**Technical Specifications:**

|  |  |  |
| --- | --- | --- |
| **Development Area** | **Tool/Technology** | **Version** |
| Database Design | SQL Server Management Studio | 2017 |
| Api Logic | .Net Framework, Web Api,  Entity Framework 6.0 | 4.7.1 |
| Unit Tests | MSunit |  |

**Please find breakdown of time spent on different development areas:**

|  |  |  |
| --- | --- | --- |
| **Development Area** | **Tasks** | **Time Spent** |
| Data store design | 1.Creation of inventory database  2. Creation of items table. | 0.5hr |
| Api Logic | 1.Project folder structure for Controller, DAL, BLL, BO  2. Add model using entity framework.  3. Write API methods to add, update, delete items and get list of items using exception handling, asynchronous programming, data entity validations. | 1.5hr |
| Unit Test Coverage | 1. Writing unit cases.  2. Testing unit cases | 1hr |

**Please find the steps to run the project:**

1. The project is available under the following master or develop branch.

2. The Git Bash Here command for cloning repository is

git clone <https://github.com/Ramesh9437/ThinkBridgeCollaboration.git>

3. Download code and open the project in visual studio solution.

5. Create database with name **extdbInventorySystem** and create table with name by running the following script.

USE [extdbInventorySystem]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Items] Script Date: 6/19/2021 12:35:52 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Items](

[ItemId] [bigint] IDENTITY(1,1) NOT NULL,

[ItemName] [nvarchar](50) NOT NULL,

[ItemDescription] [nvarchar](250) NULL,

[Price] [decimal](18, 2) NOT NULL,

[IsDeleted] [bit] NOT NULL,

[AddedOn] [datetime] NOT NULL,

CONSTRAINT [PK\_Items] PRIMARY KEY CLUSTERED

(

[ItemId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Items] ADD CONSTRAINT [DF\_Items\_IsDeleted] DEFAULT ((0)) FOR [IsDeleted]

