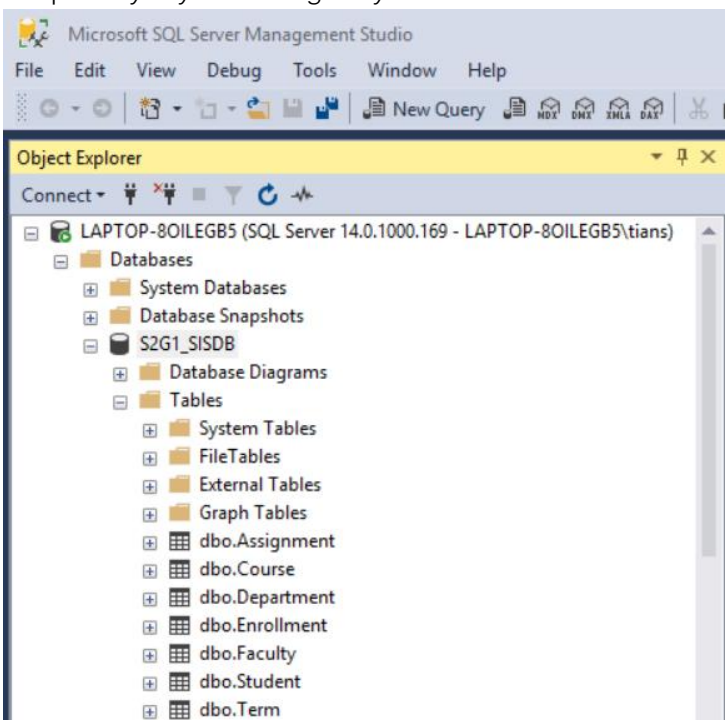
Student Information System

1. Azure and Github

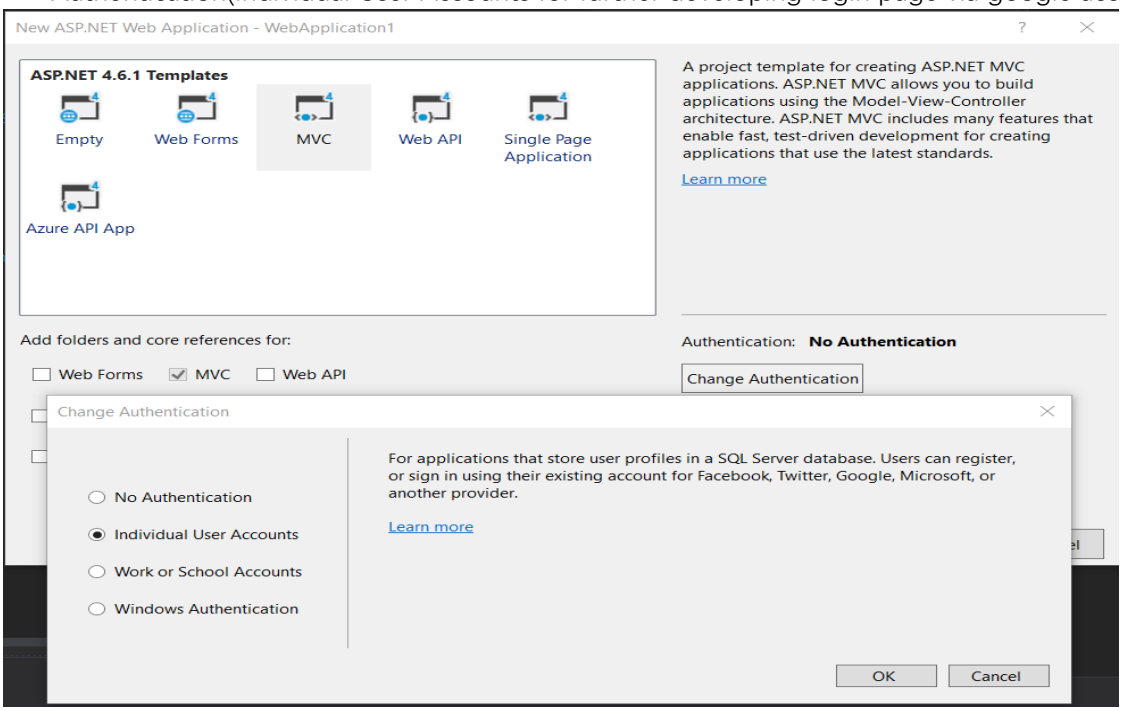
Source code published on github: https://github.com/hongshuidang/courseRegisterSystem
Application deployed on Azure: https://s2g1sis.azurewebsites.net

2. Prerequisite

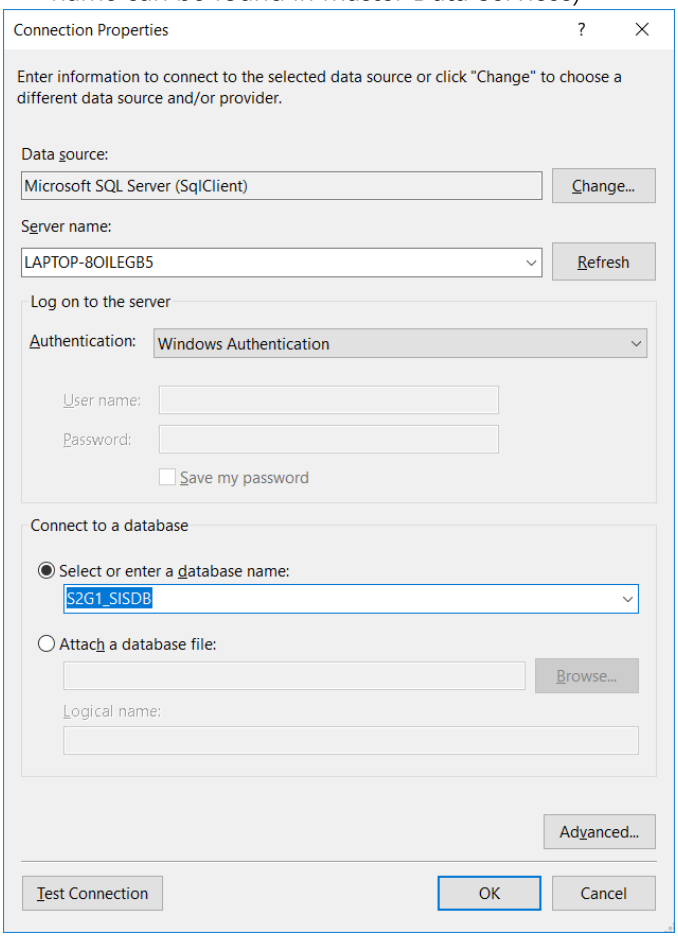
You have designed the schema in your local Sql server Database, including attributes and their domain, and constraints like primary key and foreign keys.

