Courses Management POC

# Description

A simple application that manages Courses CRUD operations. However, the main purpose of this POC is to show the code, technique and solution structure. It is not meant to be a complete solution. Also, some non-functional requirements in any normal application was **not implemented** for the sake of **time**, such as Security, logging, unit testing, and Swagger documentation of the API.

# Technologies used

1. Front End 🡪 MVC
2. Back End 🡪 Web API
3. Database 🡪 SQL Server
4. ORM 🡪 Entity Framework (Code First)

# Design Patterns Used

1. Repository

A generic repository is used to act as a container for the data.

1. Unit of Work

A Unit of Work is used to handle the transactions, and saving changes to the database

1. Adaptor

Adaptor pattern was used in the mappers to map the Domain models, to the DTOs (Data Transfer Objects), and vise verse.

# IOC Container

Simple Injector was used as the IOC container, to handle the dependency injections.

# Solution Structure

Consists of the following solution folders as numbered inside the solution:

1. Shared 🡪 Should Contains any shared components across all layers.
2. Data 🡪 Should contain all the data access layer components
   1. SmartTraining.Data.Domain 🡪 Contains the domain models
   2. SmartTraining.Data.EntityFramework 🡪 Contains the DataContext, and Code First Migrations
   3. SmartTraining.Data.Interfaces 🡪 Contains the IRepository, and IUnitOfWork abstractions.
   4. SmartTraining.Data.Repository 🡪 Contains the concrete classes of both the Repository, and Unit of Work.
3. Business
   1. SmartTraining.Business.Handlers 🡪 The business goes here, Also, this class orchestrates between the Repository, and Unit of Work, and it is the layer that controller interacts with.
   2. SmartTraining.Business.DTO 🡪 The DTO models, also named as View Models that will be sent to the front end to be rendered.
   3. SmartTraining.Business.Mappers 🡪 The mappers between the Domain models (That comes from the data source) and the DTO models.
4. Presentation
   1. SmartTraining.API 🡪 The main API of the application, and also for simplicity, I put the front end views in the same project.

# How it works

1. Database Creation
   1. Open solution in Visual Studio.
   2. Open (View - Other Windows - Package Manager Console).
   3. Make sure that the project “SmartTraining.Data.EntityFramework” is selected in the Default project list.
   4. Run the following command ( update-database )
   5. When finished, the database should be locally created.
   6. Run the application by clicking F5