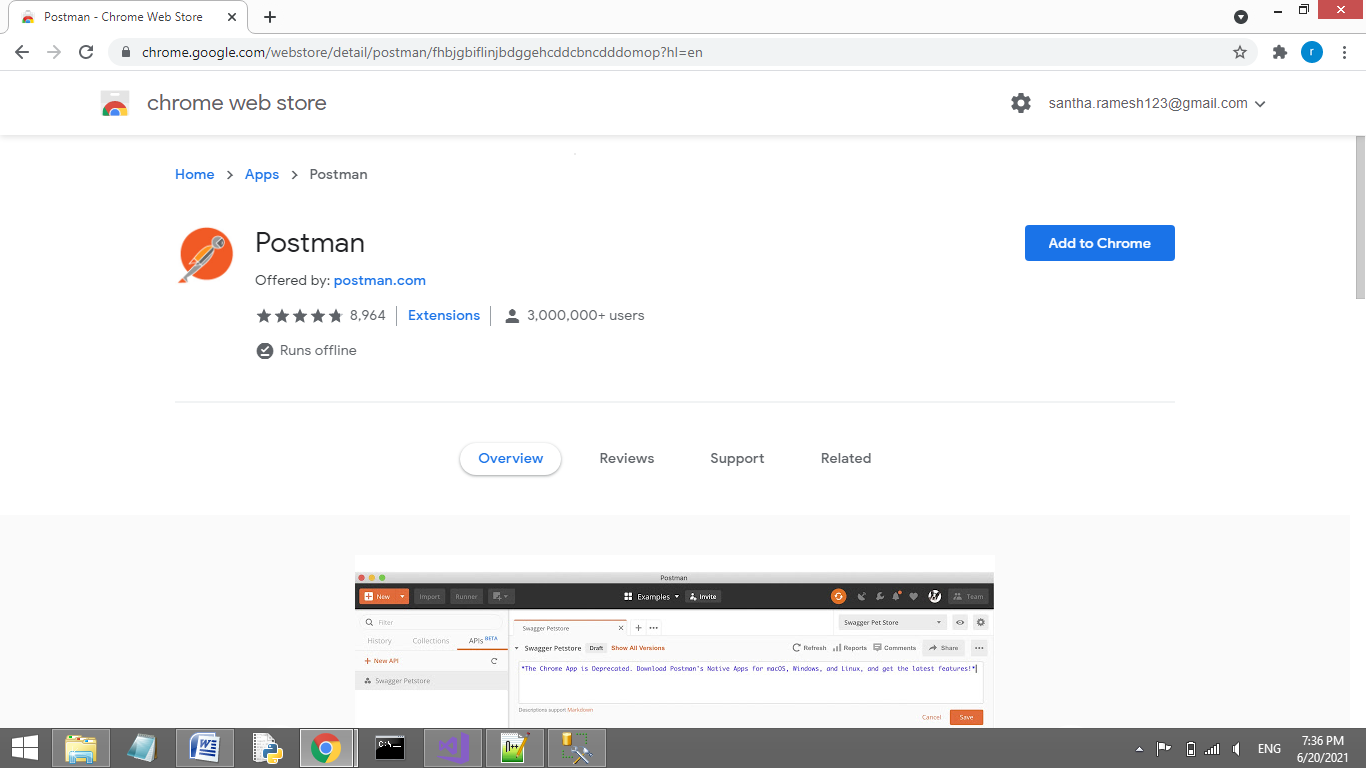
GO

ALTER TABLE [dbo].[Items] ADD CONSTRAINT [DF\_Items\_AddedOn] DEFAULT (getdate()) FOR [AddedOn]