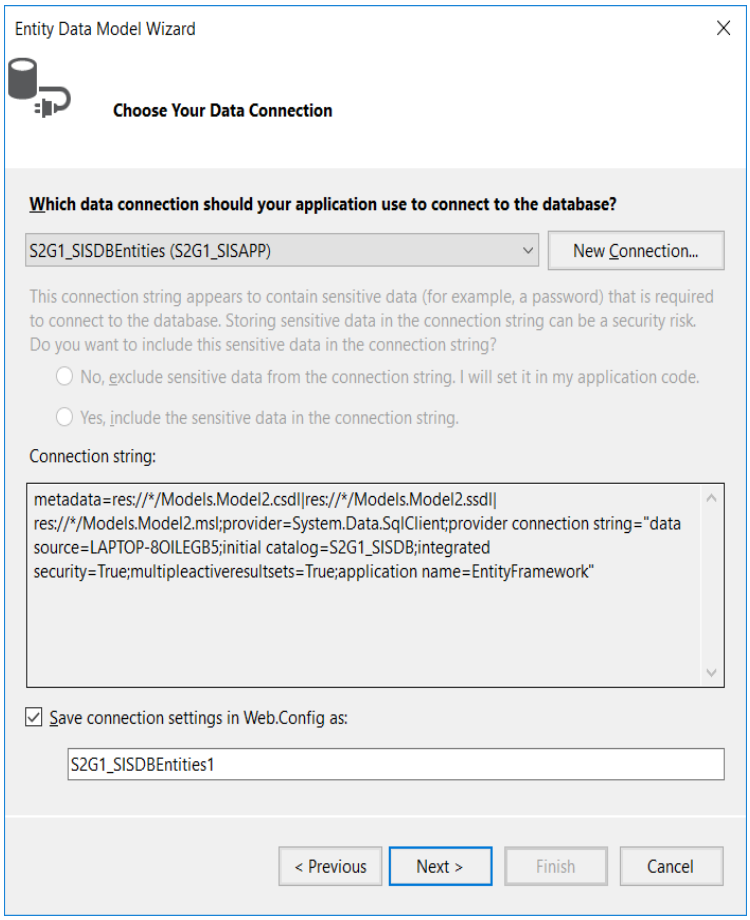


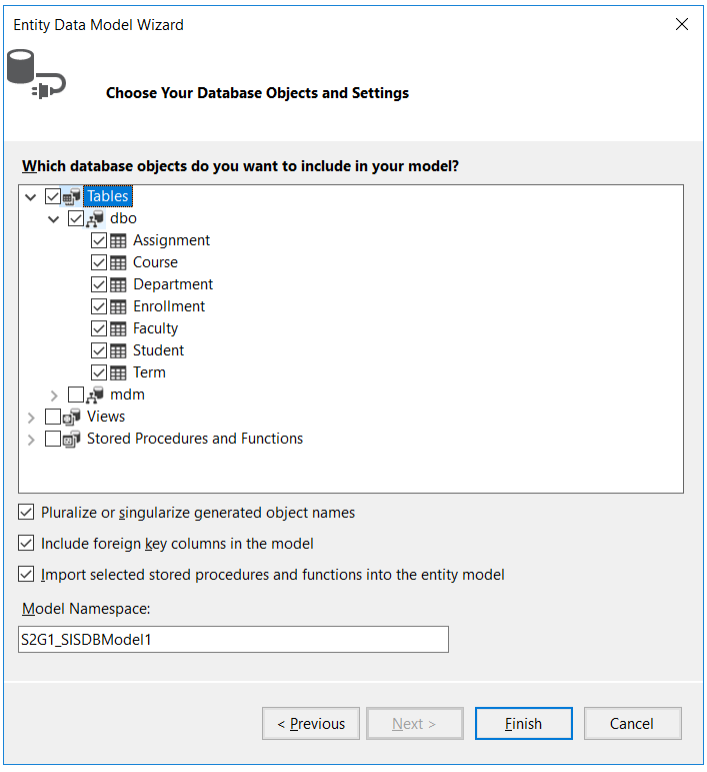
3. Create a new project in Visual Studio Enterprise IDE: File->New->ASP.NET Web Application->MVC->Change Authentication(Individual User Accounts for further developing login page via google account).



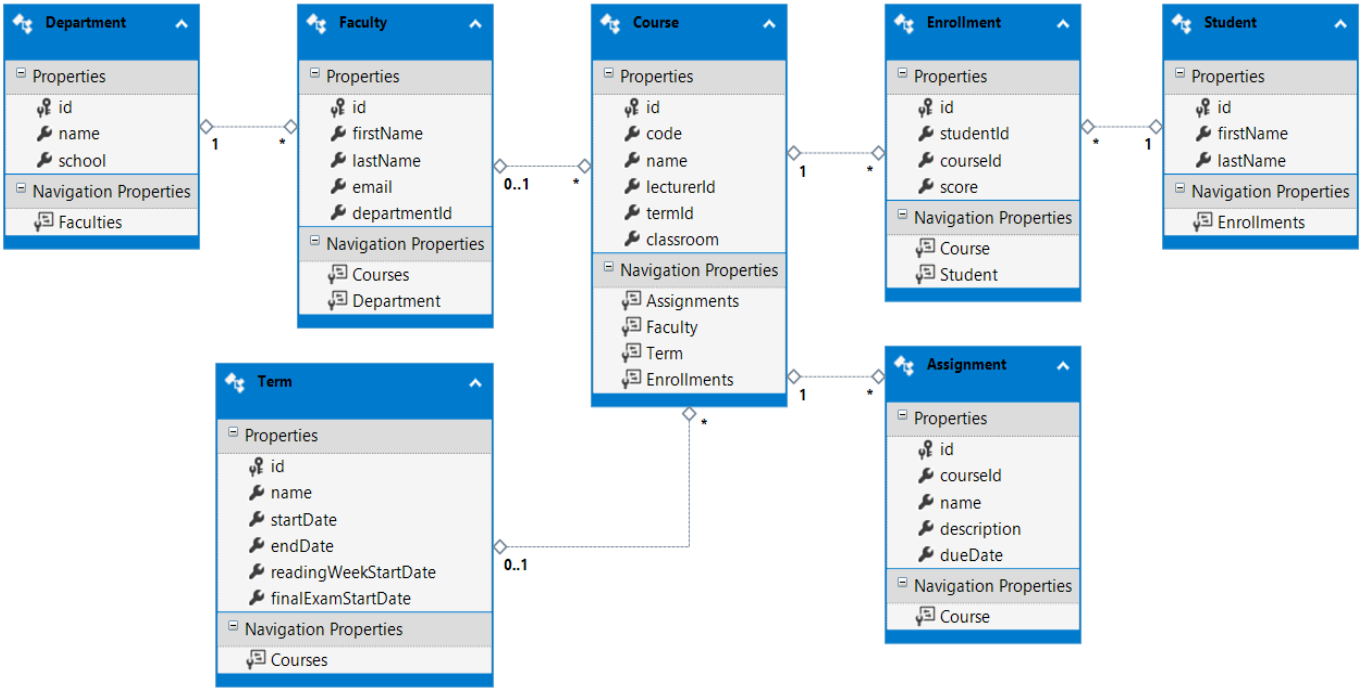
4. Create Model: Models->Add->New Item->Data->ADO.NET Entity Data Model->EF Designer from database(Server name can be found in Master Data Services)

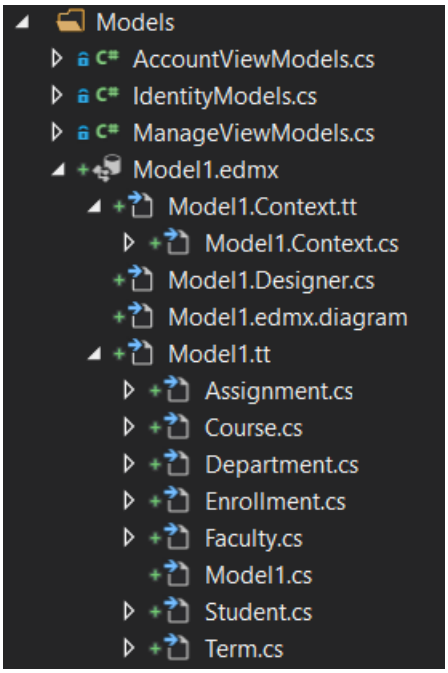






After this step, a class diagram and all entity classes are generated.



