MONASH University

FIT5032 - Internet Applications Development

WEEK 07A - USING ASP.NET IDENTITY

Last updated: 16th August 2018 and 24th August
Author: Jian Liew (and Yiwei Zhong for section 1)

Tutorial Objective

In this tutorial, you will need to form a 2 to 3-person group to understand the most common security threat on the internet and see the possible countermeasures about it. We will also cover ASP.Net Identity in this tutorial.

DoubtFire Submissions

T7.1 A document containing the notes of you and your classmate's discussion regarding the security vulnerabilities. (Need to be readable with proper headings) (in PDF document)

T7.2 Screenshots of ASP.Net Sign on (One for login page, one for after logging in, database details) (in PDF document)

1. Research the security threat

Please form a 3-person group, choose one of the area below and create a 2-3 slides PowerPoint (25 minutes), hint is given inside the bracket and share the class with your findings (Total maximum 45 minutes for all the groups).

Your tutor will correct your team's problem at the end of your presentation.

1.1 SQL Injection (Issue and its counter measure)
1.2 XSS (Issue and its counter measure)
1.3 MD5, SHA1 (Its weakness and what's the alternative)
1.4 Password Salting (A way to enhance password)
1.5 DDoS (Issue and its counter measure)
1.6 Session Hijacking (Issue and its counter measure)

2. USING ASP.NET IDENTITY

Introduction

This material showcases how to use MS Identity. In this simple example, it will showcase how by using MS Identity it would simplify the complex process of building your own authorization and authentication system. These days it is very rare for companies to roll their own authorization and authentication by reinventing the wheel.

Working with users is a very common use case. The objective of this supplementary material to give a quick introduction on how to use a MS Identity. The end product of this tutorial will demo how each user that logs on would be able to create their own entries to the database. **It will not show how to introduce roles to each users, however it can be also done easily.**

