**Project Description:**

Design and implement a production ready application for maintaining contact information.

**Tools:**

Visual Studio 2015, SQLExpress

Designed a webAPI project for ContactManagent and also created a Contact.html page to consume/check the functionality of the webAPI calls.

I have not used any entity framework/Business layer for connecting to Database just to avoid system generated code. Added a **Contact.cs** class in “Model” and **ContactRepository.cs** class in “Services”, which contains services like **GetAllContacts, AddContact, SaveContact** and **DeleteContact**. This file contains all ADO.net code for connecting to database and retrieving values.

Database – All queries are there in SQLQueries.txt under folder “test\Files\”.

|  |  |
| --- | --- |
| Database | ContactInfoDB |
| Table (in this id column is identity column) | tblContactInfo |
| Stored Procedure for getting all contacts | spGetAllContacts |
| Stored Procedure for getting adding contact | spAddContact |
| Stored Procedure for getting saving/updating contact | spSaveContact |
| Stored Procedure for getting deleting contact | spDeleteContact |

**Below are the calls for API**

|  |  |  |
| --- | --- | --- |
| GET | /api/Contacts/ | Gets list of all contacts |
| GET | /api/Contacts/1 | Gets contact with id=1 |
| POST | /api/Contacts/ | Creates/Add a new contact |
| PUT | /api/Contacts/1 | Updates contact with id=1 |
| DELETE | /api/Contacts/1 | Deletes contact with id=1 |

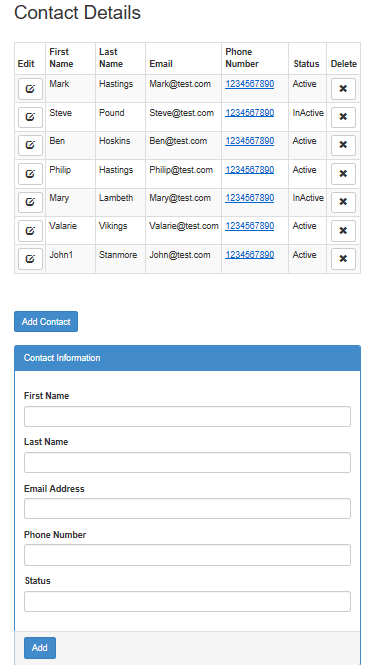
**Additional functionality added**

|  |  |
| --- | --- |
| /api/Contacts?status=all | This is default functionality. By default “/api/Contacts/” works as “api/Contacts?status=all”. Displays/Gets contacts based on status of the contact. In this case it displays contacts with both status. |
| /api/Contacts?status=active | Displays/Gets all contacts which is having status value active. |
| /api/Contacts?status=inactive | Displays/Gets all contacts which is having status value inactive. |
| POST | While adding a new contact, we don’t supply ID field. After adding a new contact, contact id will be generated by the database. We have updated that contact object with newly created id field (in case of datacontext object that does this automatically). After updating this contact object we are also sending the same to the client as location of newly created contact. We can check this in fiddler, as <http://localhost/api/Contacts/17>  Here 17 is the newly added contacts id. |

**Contact.html –** This is basic html page created to test the webAPI functionality.

When user opens this page,

It loads with the existing data in database as shown below.



1. It loads a Page using **GET** api call.
2. For **POST** API call – User can click on **Add Contact** button/Fill the details and click on **Add** button.
3. For **PUT** API call – User can click on pen symbol in **Edit** column, as soon as he/she click on it, Contact information will get loaded, and below **Add** button will converted to **Update** button. User can do necessary changes and save changes.
4. For **DELETE** API call – User can simply Delete using **X** symbol placed in front of record.

**Note**: Validation part is not done in HTML pages as this is designed for basic functionality testing.

test\ContactManager – Contains a project

test\ContactManager\_Publish – Contains Published files

test\Files – Contains README and SQLQueries

Using the published files client can host this solution.

If user has IIS configured he/she can open IIS ->Right Click on sites -> select Add website -> provide all necessary information, browse the physical path for folder where these publish folder files are copied.

<https://github.com/AnoupamaWaggh/Contact-Manager/upload/master>