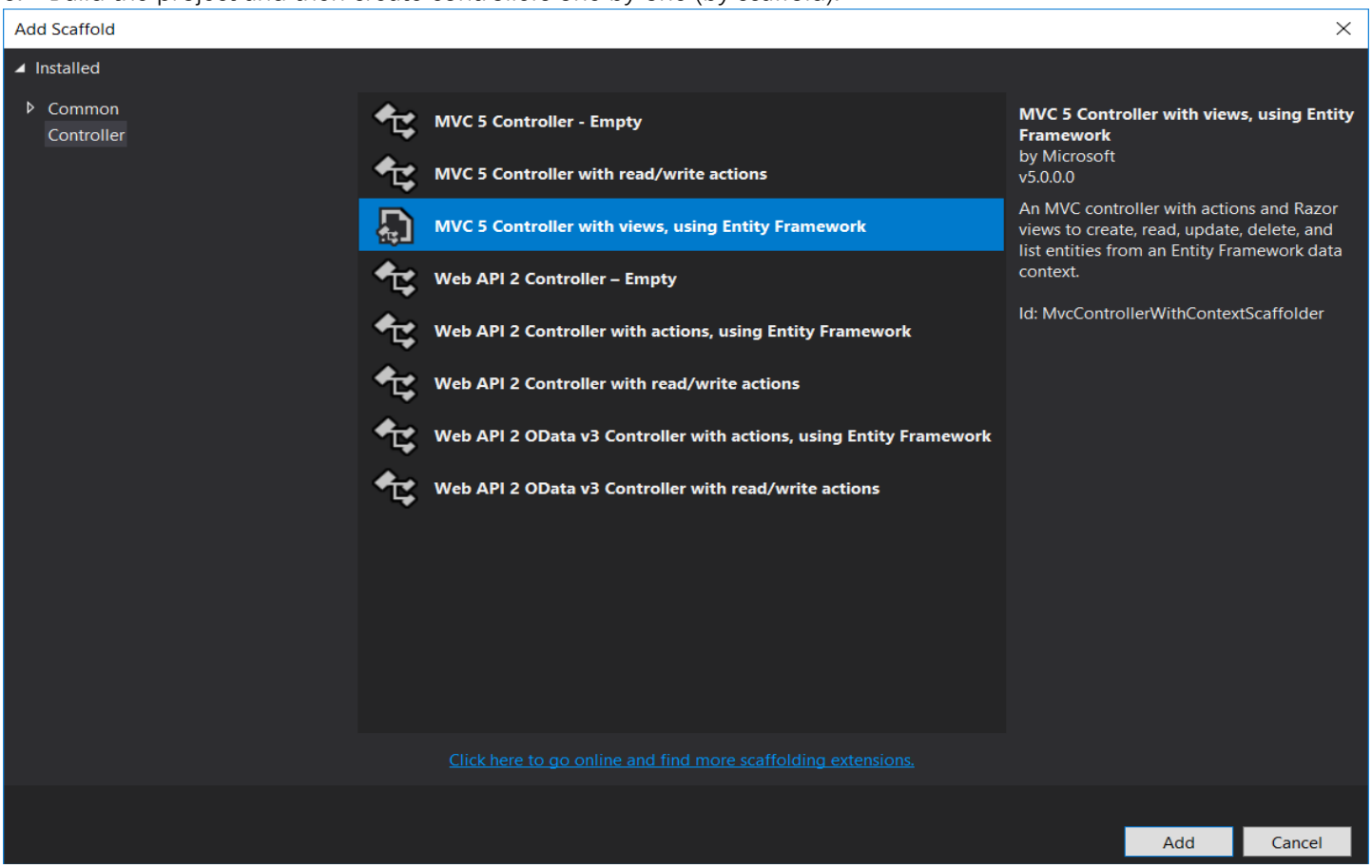


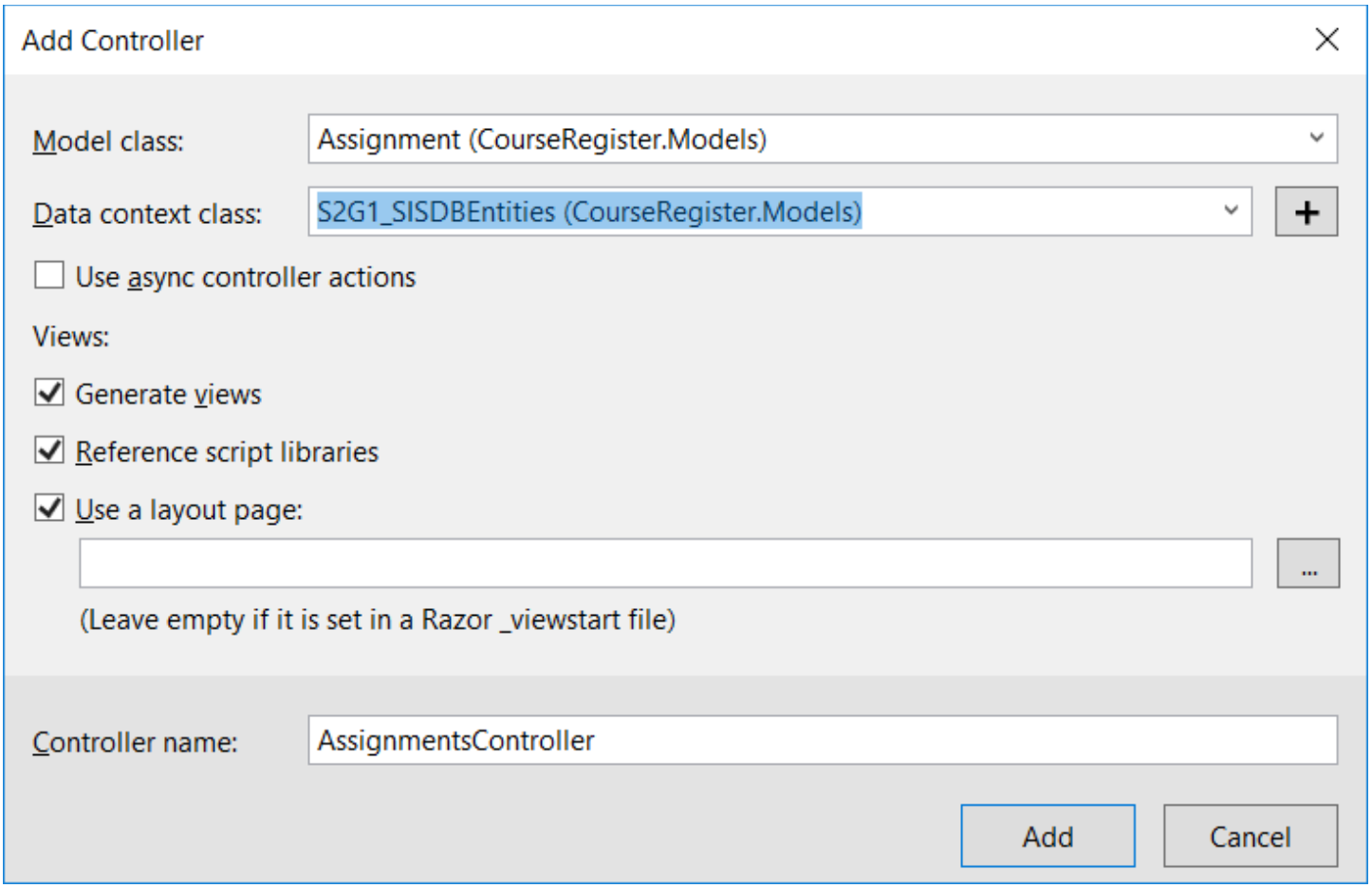

```
namespace CourseRegister.Models
{
    using System;
    using System.Collections.Generic;

    3 references | 0 changes | 0 authors, 0 changes
    public partial class Assignment
    {
        0 references | 0 changes | 0 authors, 0 changes
        public int id { get; set; }
        0 references | 0 changes | 0 authors, 0 changes
        public int courseId { get; set; }
        0 references | 0 changes | 0 authors, 0 changes
        public string name { get; set; }
        0 references | 0 changes | 0 authors, 0 changes
        public string description { get; set; }
        0 references | 0 changes | 0 authors, 0 changes
        public Nullable<System.DateTime> dueDate { get; set; }

        0 references | 0 changes | 0 authors, 0 changes
        public virtual Course Course { get; set; }