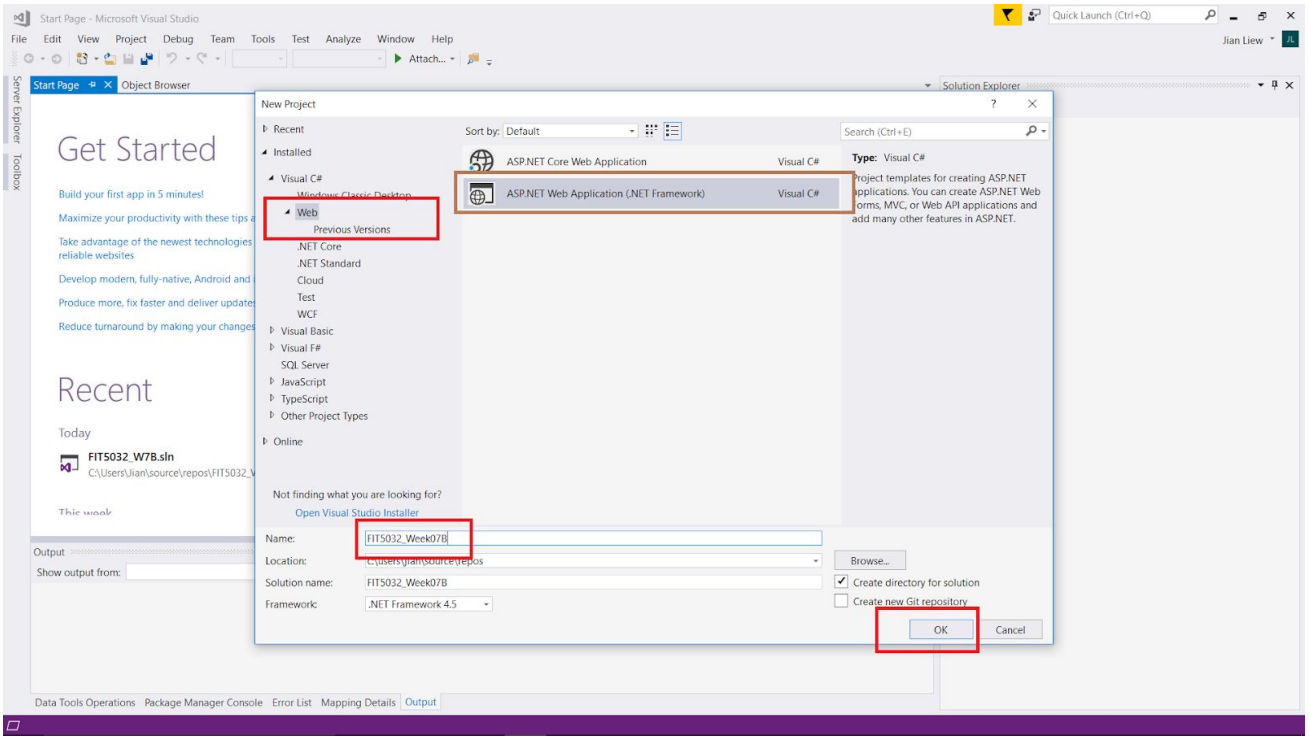
Objectives

Estimated Time To Complete - ~1 hours

Upon the completion of this tutorial, you will gain a basic understanding of

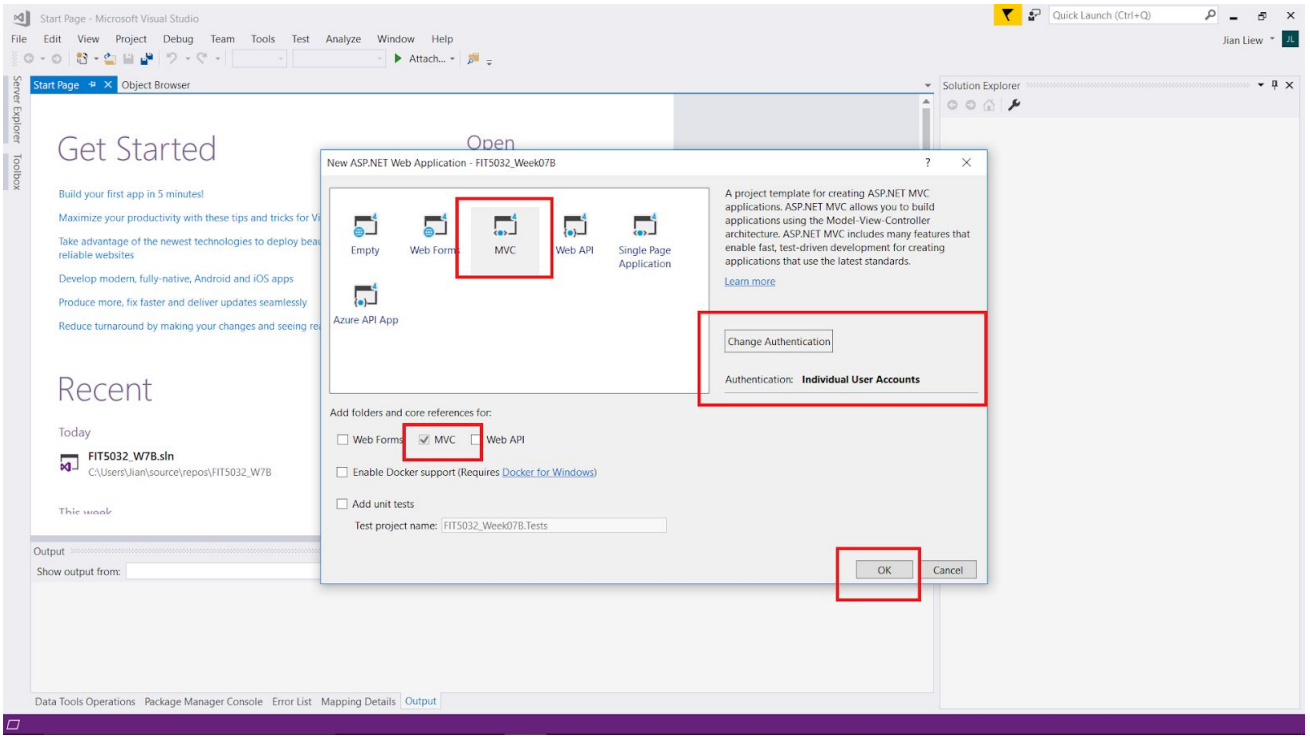
- How to use ASP.NET Identity at basic level
- How to use Attribute to Authorize controllers
- How to query information that belongs to a user.

Step 1



Step 2

I will create the Project with Individual User Accounts enabled.



Step 3

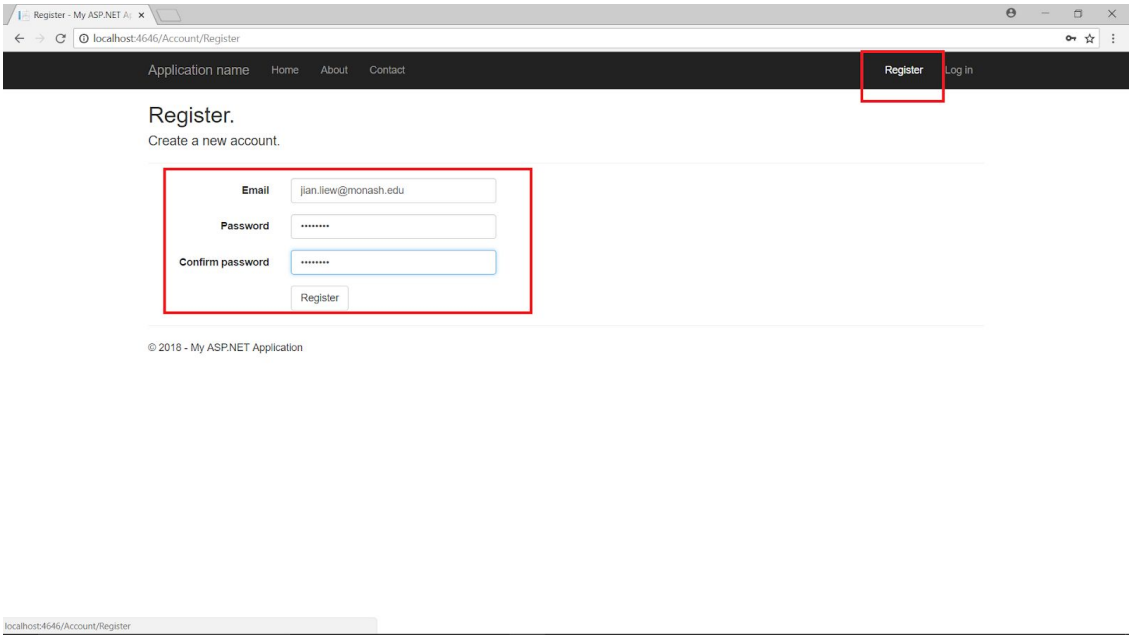
By default, the default project comes with Registration features, however it is done via a CodeFirst approach, so you will need to first Register an account.

