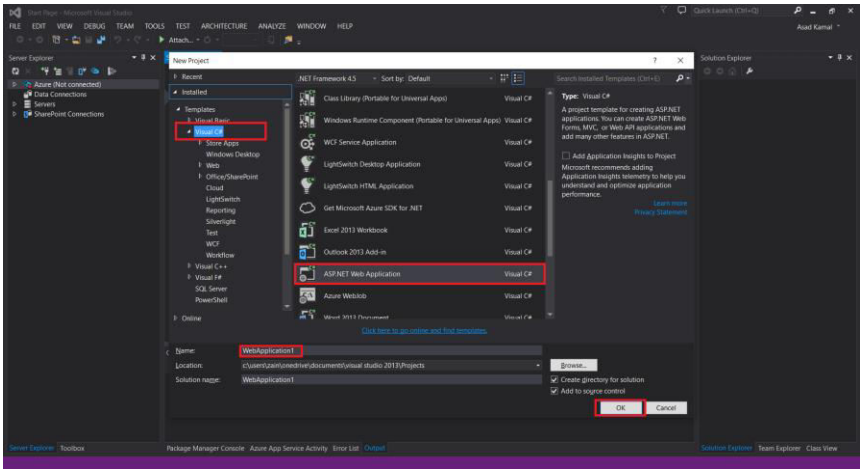
Entity framework 6

Step 1:

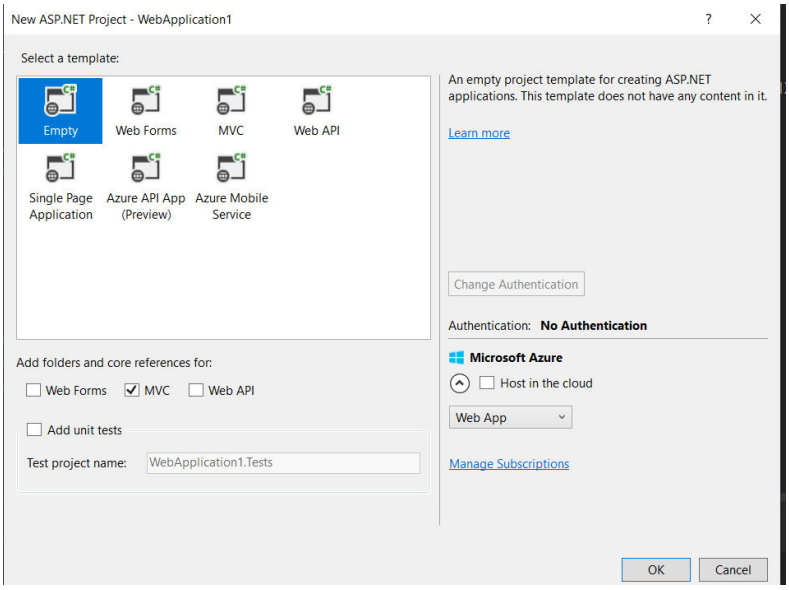
Create a Asp.net Application. Open visual studio. Select **File->New->Project**

Chose **Visual C#** and after that chose Asp.Net Framework web application.

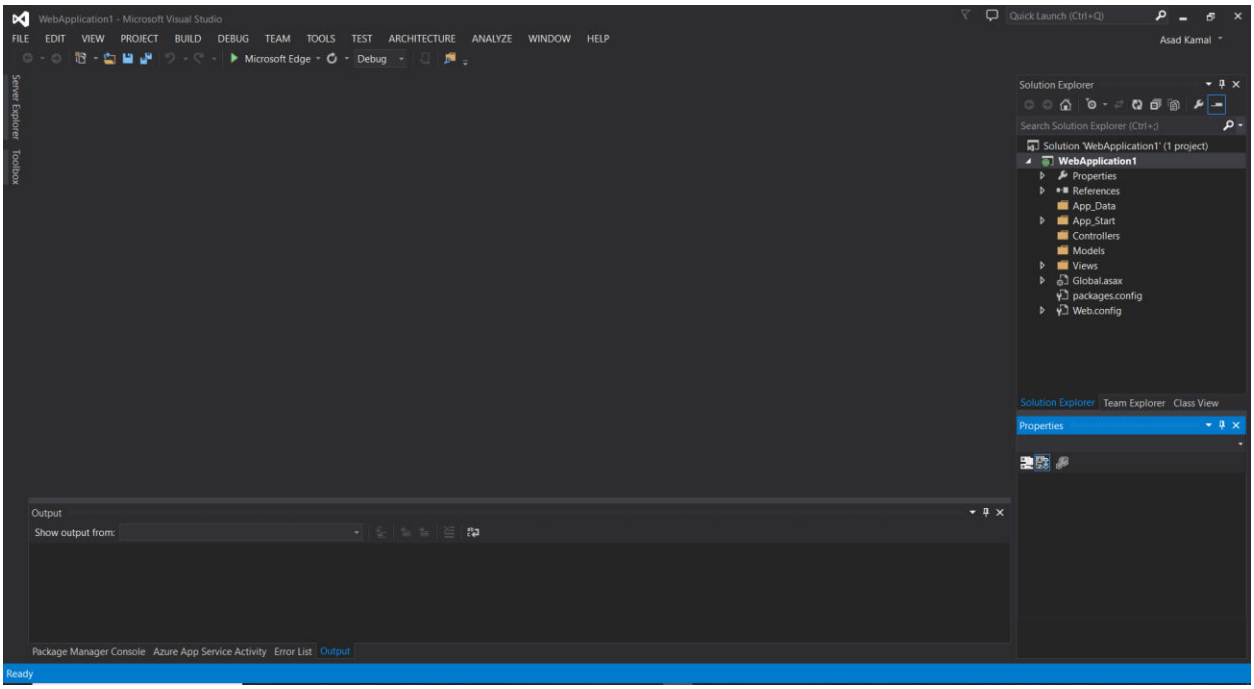
Choose name if you want to and then click ok button