    }
}
```

Add also, a connection string is generated in Web.config file:

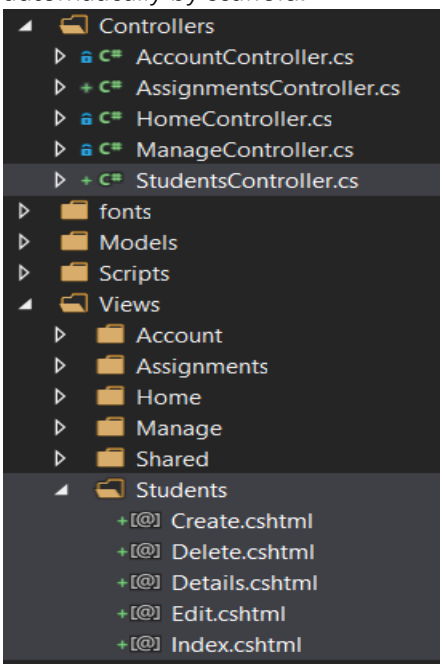
```
<add name="S2G1_SISDBEntities"
connectionString="metadata=res://*/Models.Model1.csdl|res://*/Models.Model1.ssdl|res://*/Models.Model1.
msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=LAPTOP-8OILEGB5;initial
catalog=S2G1_SISDB;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;"
providerName="System.Data.EntityClient" />
```

5. Build the project and then create controllers one by one (by scaffold):

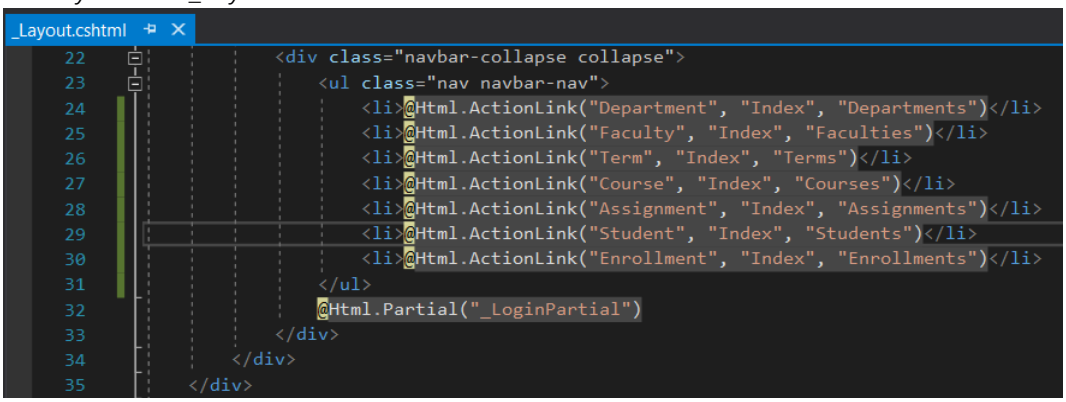




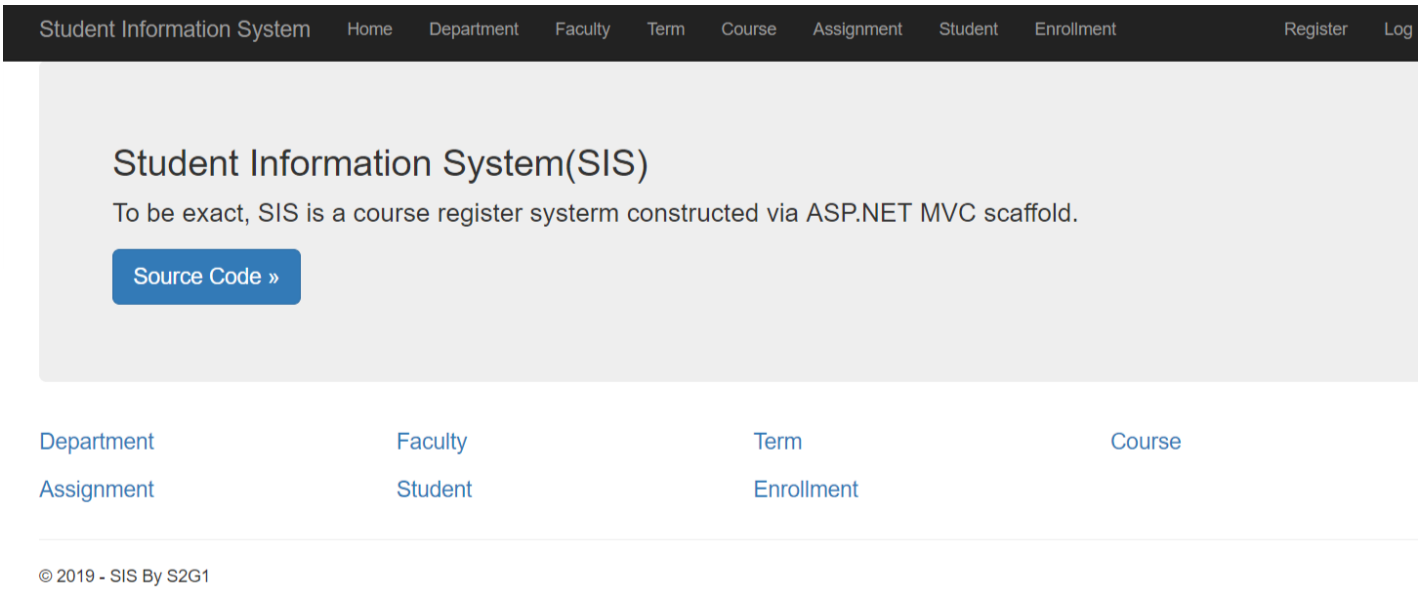
S2G1_SISDBEntities is the name of the connection string and CourseRegister.Models is the namespace(CourseRegister is the previous name of the project and solution). After this step, you can see the controller and its relevant views generated automatically by scaffold.