I will go ahead and register an account by running the project.

I created my account with a super simple username and password

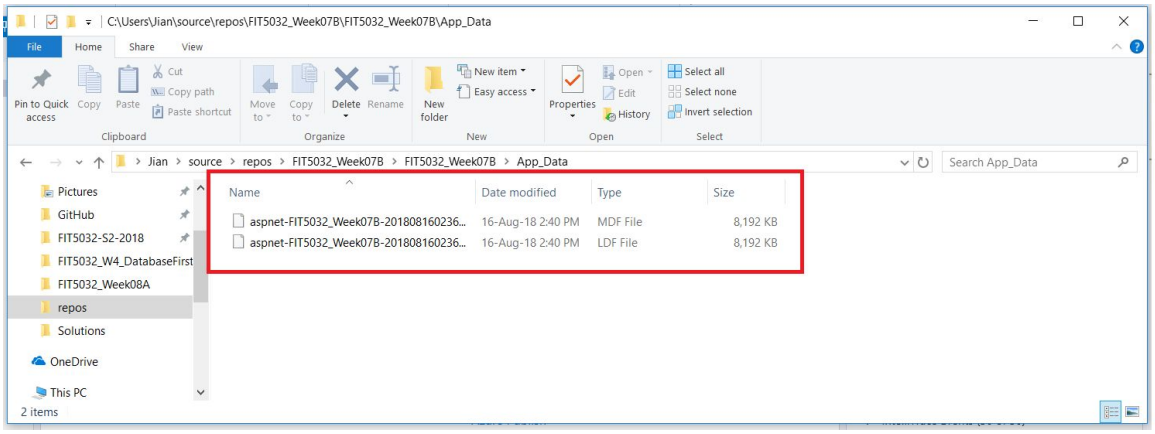
Email: jian.liew@monash.edu
Password: Potato1!

Click "Register"



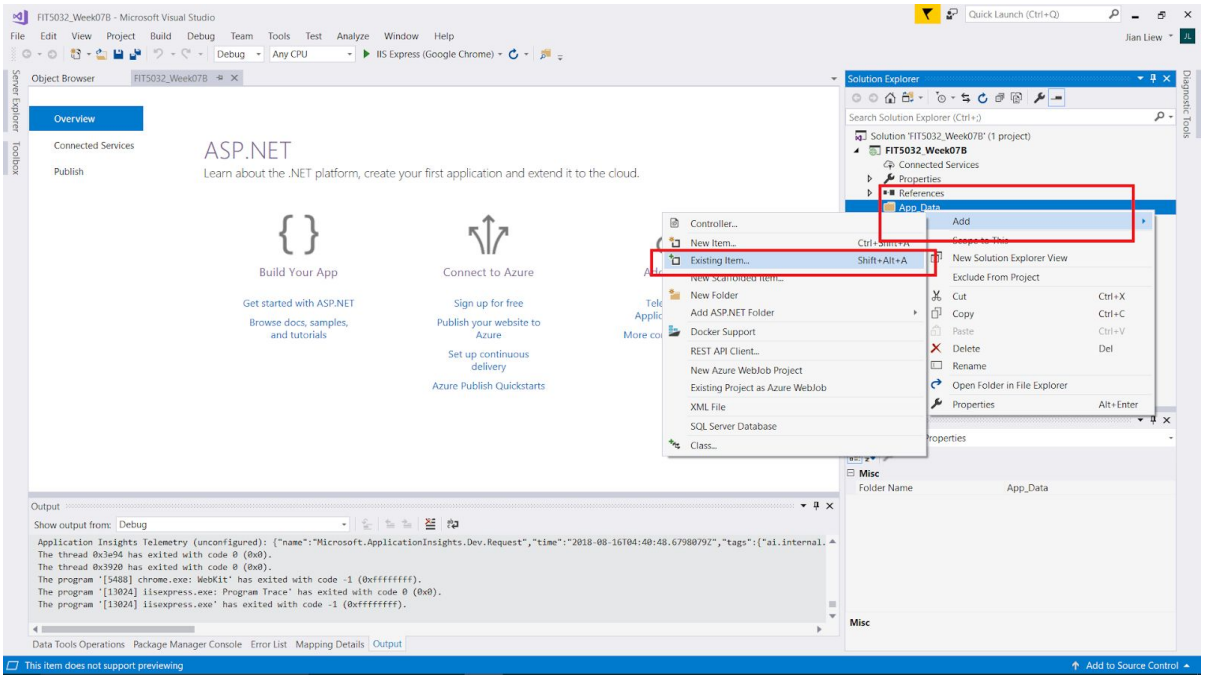
Step 4

After that, you can stop the application. By default the .mdf file should be created.