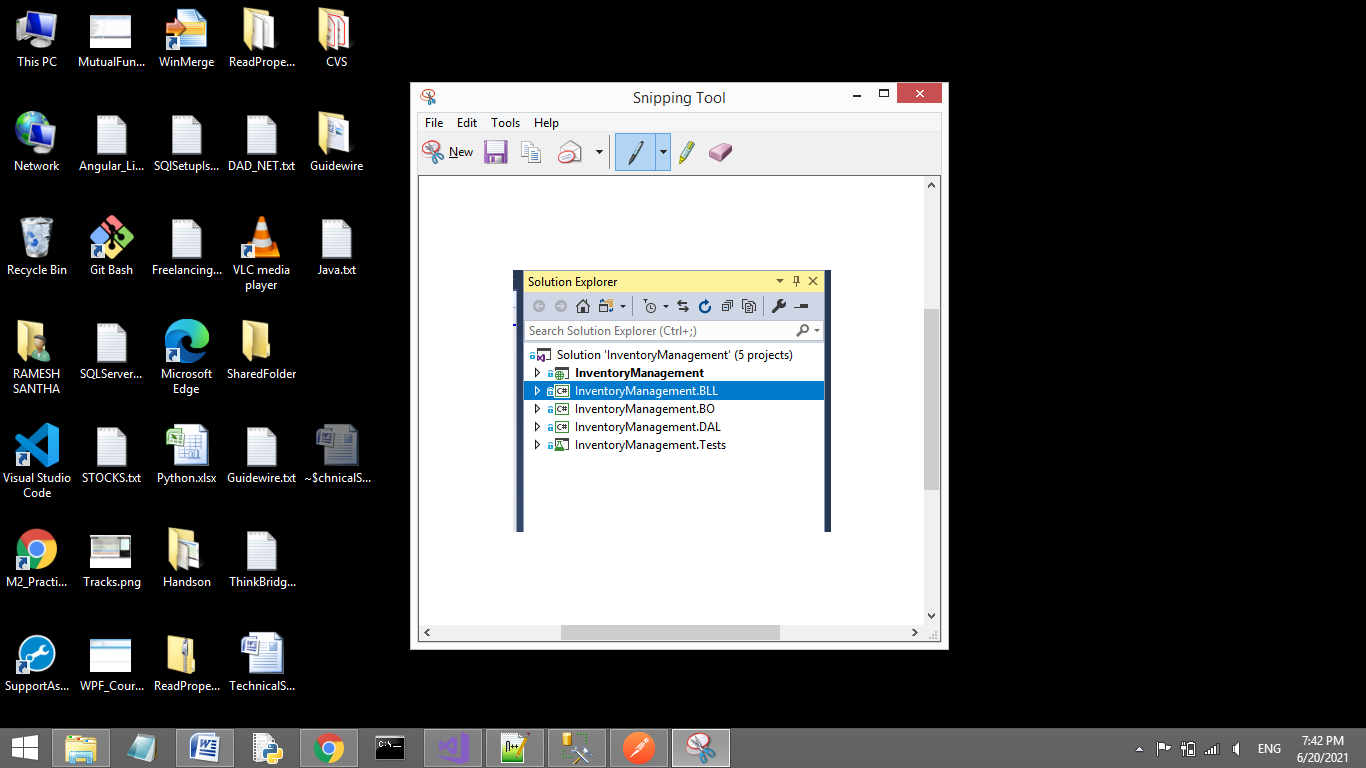
GO

6. After creation of table update the model using Entity Framework.

7. Add extension for postman in chrome as shown in image below.



8. The solution contains 5 projects as shown below.



9. The starting point for code is InventoryManagement -> Controllers ->InventoryManagementController

10. There are four methods namely **AddItemInInventory**, **UpdateItemInInventory**, **GetAllItems**, **DeleteItemFromInventory**.

**11. Test AddItemInInventory method:**

11.1. Select type as POST, given url in the tab as http://localhost:11880/InventoryManagement/AddItemInInventory .

The local host name will change as you run the project.

11.2. In headers section select key as Content-Type, value as application/x-www-form-urlencoded.

11.3. In Body -> Select raw and give json as {"itemName":"Battery", "itemDescription":"Used in remotes", "price":1000,"IsDeleted":false } in text area available below.

11.4. Click on send button.

11.5. The output is seen in the response body if item is added in database.

11.6. Keep debugger point on AddItemInInventory in InventoryManagementController if needed to debug code.

.

**12.Test UpdateItemInInventory method:**

12.1. Select type as POST, given url in the tab as http://localhost:11880/InventoryManagement/UpdateItemInInventory .

The local host name will change as you run the project.

12.2. In headers section select key as Content-Type, value as application/x-www-form-urlencoded.

12.3. In Body -> Select raw and give json as {"itemId":1,"itemName":"Key","itemDescription":"Used for doors","price":200,"IsDeleted":false} in text area available below.

12.4. Send the item id which was create earlier.

12.4. Click on send button.

12.5. The output is seen as Success in the response body if item details are updated.

12.6. Keep debugger point on UpdateItemInInventory in InventoryManagementController if needed to debug code.

**13.Test GetAllItems method:**

13.1. Select type as GET, give url in the tab as <http://localhost:11880/InventoryManagement/GetAllItems>.

The local host name will change as you run the project.

13.2. In headers section select key as Content-Type, value as application/json.

13.3. Click on send button.

13.4. The output is seen in the response body

13.5. Keep debugger point on GetAllItems in InventoryManagementController if needed to debug code.

**14.Test DeleteItemFromInventory method:**

14.1. Select type as POST, given url in the tab as http://localhost:11880/InventoryManagement/DeleteItemFromInventory? itemId=1 .

The local host name will change as you run the project.

14.2. In headers section select key as Content-Type, value as application/x-www-form-urlencoded..

14.3. Click on send button.

14.4. The output is seen as Success in the response body if item is deleted.

14.5. Keep debugger point on DeleteItemFromInventory in InventoryManagementController if needed to debug code.

**15. To run test cases:**

Goto project InventoryManagement.Tests -> InventoryManagementTests.cs

Click on Test in menu bar as shown in screenshot below.