Modify Shared/_Layout.cshtml and Home/Index.cshtml files.



6. Run application:



7. Remedy

7.1 Change the date type from intrinsic one to smalldatetime:

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

```
[DataType(DataType.Date)]
[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
0 references | 0 changes | 0 authors, 0 changes
public Nullable<System.DateTime> startDate { get; set; }
```

Start Date	End Date	Reading Week Start Date	Final Exam Start Date
2018-09-03	2018-12-19	2018-10-08	2018-12-16
2019-01-04	2019-04-19	2019-02-16	2019-04-15

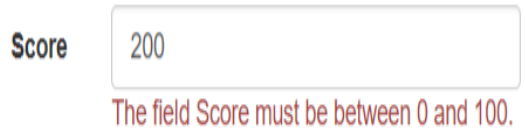
7.2 Change the column name displayed:

```
[Display(Name = "Student's First Name")]
0 references | 0 changes | 0 authors, 0 changes
public string firstName { get; set; }
```

Faculty's First Name	Faculty's Last Name	Faculty's Email	Name of Department
Si	Li	si.li@uwindsor.ca	Computer Science
San	Zhang	san.zhang@uwindsor.ca	Computer Science

7.3 Attribute range

```
[Range(0, 100)]
[Display(Name = "Score")]
0 references | 0 changes | 0 authors, 0 changes
public int score { get; set; }
```



7.4 Foreign key: Department.id is the primary key and Faculty.departmentId is the foreign key referring to the Department

entity.

In Faculty.cs:

```
[ForeignKey("Department")]
[Display(Name = "Department Id")]
3 references | 0 changes | 0 authors, 0 changes
public int departmentId { get; set; }
```

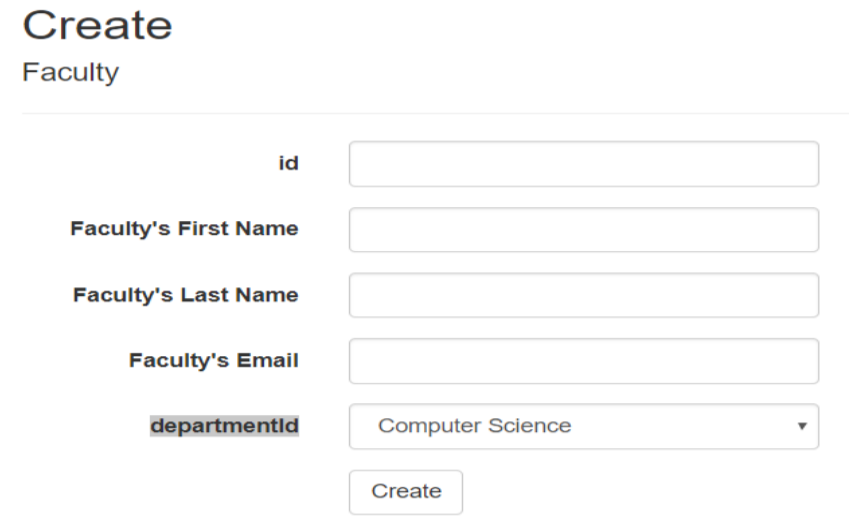
```
1 reference | 0 changes | 0 authors, 0 changes
public virtual Department Department { get; set; }
```

In FacultiesController.cs

```
// GET: Faculties/Create
0 references | 0 changes | 0 authors, 0 changes
public ActionResult Create()
{
    ViewBag.departmentId = new SelectList(db.Departments, "id", "name");
    return View();
}
```

```
SelectList.SelectList(System.Collections.IEnumerable items, string dataValueField, string dataTextField) (+ 8 overloads)
Initializes a new instance of the SelectList class by using the specified items for the list, the data value field, and the data text field.
```

That is why in the create web page you can see the label departmentId while the value list shows you the name list rather than the department id list.



While in fact the attribute the action inserts into database is not the value of department name, but right the department id. It is a little bit tricky but meaningful. If you really want to show value list of department id on web page, just modify the third parameter in the SelectList function above.

```
// POST: Faculties/Create ...
[HttpPost]
[ValidateAntiForgeryToken]
0 references | 0 changes | 0 authors, 0 changes
public ActionResult Create([Bind(Include = "id,firstName,lastName,email,departmentId")] Faculty faculty)
{
    if (ModelState.IsValid)
    {
        db.Faculties.Add(faculty);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    ViewBag.departmentId = new SelectList(db.Departments, "id", "name", faculty.departmentId);
    return View(faculty);
}
```