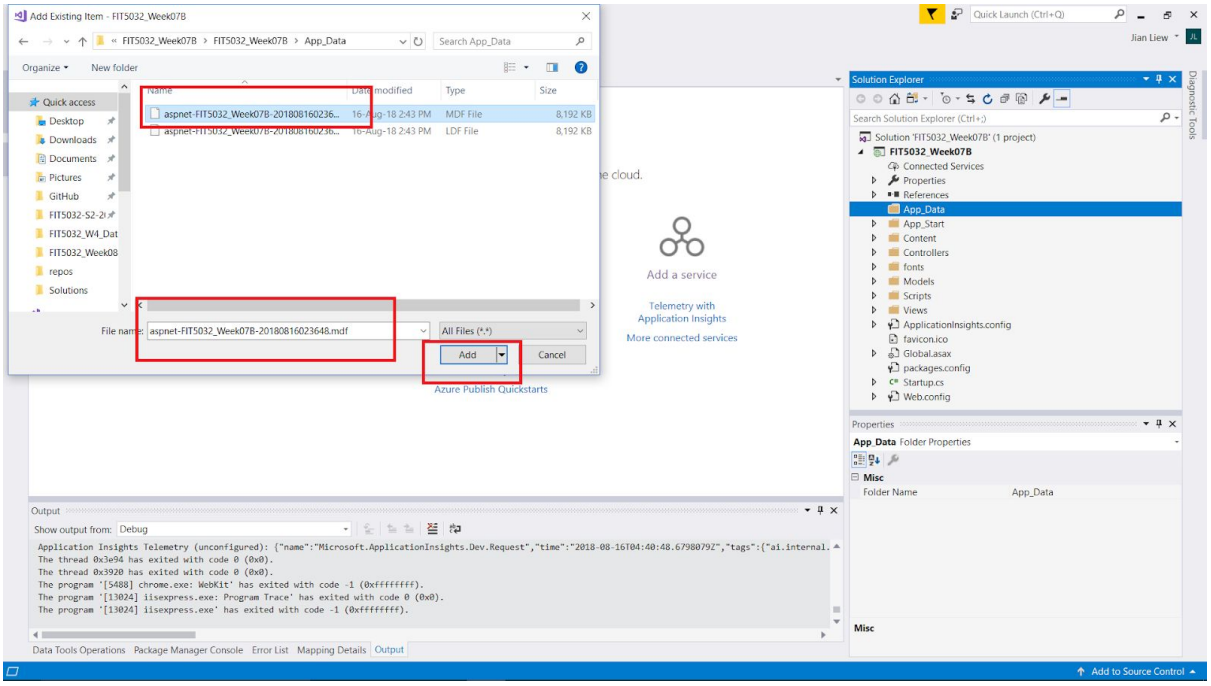
Step 5

Go ahead and add this Database to the Solution explorer as this is not done by default. This is done by Adding an Existing Item. Remember to put everything in the right place.



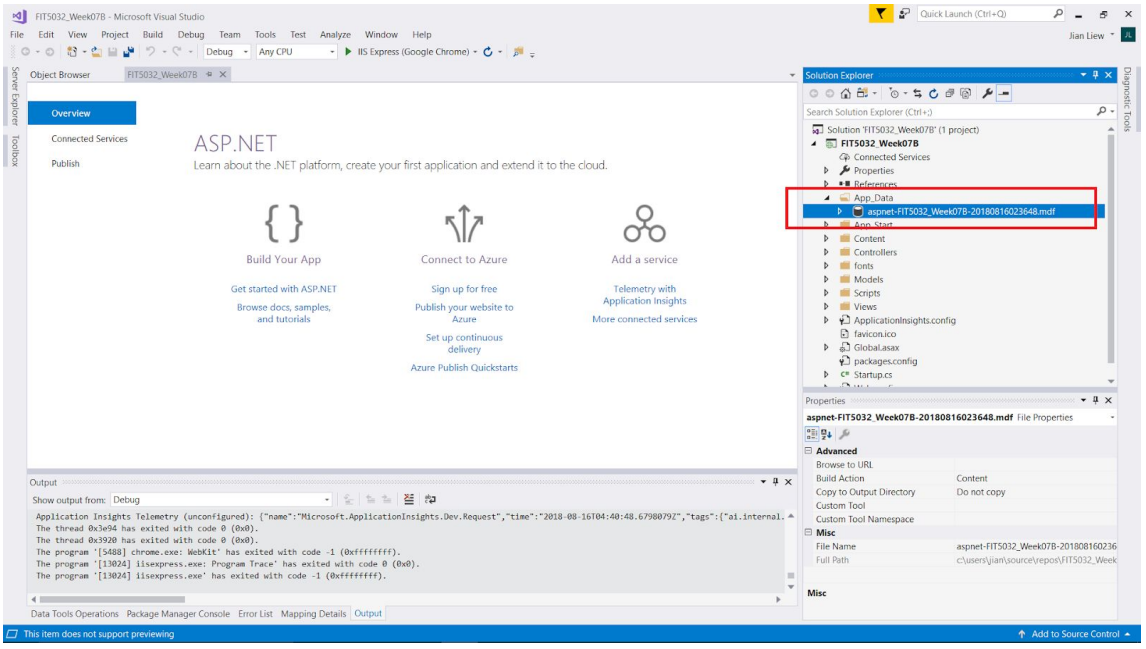
Step 6

Add the .mdf file. Your mdf file will have a different name. If you do not see the mdf file, you might need to register your user properly.

