**This is Healthcare API contains following functionalities:-**

**Contact management System:**

This application maintain contact details like First Name, Last Name, Address ,Email, Phone Number and status of contact.

**Application Functionality:**

- List contacts

- Add a contact

- Edit contact

- Delete

**Contact model fields:**

- First Name

- Last Name

- Address

- Email

- Phone Number

- Status (Values: Active/Inactive)

**Prerequisites:**

- Application required Visual Studio 2015 and .Net Framework 4.5.2.

- IIS to host the Rest API

- SQL server

**Installing:**

**Setup Database in Microsoft SQL**

1. Provided the sql scripts to create the database in Sql Server. Running the scripts will create database and tables. I have given my database name as "HealthCare". In this project we will only be working with "Contacts" table to perform CURD operations using Web API and Entity framework.

**Build the Web API application**

1. Clone the application from GIT repository and after it is successful, build the application. Build the application "**HealthCare**” which is API project, so that it gets compiled successfully.

**Host the Web API application**

1. Run the application "**HealthCare**" or it can be hosted in IIS.

**Build the Client application**

1. Build the application "**HealthCareClient**" so that it gets compiled successfully.

**Run the Client application**

1. Copy the URL of **" HealthCare " (web API)** and replace the URL present in **Web.Config** of

"**HealthCareClient**" **(client project)** with key name "ContactAPI" in "appSettings" with the URL of Host Application

1. Run the client application " **HealthCareClient** "

**Application architectural and implementation:**

**Web API**

Evolvable web API Application contain four projects which currently provides **contact management services**:

1. **HealthCare**: Restful API server which handles request from client of any form client side (browser)/server request. Used **MEF** (Managed Extensibility Framework) to resolve dependency of dependencies exist in all layers. (Working on Alternative approach of solving dependency of dependencies without using MEF which is **Unit.Extentions** since MEF introduced little complexity and contains code DLL operations which is unappropriate)
2. **Entities**: A Separate class library which contains business level model which can be used throughout the applications.
3. **BusinessLogic**: Rest API will call this layer which is done by **dependency injection / IOC (Unity)** this layer will perform all business logic.
4. **DataAccess**: Business Logic layer will call data access layerdone by **dependency injection / IOC (Unity)**
   1. since this layer is used to do all persistence level operations such as accessing database we have used Entity Framework with Unit of work handling transactions and repositories
   2. So EF with the help of unit of work pattern helping Repository pattern which is very helpful if in future we need to change persistence layer as well unit of work will take responsibility of managing repositories and transactions. Repositories will also help in keeping common code together and will return iEnumerable unlike EF which returns iqueryable and we can accept parameters through func and expression tree.
5. **GenericResolver** : Is generic resolver is a single point of contact to resolve dependancies of other components / class libraries i.e **Web API-** > **BusinessLogic -> DataAccess** this uses **unity** dll

**Note** : This all the design patterns are implemented while keeping in mind the web world is evolving so client framework , database/ persistence mechanism ,business logic can change and evolve independently also this patterns help Unit Test Projects which can liberate itself with **Nunit** and **Moq** with help of **Automapper**.

**Future Enhancements**:

1. Global error handling and tracing in Database.
2. Owin authentication (Tokens JWT , bearer)
3. Unit Test Project for all layers BL/DL/Resolver/API

**HealthCareClient:**

Client is built on mvc pattern in .net framework and provides UI to manage **contacts**. Because of the time constraints this client is application is not following Dependency injections principle but we are working on the same to accomplish. Since this can also be implemented in other form of technologies much of the attention is paid on Evolvable web API.

**Future Enhancements**:

1. Angular client which will call the same Rest service.