**Project Description**

The implemented solution is a simple online Contacts App with a web interface that can be used in all popular web browsers. The application store necessary data in a database and it is built using the following technologies/frameworks:

ASP.NET Core 3.1

.NET MVC 5.2.7 with Entity framework Core 3.1

SQL Server 2018

HTML, CSS

The solution can be found in the following github account:

<https://github.com/kdiakatos/Contacts>

**Solution Explanation**

**Architecture**

The technique used for designing the solution is the 3-Layer architecture. A short description about the three layers follows:

1. Data Access Layer

It is a class library project that implements this layer, which is responsible for storing and retrieving data to a persistent database.

1. Business Layer

It is a class library project that implements this layer, which coordinates the application and it is the “middle man” between the data access layer and the presentation layer.

1. Presentation Layer

It is a web application project that serves the user interface of the application.

In more details, the data access layer implements the repository pattern for handling all the transaction with a SQL Server database.

The business layer contains the business models and also the classes which coordinate the flow between the two other layers. Due to the complexity of the project, this layer is responsible for mapping (using Automapper library) business models to Entities and vice versa and furthermore, it implements the validation of the model. Therefore, the business models (instead of the Entities) are exposed to the Presentation Layer.

The presentation layer contains the UI of the application and it contains the Views.

These three (3) layers are loosely coupled, meaning the data layer communicates with business layer and the latter communicates with the presentation layer. Therefore, there is no direct connection between the data and the presentation layer. This is achieved by using the technique of dependency of injection, which is implemented with the use of Unity library. This technique was chosen because it offers more scalability and maintainability to the application.

**Functionality**

The application starts with the main page of the application, where the user can manage his/her contacts. In more detail, the user can add, removes and edit contacts.

**Remarks/Issues**

I decided to create the database on my local SQL Server 2018. For setting the database with Entity Framework, I followed the Code-First workflow.

Therefore, I followed the steps below:

* Create the entities in the Data Access Layer project
* Create the ContactContext class
* Add the connection string in the appsettings.json file
* From the package manager console, I changed the default project to DataAccessLayer and typed the following commands.  
  Enable-Migrations -> Add-Migration <<name>> -> Update-Database