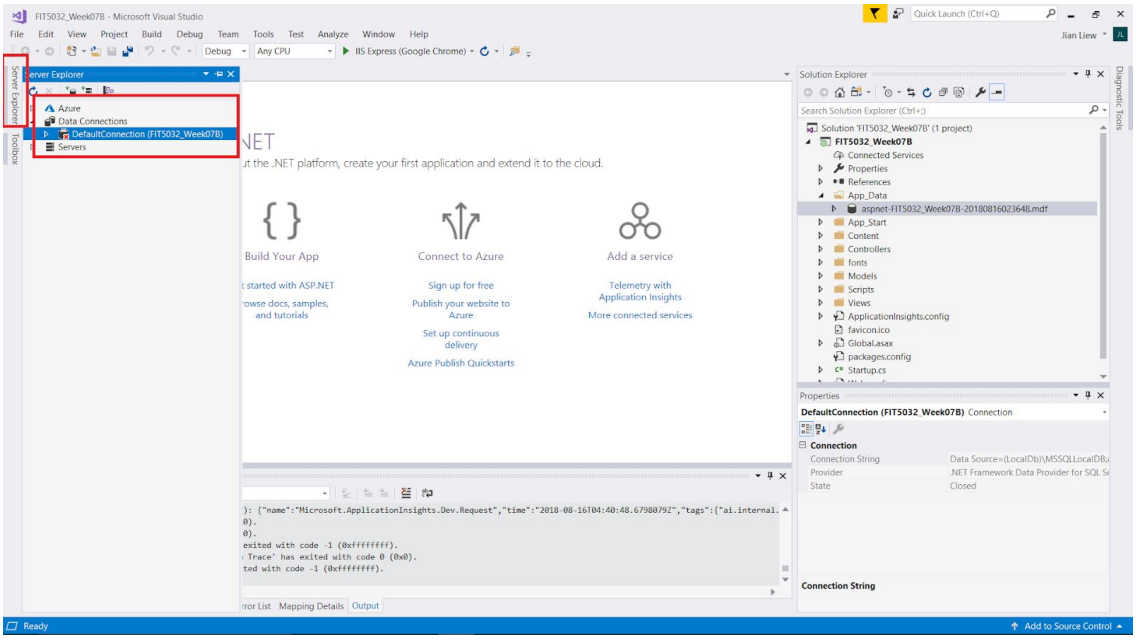


Step 7

The database should appear under the App_Data folder. This is your "Users" database. Normally it is good practice to have a different database for users and also for the data used in the business domain.

