**Architecture Overview**

* **Presentation Layer (SalesRep.UI)**
  + Uses **Razor Pages** and some MVC controllers for web UI.
  + Handles user interaction, input validation, and view rendering.
* **API Layer (SalesRep.API)**
  + Exposes RESTful endpoints for integration or SPA/mobile clients.
  + Handles authentication, data access, and business logic exposure.
* **Business Logic Layer (SalesRep.Infrastructure)**
  + Contains services (e.g., SalesRepService, SaleService) implementing business rules.
  + Uses dependency injection for testability and separation of concerns.
* **Data Access Layer**
  + Uses **Entity Framework Core** (AppDbContext) for ORM and database access.
  + Models and interfaces are defined in **SalesRep.Core** for reusability and decoupling.

**Reasoning Behind Design Choices**

* **Razor Pages**: Chosen for its page-centric model, which simplifies CRUD and admin-style UIs compared to MVC.
* **Layered Architecture**: Promotes separation of concerns, making the codebase easier to maintain, test, and extend.
* **Dependency Injection**: Standard in ASP.NET Core, enabling loose coupling and easier unit testing.
* **Entity Framework Core**: Provides a modern, efficient ORM for .NET, supporting migrations and LINQ queries.
* **API Layer**: Allows for future expansion (e.g., mobile apps, third-party integrations) without changing the core logic.

**Testing Methodologies**

* **Unit Testing**
  + Business logic and services are tested in isolation using xUnit.
  + Mocking (e.g., Moq) is used for dependencies.
* **Manual UI Testing**
  + Razor Pages and controllers are tested in-browser for user flows and validation.
* **Validation**
  + Data annotations and model validation ensure data integrity at the UI and API levels.
* **API Testing**
  + This API uses JWT Bearer Token for authentication. Obtain a token from api/AuthAPI/login.
  + Include the token in the Authorization header of your API requests
  + JWT username-admin, password - password

**Notes**

Connection string is configured in appsettings.json under "DefaultConnection" in SalesRep.UI and SalesRep.API. You can add your PostgreSQL database details here.

Then apply migration using below scripts

dotnet ef migrations add InitMigration

dotnet ef database update