After that a new window will be open. In that window select **Empty** and check **MVC checkbook** and after that click okay button

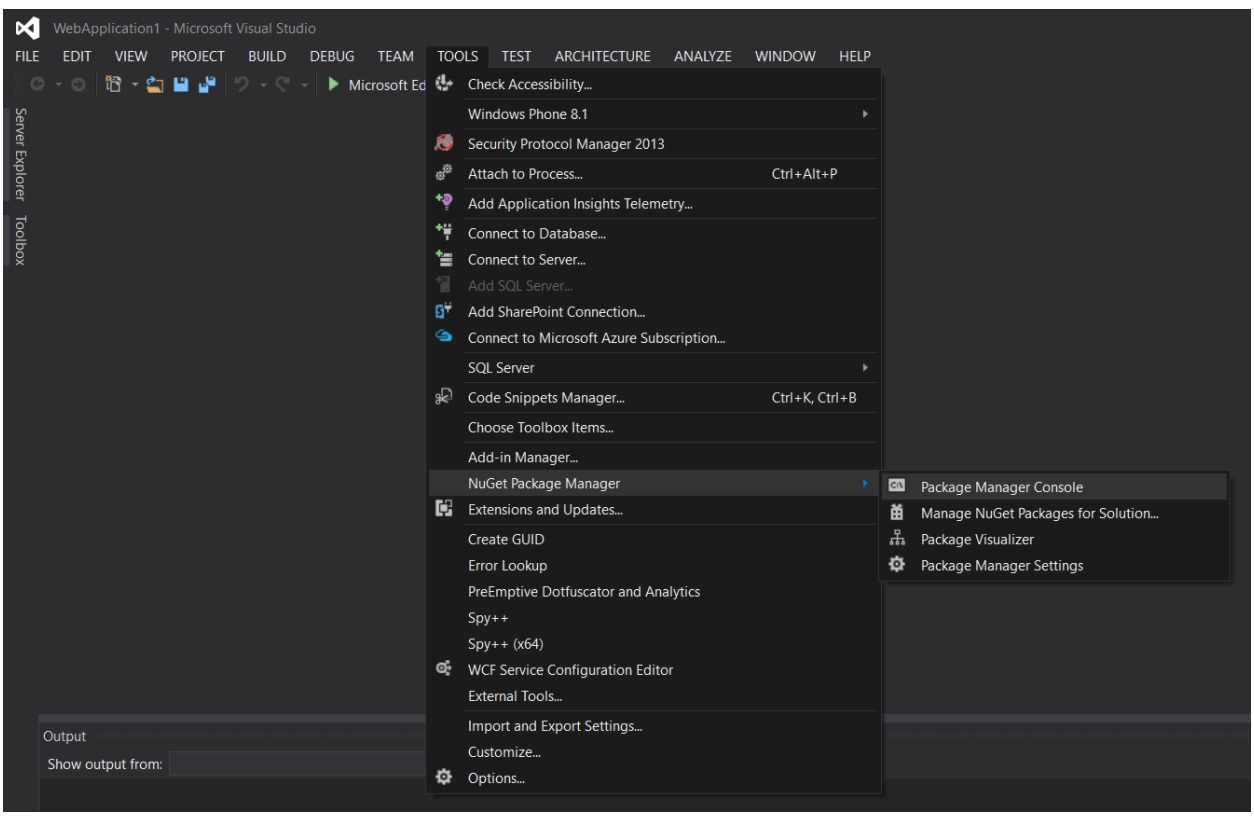


You project has been created successfully.



Step 2: Active nugget library and then install entity framework

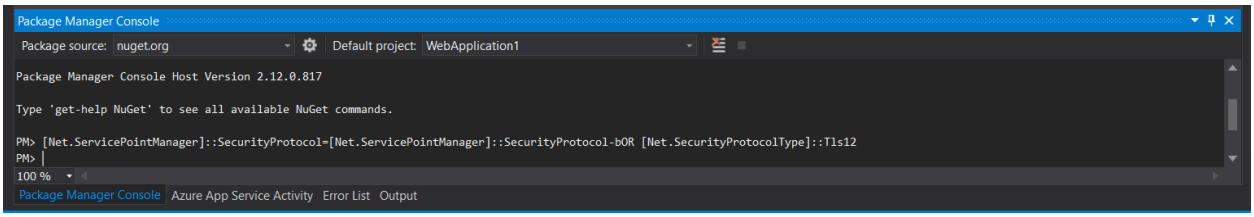
First go to **tools-> nugGet package manager-> Package Manager Console**



A window will be shown in the bottom of the window. Copy and past this command to console.