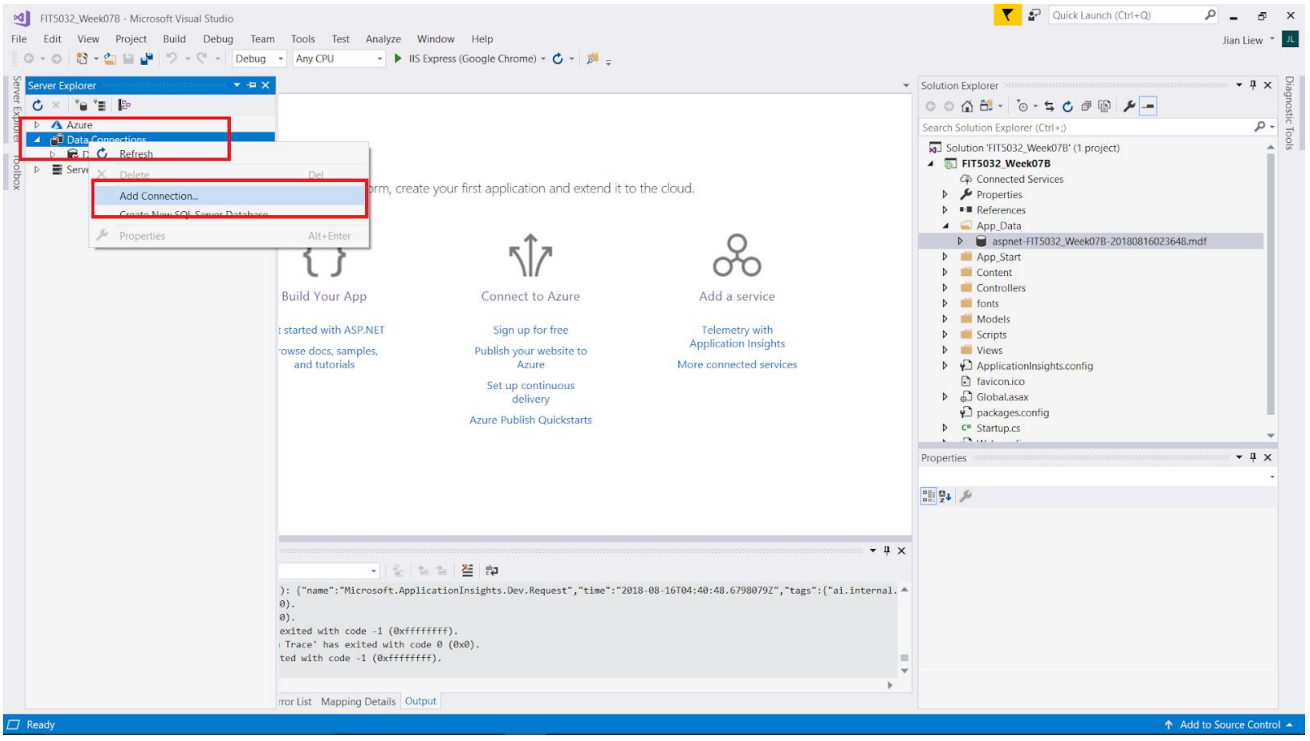


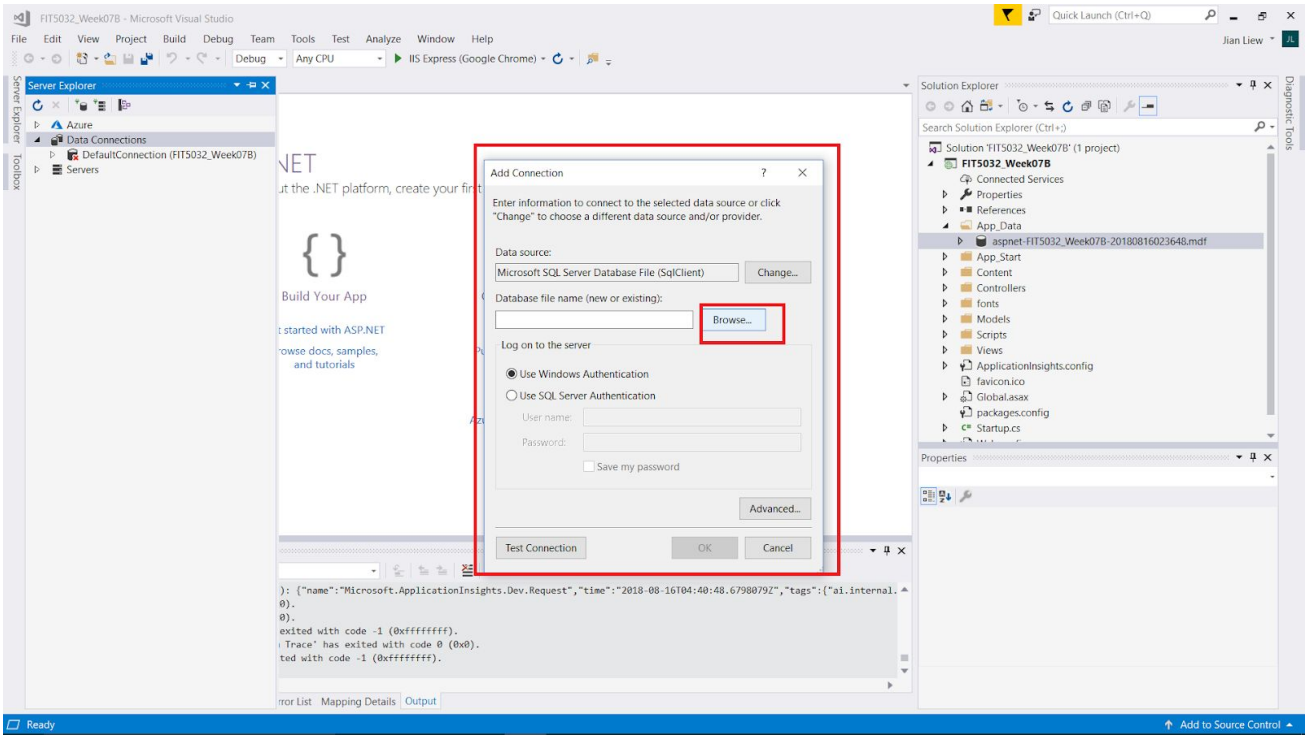
You will also realise that there will be a new "DefaultConnection" under the Data Connections however, I will proceed to add the "Users" database again to prevent IDE issues. The "Default Connection" is actually the connection to the Users database however, sometimes you do not see it. (Due to IDE issues).

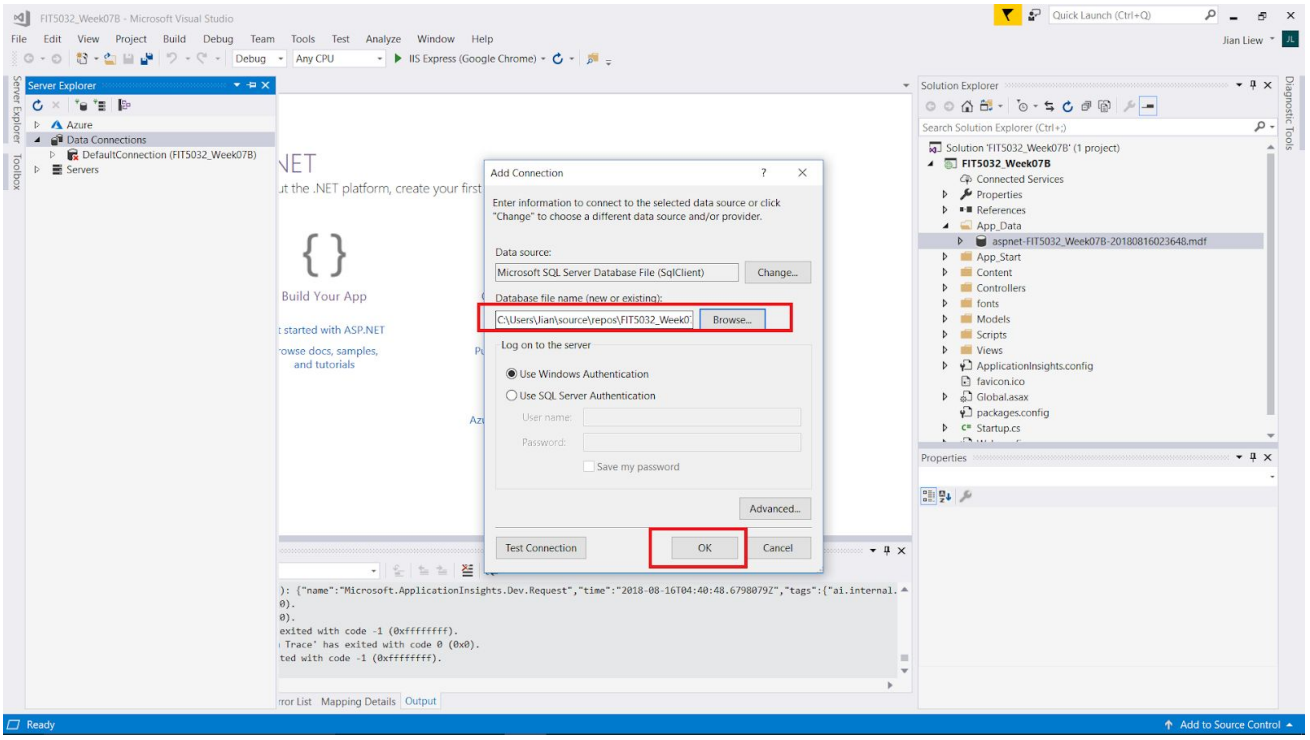


Step 8

Add a connection to the Users database to the Server Explorer. **If you see an error regarding that the database is in use, you can close the IDE and open it again. (Or you can close the database query window). The reason this is done, is to prevent issues later.**