SQLQuery9.sql - LA...OILEGB5\tians (57))*

```
select * from Faculty
```

100 %

Results | Messages

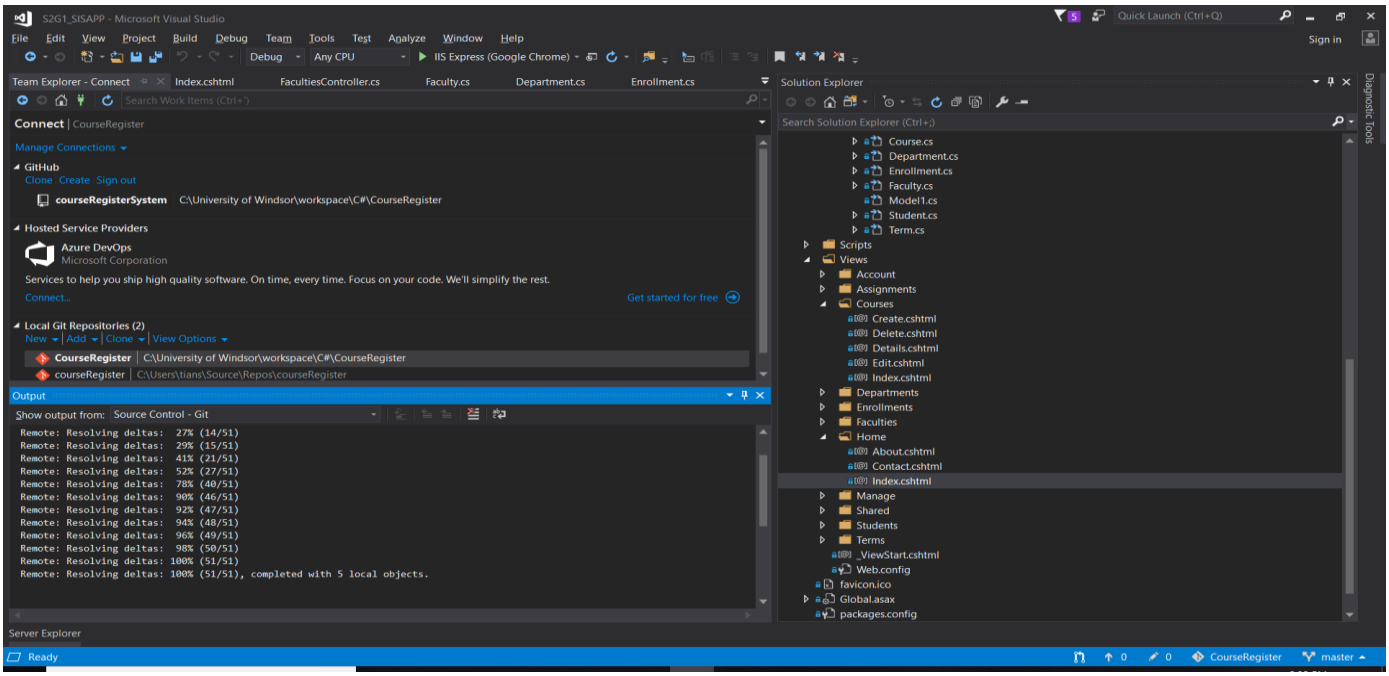
	id	firstName	lastName	email	departmentId
1	F-001	Si	Li	si.li@uwindsor.ca	1
2	F-002	San	Zhang	san.zhang@uwindsor.ca	1
3	F-003	Wu	Wang	wu.wang@uwindsor.ca	2

7.5 If the database table does not contain an attribute called id or className+Id (for e.g. StudentId), scaffold may fail to generate entity class. To walk around, a [key] annotation should be added in front of the primary non-id attribute.

```
[Key]
[ForeignKey("Instructor")]
public int InstructorID { get; set; }
```

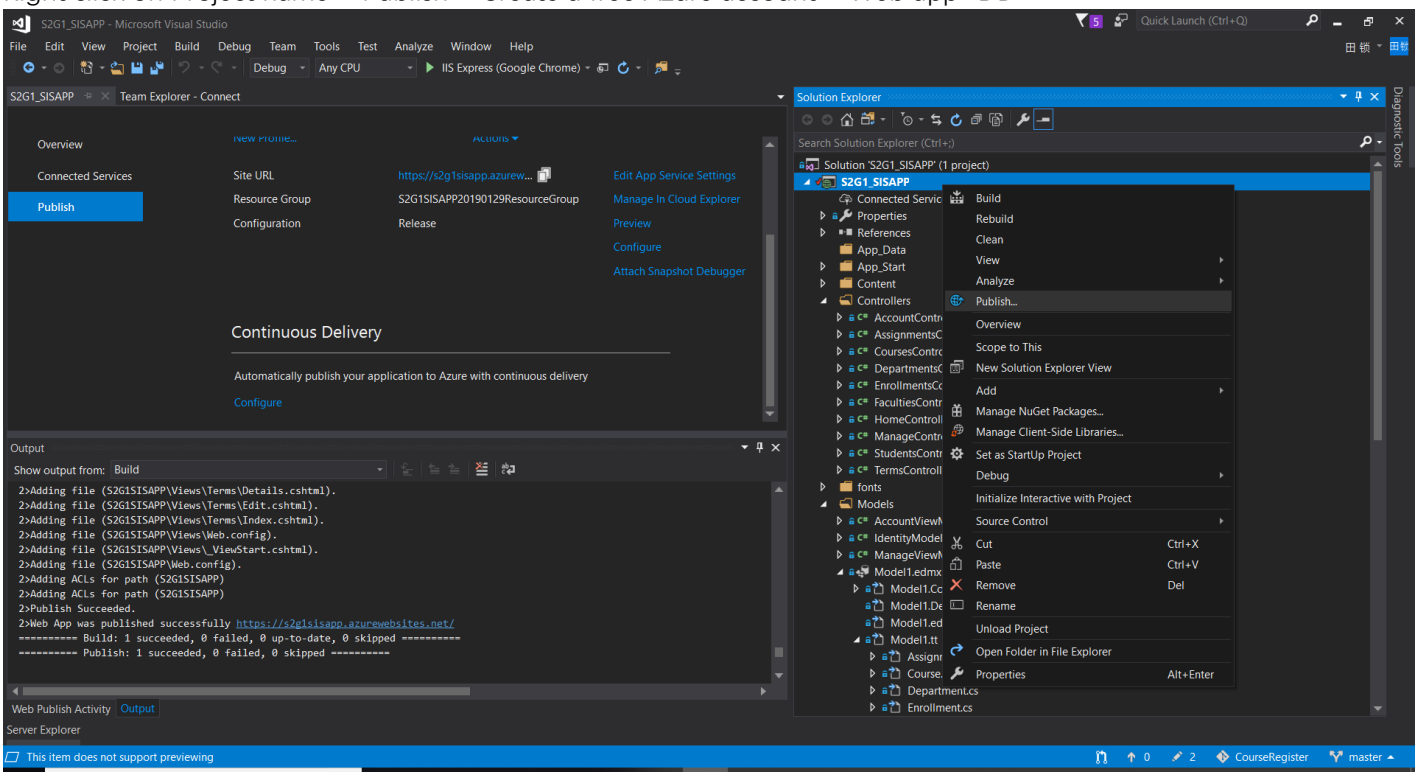
8. Share it onto Github

Download Github Extension for Visual Studio and install it. VS->View->Team Explorer->Connect. Publish to Github and then Push, Pull, etc