[Net.ServicePointManager]::SecurityProtocol=[Net.ServicePointManager]::SecurityProtocol-bOR [Net.SecurityProtocolType]::Tls12

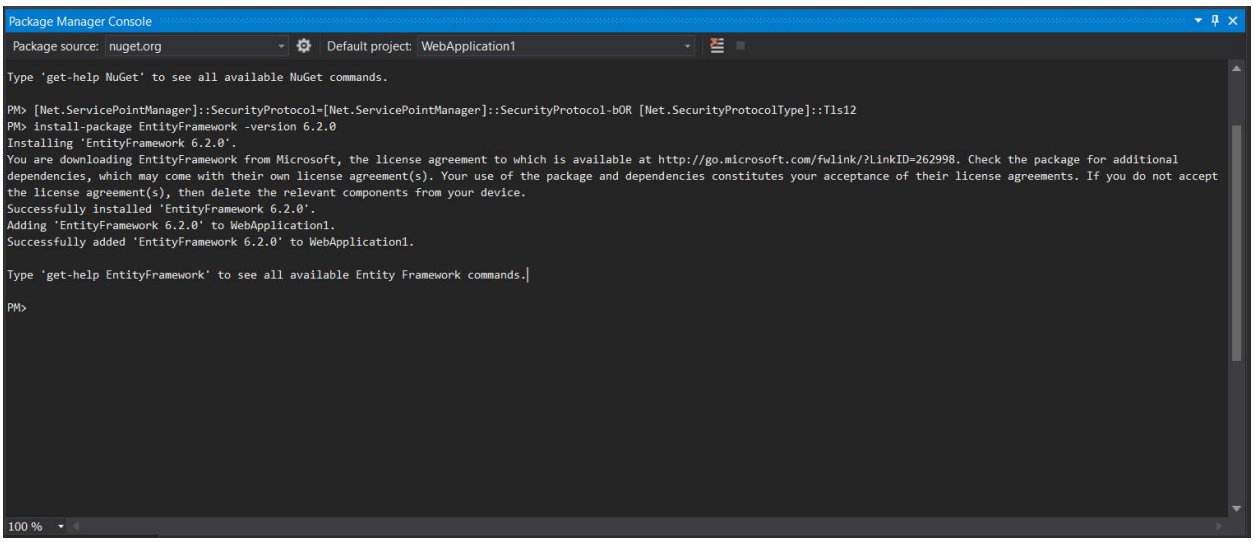
And hit enter.



Now install EntityFramework into the project.

Type command

`install-package EntityFramework -version 6.4.4` and hit Enter button



It will show the successfully added message in the console.

Or You can Install through GUI tool of nuGet Library by right click on project then click on Manage NuGet Package for Solution

A new window will open.

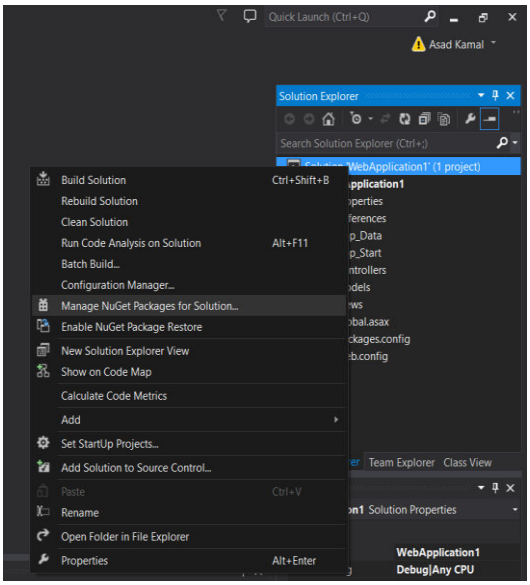
In this window select **Online** from left panel

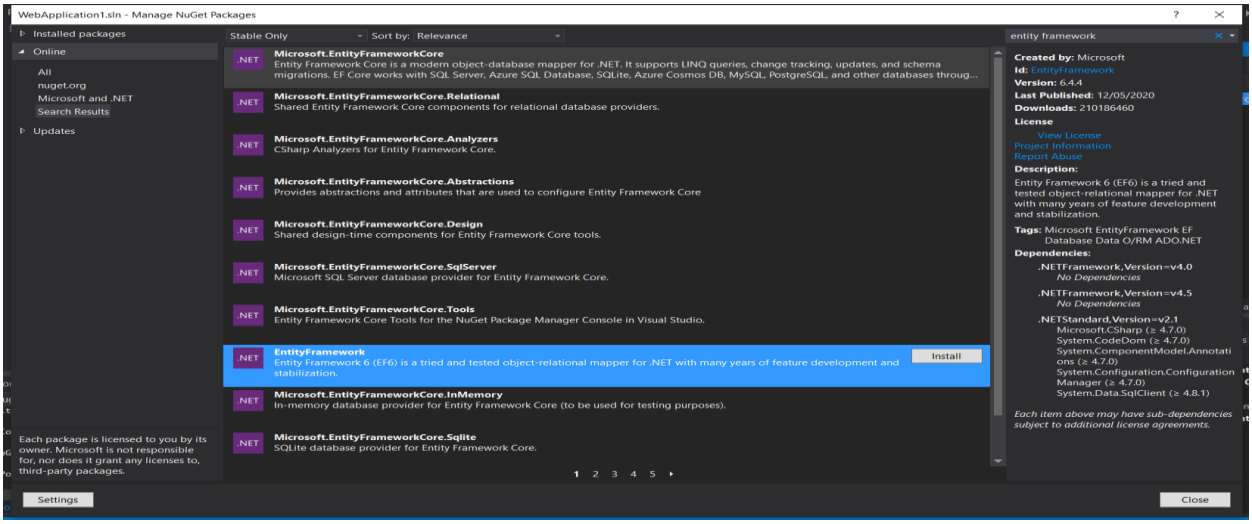
After that Write **Entity Framework** in **search bar** on right side

