submodule reference and re-adding the folder properly.
* **Frontend Feedback for Login:**  
  Originally used alert() for failed logins. Improved by integrating *react-toastify* for cleaner and non-blocking user feedback.
* **CORS Issues:**  
  Faced typical CORS errors during API calls. Solved by enabling CORS in *Startup.cs* and allowing local frontend origins.
* **Consistent API URLs:**  
  Fixed inconsistent API base URL issues by introducing .env.local support in the frontend.
* **Password Hashing:**  
  In early development, passwords were stored in plaintext (for demo/testing). Production-ready implementation would use hashing like BCrypt.