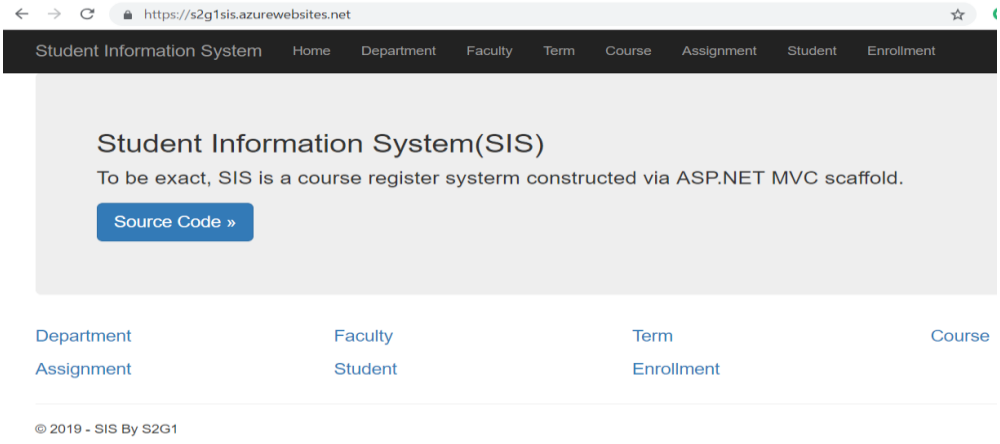


9. Launch the application to Azure

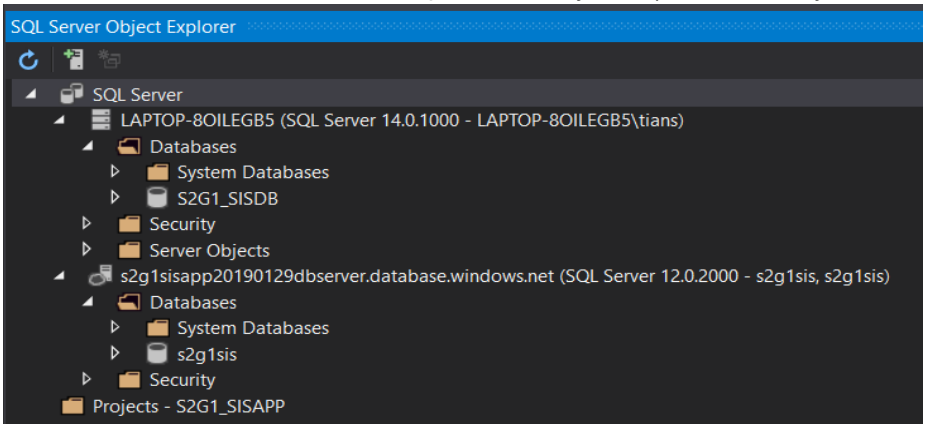
Right click on Project name->Publish->Create a free Azure account->Web app+DB



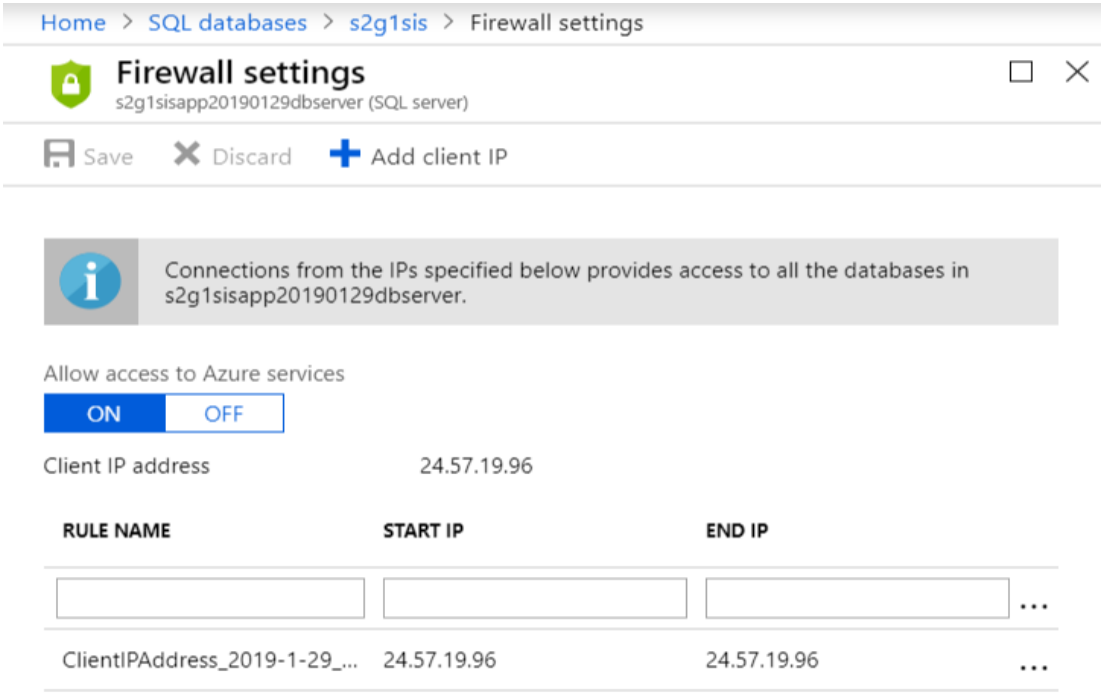
A web page with an Azure url will pop up. It is cool, but you cannot access any further functions until now, because of the connection issue between Azure app and Azure DB.



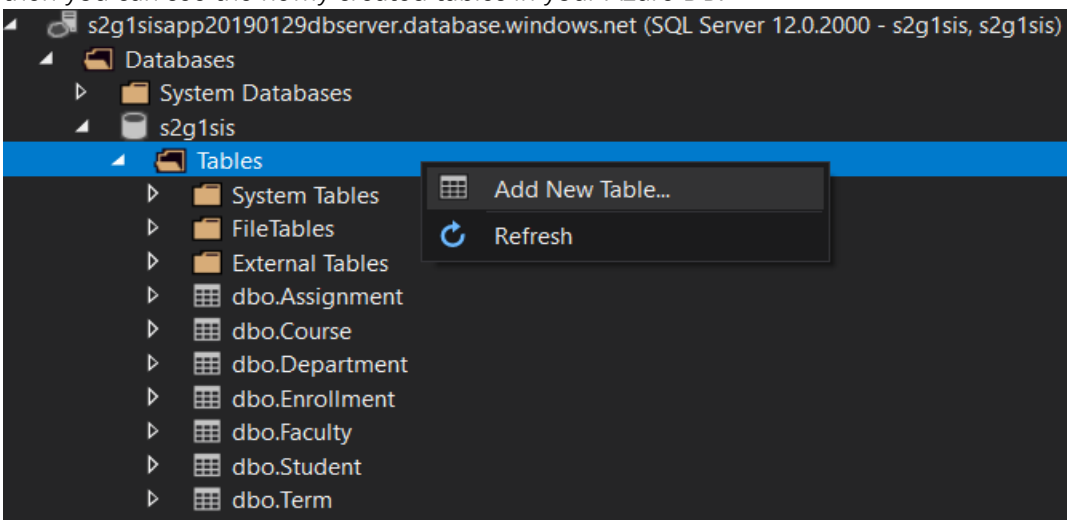
Go back to Visual Studio: View->SQL Server Object Explorer where you can edit local DB and Azure DB as well.