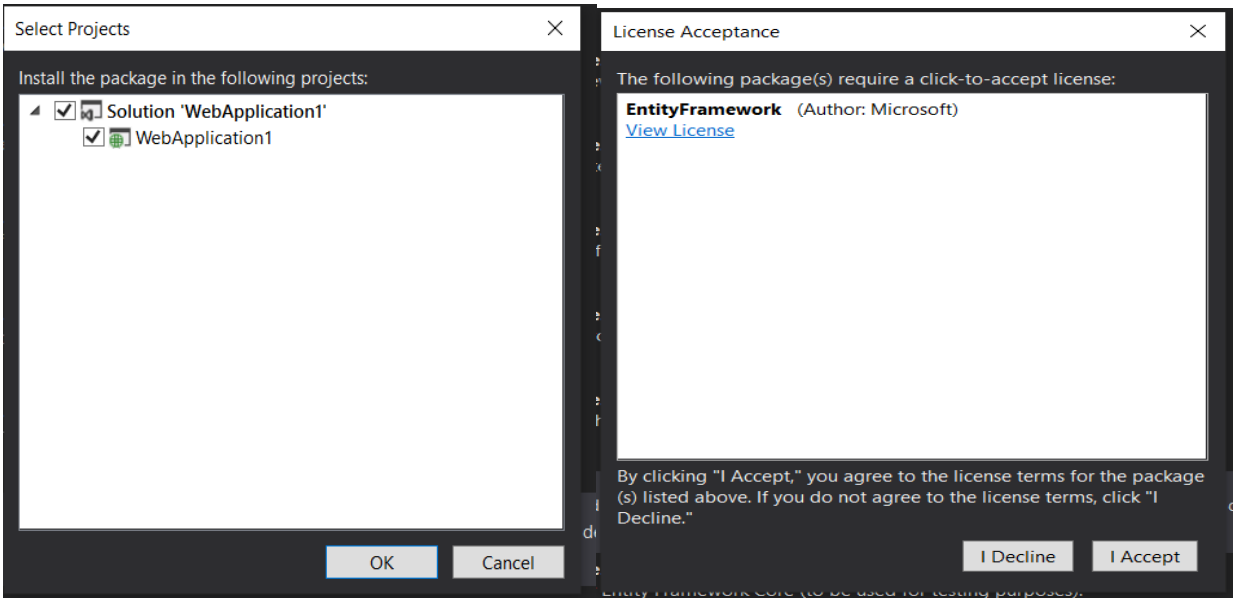
Then all entity framework libraries will be shown to you. Now select **Entity Framework 6** from that as shown in to the given below image.

After that click **install** button. And then **OK** button

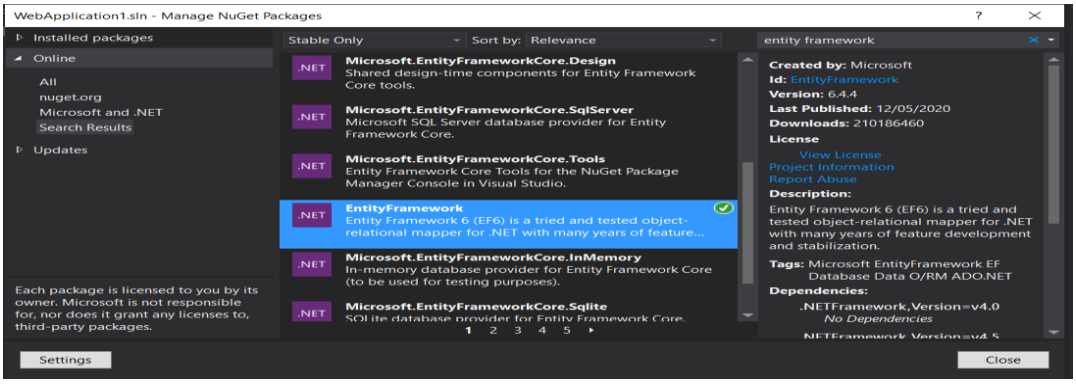
Then Click on **I Accept** Button







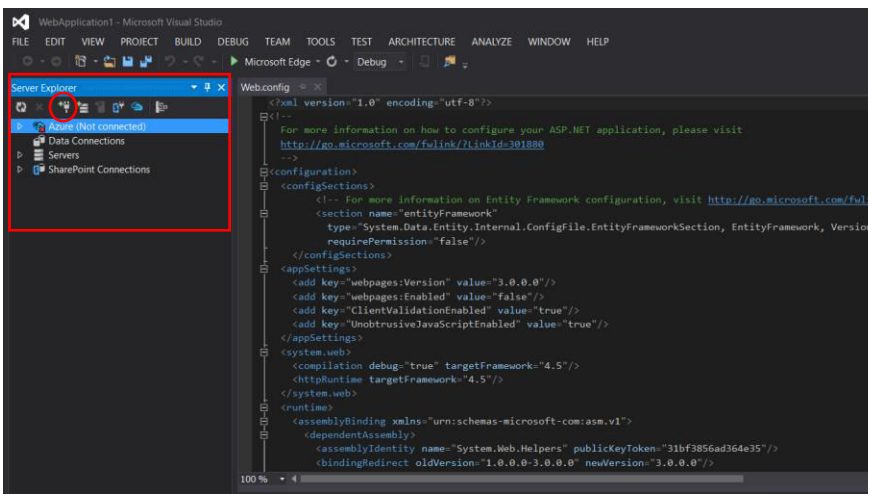
After Installation it will show green tick sign on that package and then click on **close** button



