This project is a Sales Representative Management System built using ASP.NET Core Razor Pages. It provides features for managing sales representatives, sales data, and filtering/reporting, with a clean separation of concerns and a focus on maintainability and testability.

* Create, read, filter sales entries
* Sales filtering by:
* Sales Representative
* Product
* Date Range
* Region
* Sales Performance (min/max)
* Validations (client & server side)
* Clean Architecture with separate layers:
* Core (domain models and interfaces)
* Infrastructure (data access using Entity Framework Core)
* Application/Services
* UI (ASP.NET Core MVC)

## Tech Stack

* .NET 8 / ASP.NET Core MVC
* Entity Framework Core 8
* PostgreSQL
* pgAdmin 4 (UI for PostgreSQL)
* Bootstrap 5 (for UI)
* Clean Architecture Pattern

Setup Instructions

* Clone the repository
* Update the connection string in appsettings.json of SalesRep.UI project and SalesRep.API project and SalesRep.Infrastructure - AppDbContextFactory
* Run EF Core Migrations
* Run the application

2. Design Choices

- Razor Pages

Chosen for its page-focused approach, which simplifies UI logic for CRUD operations and fits well for admin-style applications.

- Separation of Concerns

- Core: Defines contracts and models, no dependencies on infrastructure or UI.

- Infrastructure: Handles data persistence and business logic.

- UI: Handles HTTP requests, user input, and rendering.

- Dependency Injection

All services and repositories are injected, promoting testability and loose coupling.

- DTOs & ViewModels

Used to decouple domain models from UI and to shape data for specific views.

3. Data Access

- Entity Framework Core

Used for ORM and database migrations.

- DbContext

Centralized in `AppDbContext`, registered in DI.