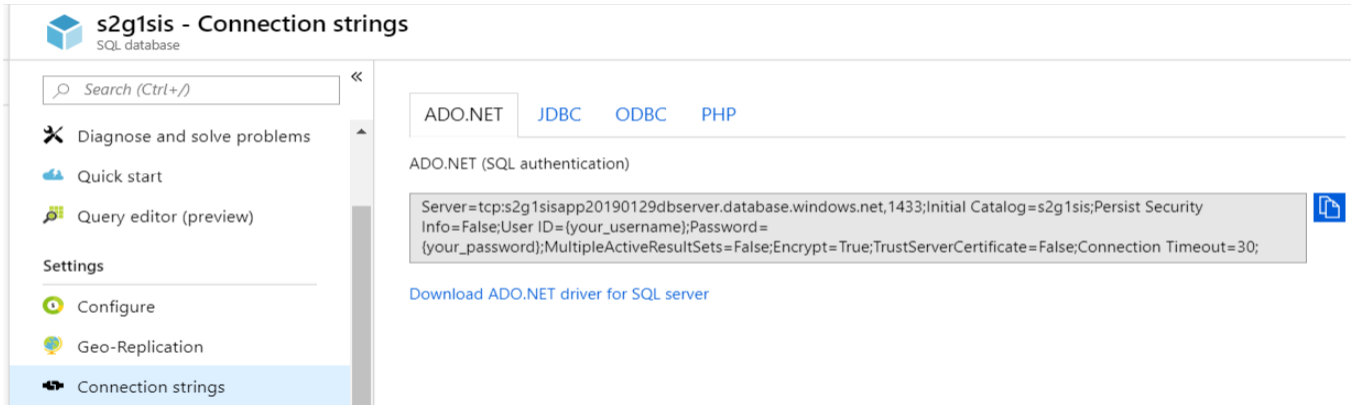
If you get error about firewall rule, go back to Azure portal authorizing your working ip.

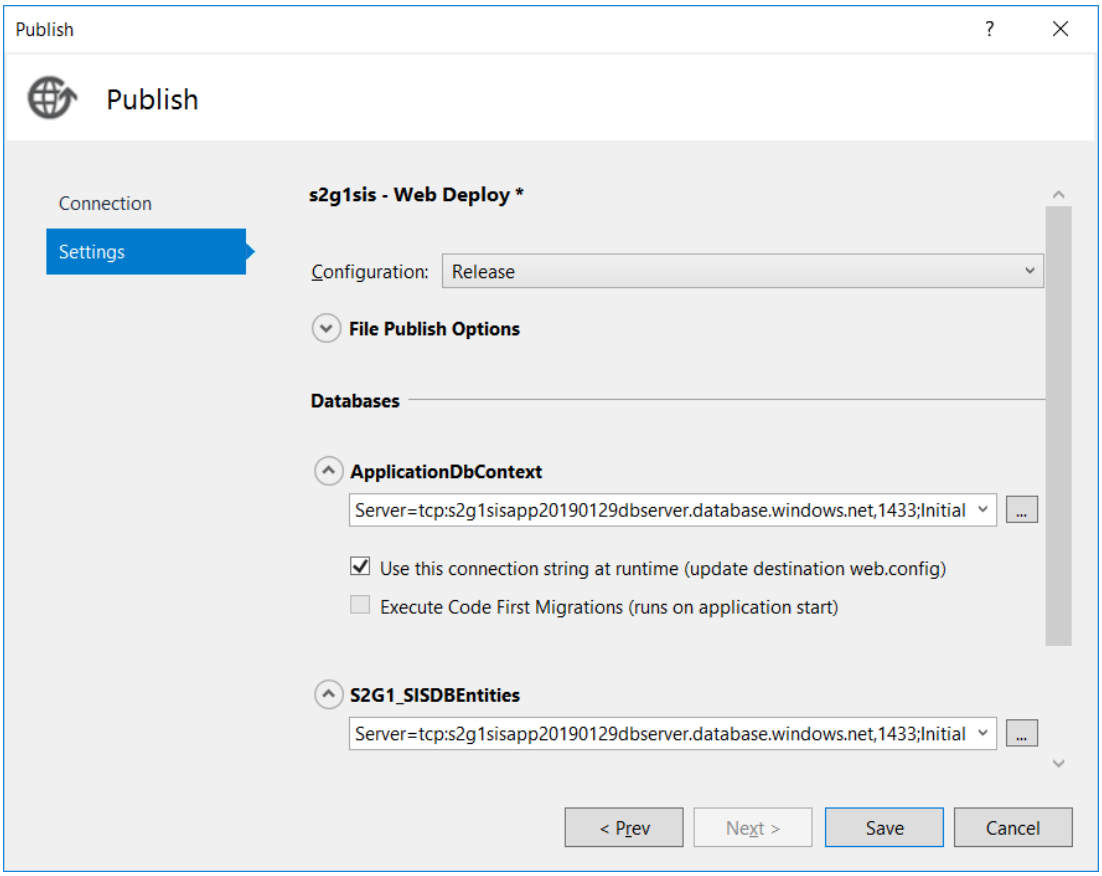


Right click on s2g1sis/Tables->Add New Table, paste the content of the .sql script file generated from your local DB, and then you can see the newly created tables in your Azure DB.

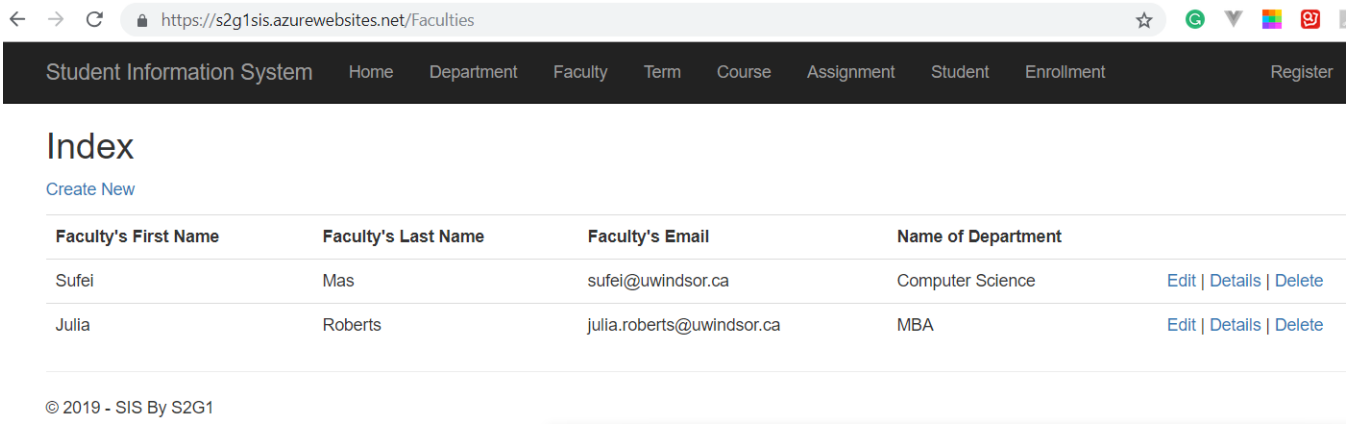


Copy the connection string from Azure DB portal and put it into VS: Publish->Configure. Do not forget to substitute the username and password with the real values you have created in the Azure DB setup wizard.





Now you can operate your Azure DB with the CRUD actions.



10. Reference

https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-a-more-complex-data-model-for-an-asp-net-mvc-application

https://docs.microsoft.com/en-ca/azure/app-service/app-service-web-get-started-dotnet-framework

https://docs.microsoft.com/en-ca/azure/app-service/app-service-web-tutorial-dotnet-sqldatabase#publish-to-azure-with-sql-database