Now go to web.config file from right side panel. The red box line is showing that entity framework is installed in this project

```
Web.config
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=301880
  -->
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
      type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e08
      requirePermission="false"/>
  </configSections>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0"/>
    <add key="webpages:Enabled" value="false"/>
    <add key="ClientValidationEnabled" value="true"/>
    <add key="UnobtrusiveJavaScriptEnabled" value="true"/>
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.5"/>
    <httpRuntime targetFramework="4.5"/>
  </system.web>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35"/>
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0"/>
```

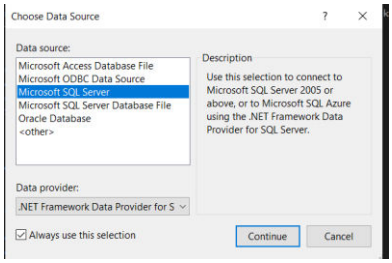
Now click View->server Explorer



Now click on Connection (Shown in red circle) and add new connection.

Choose sql server and click continue

After that a new window will open on that window you will enter the given Ip address and



Now you will create a new connection with the database like shown in the picture.

First Enter **Ip Address/Server Name** in to the Server Name portion. Like I enter **192.168.10.112**

Choose. Use SQL Server Authentication

Write Usernname: Like I write **L1F19BSCS0018**

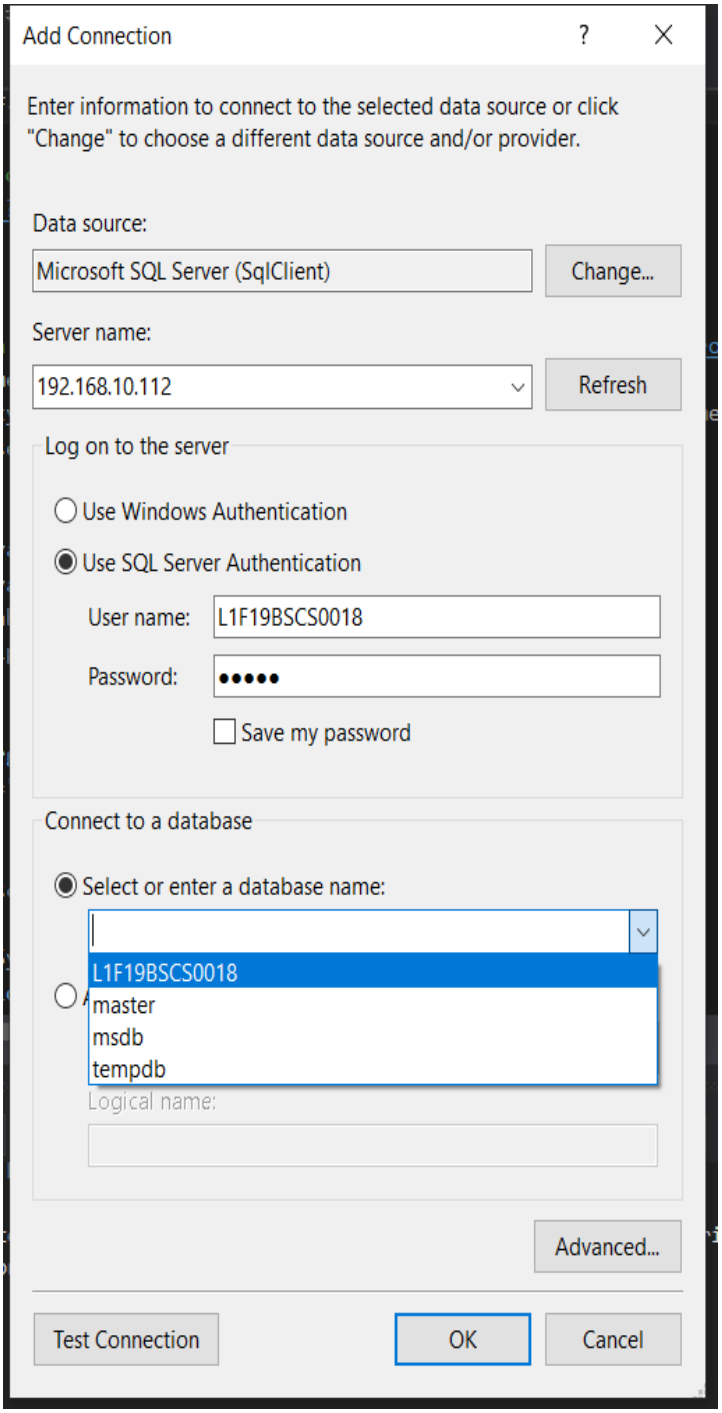
Enter Password: test1

After that click on the dropdown.

It will show all Databases available for this Id

Now choose the database like I choose

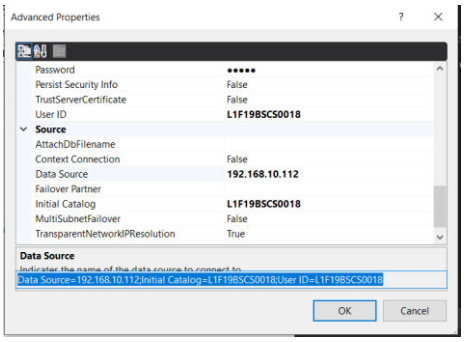
L1F19BSCS0018 Database from available databases.



After that click on the Test Connection button this will test the connection credentials and show to success message if connection is built successfully. If not it show error message.

If connection is built successfully then click on the advance button and copy the connection string from the below of the window.

Copy the connection string and save in some file that will be used later. (The blue selected portion)



Now come to the web.config file

Add connection strings tag after app settings tag as shown in image

In this connectionString= “Data Source=192.168.10.112;Initial Catalog=L1F19BSCS0018;User ID=L1F19BSCS0018; Password=test1” Is same string that we copy except password portion we add it after pasting the string.

```
<connectionStrings>
  <add name="conn" connectionString="Data Source=192.168.10.112;Initial Catalog=L1F19BSCS0018;User ID=L1F19BSCS0018; Password=test1"
      providerName="System.Data.SqlClient"/>
</connectionStrings>
```

```
In this connection string
Data Source is Server Name of databse server
Intital Catalog is Database Name
User Id is Username of server
Password is the Password of the server.
```

Now we will add model in Model folder

First add class of name **Student**

Now add two libraries in the file

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
```

After that add class with basic datamembers

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace WebApplication1.Models
{
    public class Student
    {
        [Key, Column("StudentId", Order = 0)]
        [Required(ErrorMessage = "You must enter Student Id")]
        [DisplayName("Student Id")]
        public int Id { get; set; }
        [Required(ErrorMessage = "Enter Student Name")]
        [DisplayName("Student Name")]
        public string Name { get; set; }
```

```
        [Column(TypeName = "varchar")]
        public string Email { get; set; }
        public string Password { get; set; }
        public Grade GradeId { get; set; }
    }
}
```

GradedId will be Foreign Key in the student Class from **Grade** class

After that We will Create the **Geade** Class

First add three libraries

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
```

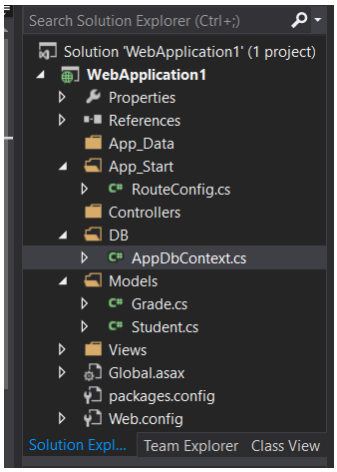
After that add class with basic datamembers

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Security.Permissions;
using System.Web;

namespace WebApplication1.Models
{
    public class Grade
    {
        [Required]
        public int Id { get; set; }
        public string GradeName { get; set; }
        public string GradeDescription { get; set; }

    }
}
```

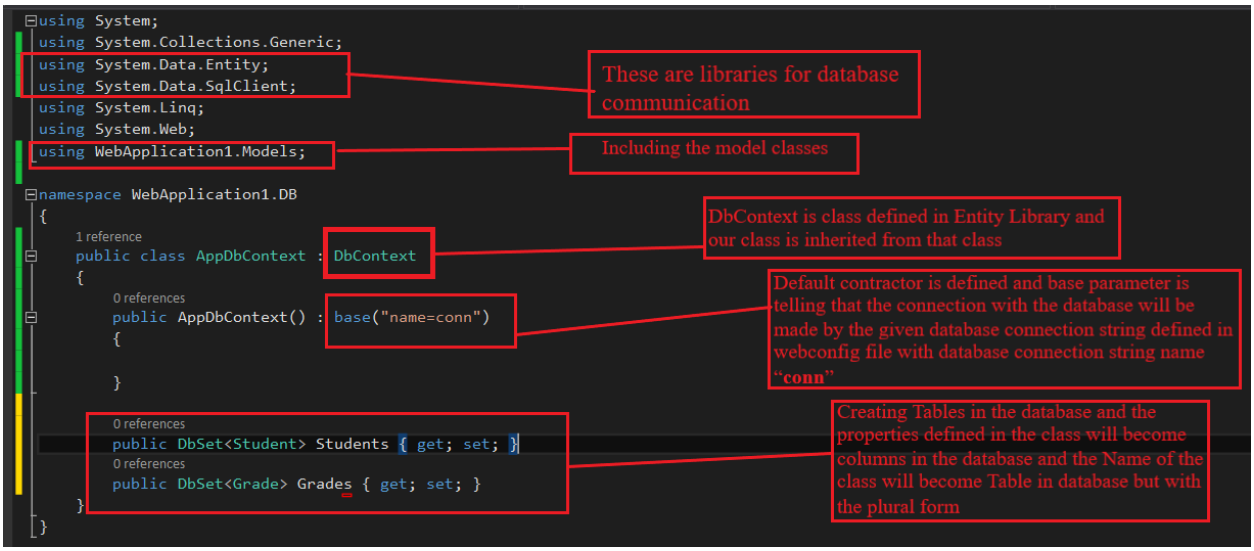
Now Add a new folder of name DB in to the solution.

In this folder add a new class of name AppDbContext.cs

In this file we will create the Database Context Class which will communicate the database for of CRUD operation.



The AppDbContext class will be defined



Code:

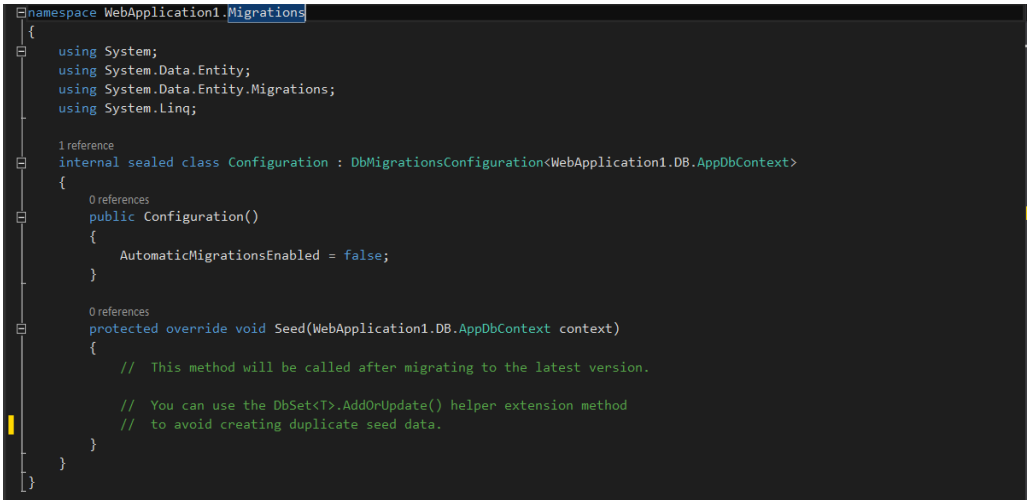
```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using WebApplication1.Models;

namespace WebApplication1.DB
{
    public class AppDbContext : DbContext
    {
        public AppDbContext() : base("name=conn")
        {

        }

        public DbSet<Student> Students { get; set; }
        public DbSet<Grade> Grades { get; set; }
    }
}
```

Now Run a command in Package Manger console

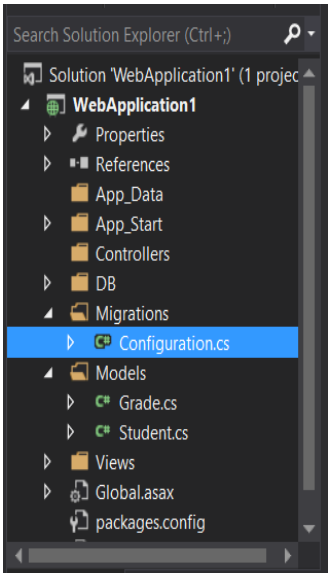
enable-migrations

```
Package Manager Console
Package source: nuget.org        Default project: WebApplication1
PM> [Net.ServicePointManager]::SecurityProtocol=[Net.ServicePointManager]::SecurityProtocol-bOR [Net.SecurityProtocolType]::Tls12
PM> install-package EntityFramework -version 6.4.4
'EntityFramework 6.4.4' already installed.
WebApplication1 already has a reference to 'EntityFramework 6.4.4'.

PM> enable-migrations
Checking if the context targets an existing database...
PM>
100 %
```

This will add **migration** folder and a class of name "**Configuration.cs**"



```
namespace WebApplication1.Migrations
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Migrations;
    using System.Linq;

    1 reference
    internal sealed class Configuration : DbMigrationsConfiguration<WebApplication1.DB.AppDbContext>
    {
        0 references
        public Configuration()
        {
            AutomaticMigrationsEnabled = false;
        }

        0 references
        protected override void Seed(WebApplication1.DB.AppDbContext context)
        {
            //  This method will be called after migrating to the latest version.

            //  You can use the DbSet<T>.AddOrUpdate() helper extension method
            //  to avoid creating duplicate seed data.
        }
    }
}
```



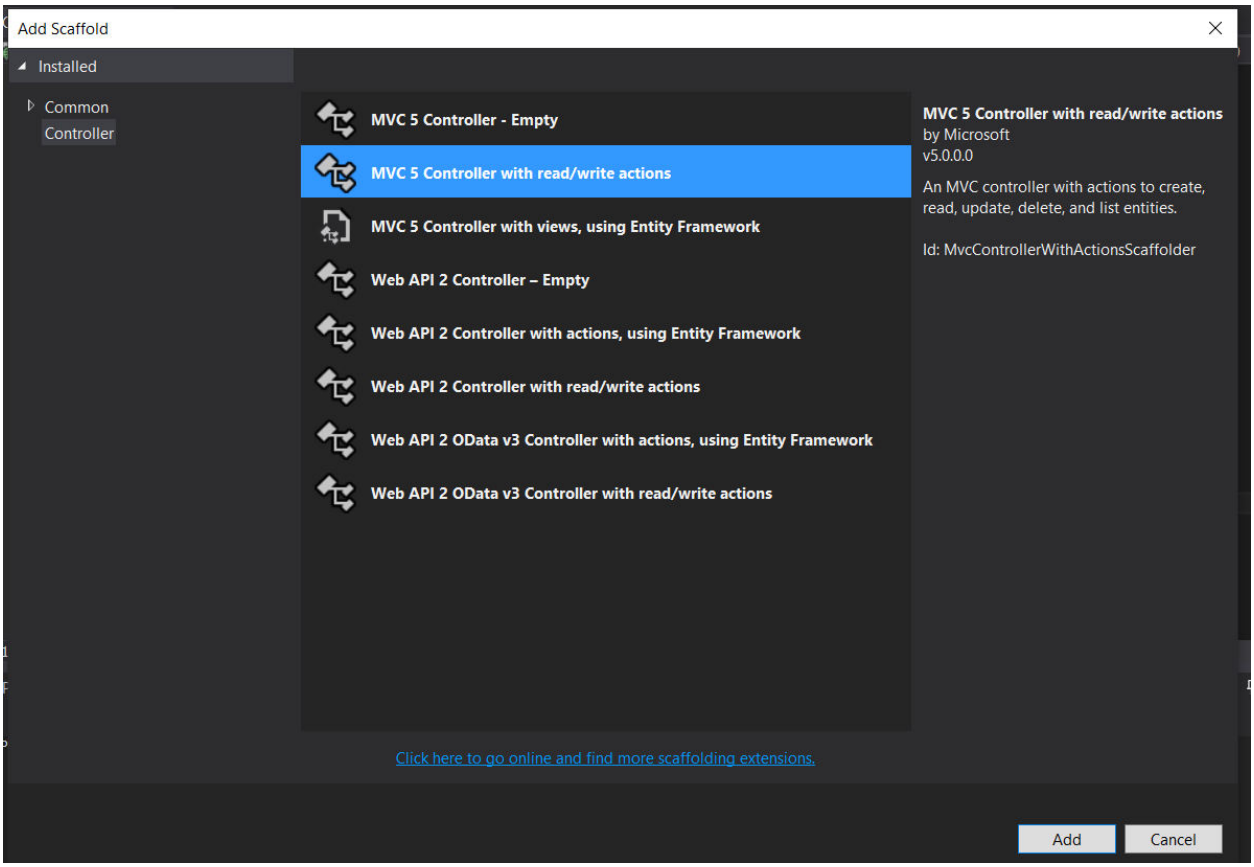
First change the value of AutomaticMigrationEnabled to True.

Now we add seed value in this file. In this context we will add Grades values as seed value.

```
namespace WebApplication1.Migrations
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Migrations;
    using System.Linq;
    using WebApplication1.Models;

    internal sealed class Configuration :
DbMigrationsConfiguration<WebApplication1.DB.AppDbContext>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
        }
        protected override void Seed(WebApplication1.DB.AppDbContext context)
        {
            context.Grades.AddOrUpdate(new Grade { Id = 1, GradeName = "A", GradeDescription
= "A Grade" },
                new Grade { Id = 2, GradeName = "B", GradeDescription = " B Grade " },
               new Grade { Id = 3, GradeName = "C", GradeDescription = " C Grade " }
        );
         }
    }
}
```

Step 5:

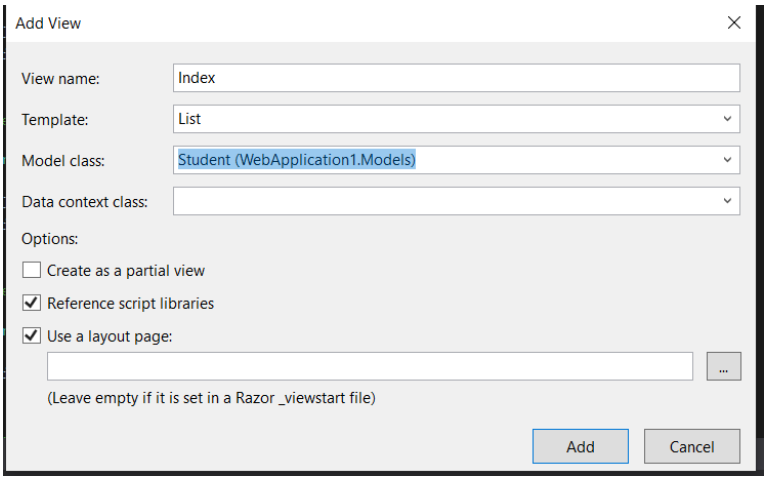
Add controller of Students



After Controller added it will write basic crud operation code by self. All you need it to call the object and store the object into database.

Add View one by one of each action in the controller.

Now, For index chose template List and in model section add model of student. It will write basic code of the list.



Now in controller add few libraries first.

```
using WebApplication1.Models;
using WebApplication1.DB;
```

and add a default constructor of Student controller

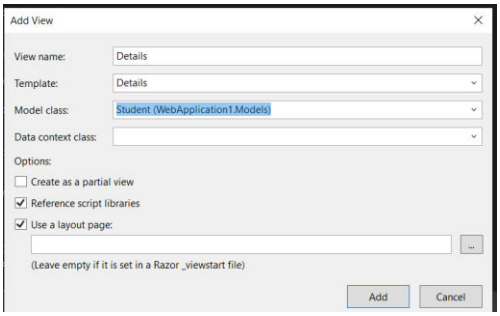
```
AppDbContext dbcontext;
        public StudentsController()
        {
            dbcontext = new AppDbContext();
        }
```

Now, Update the Index Action code by this code

```
public ActionResult Index()
        {
            var model = dbcontext.Students.ToList();
            return View(model);
        }
```

In this code dbcontext object brings all students and convert it to the list and that list is passed to the view. And in view this list will be displayed.

Now we will add the detail action view



In this section we select Template of Details and in model section we select Student Model. And click on Add now it will add complete view with necessary code.

After that update controller action Details code with this code.

```
public ActionResult Details(int id)
        {
            var model = dbcontext.Students.Find(id);
            return View(model);
        }
```

This code will find the specific student from all students and return that data row and convert it in to the student object. And that object has been passed to the view.

Same will be done by the all other actions.

```
dbcontext.Students.Add(std);
dbcontext.SaveChanges();
```

in this code a new student will be added in the database and the changes will be saved.

```
dbcontext.Entry(std).State = System.Data.Entity.EntityState.Modified;
dbcontext.SaveChanges();
```

in this code student data has been updated and the changes will be saved.

```
dbcontext.Students.Remove(std);
dbcontext.SaveChanges();
```

in this code that specific student data has been removed from database and the changes will be saved.