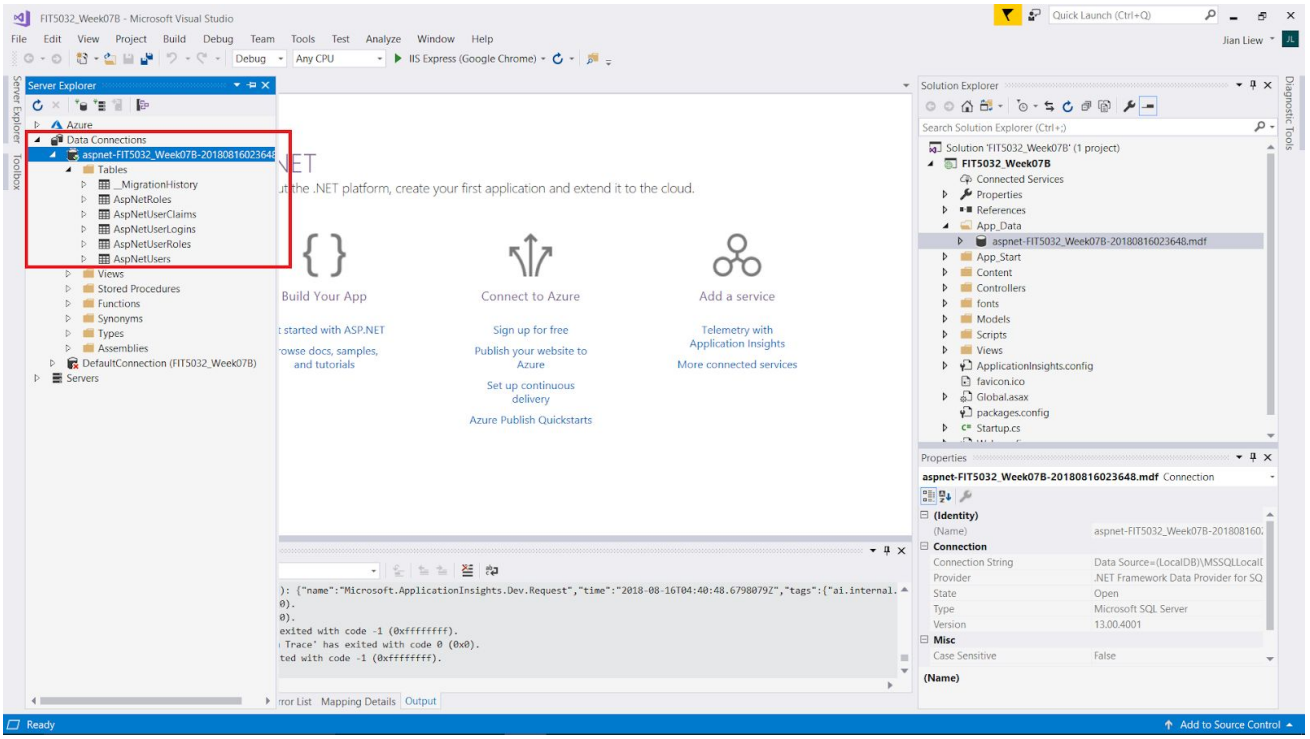




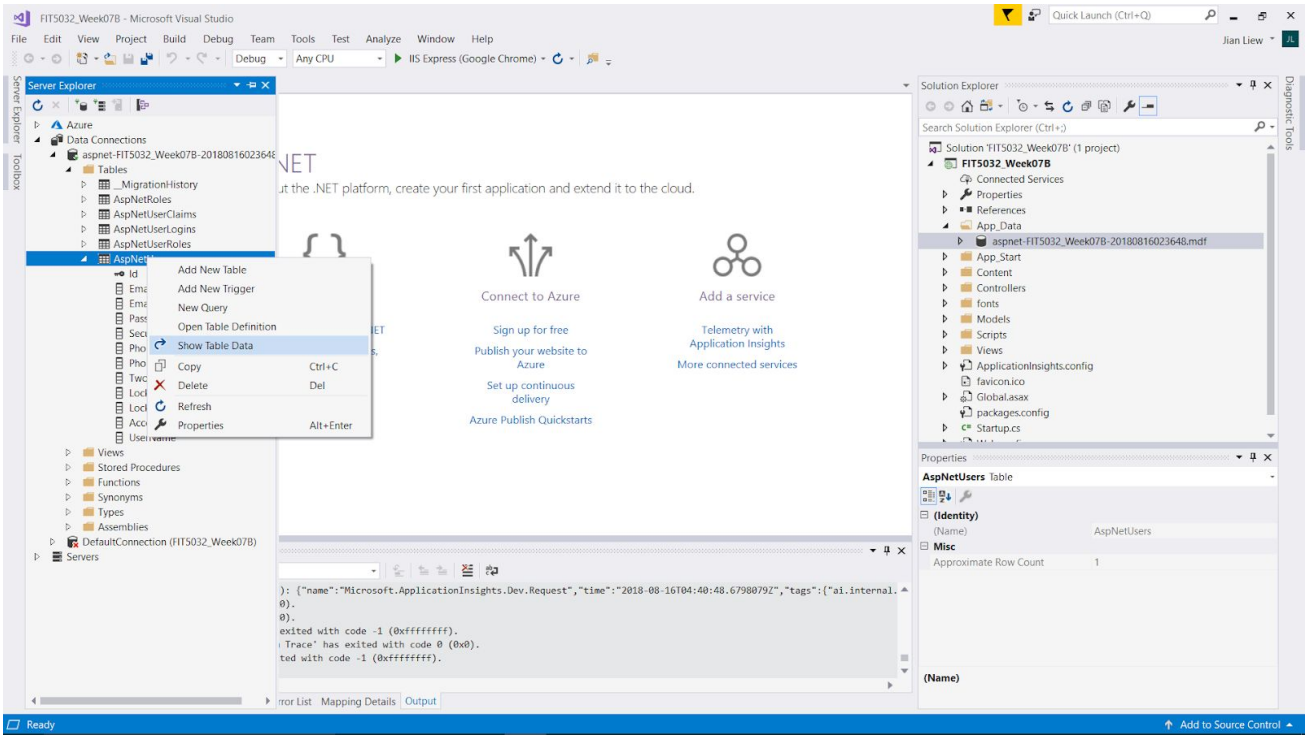


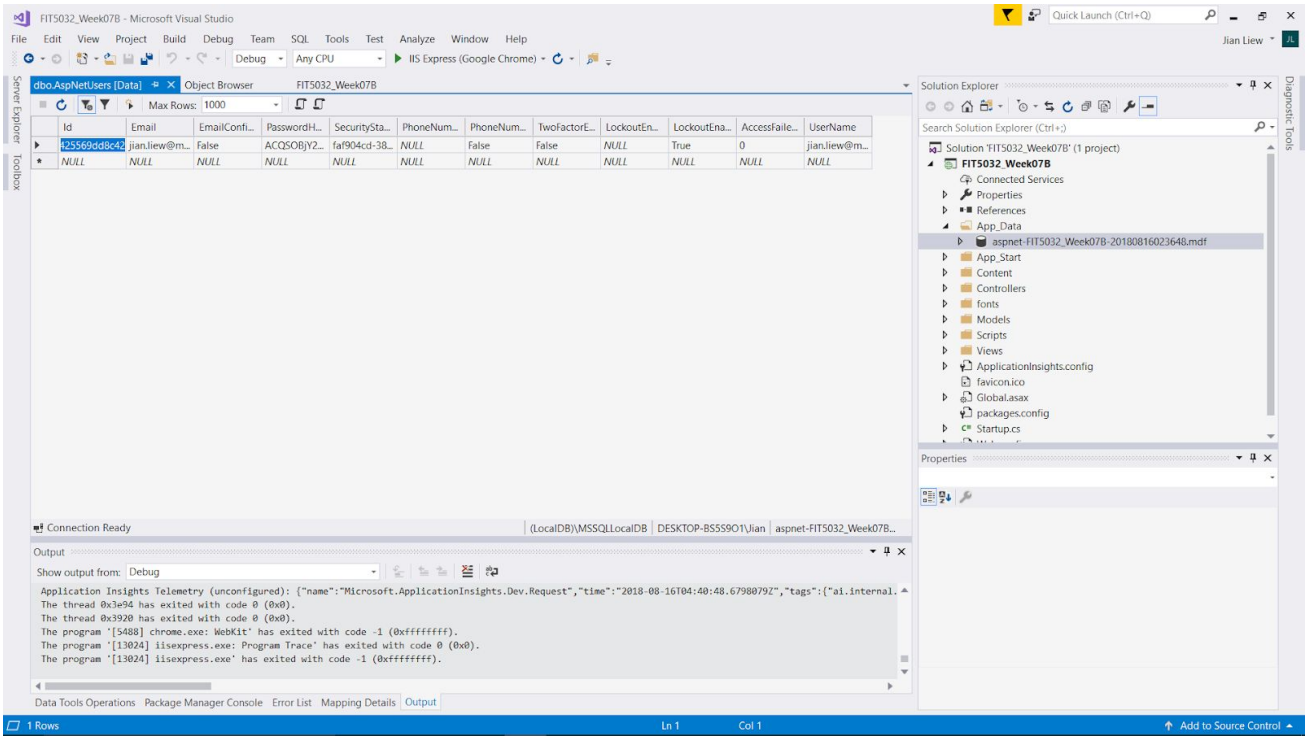
Step 9

You can now see the tables in our User Database. It is normally not a good idea to generate the edmx models from this structure as it would defeat the purpose of the CodeFirst approach used to design the authorization and authentication system.



Step 10


FIT5032_Week07B - Microsoft Visual Studio
Server Explorer
Add New Table
Add New Trigger
New Query
Open Table Definition
Show Table Data
Copy Ctrl+C
Delete Del
Refresh
Properties Alt+Enter
Connect to Azure
Add a service
Solution Explorer
Properties
AspNetUsers Table
Ready
Add to Source Control


FIT5032_Week07B - Microsoft Visual Studio
dbo.AspNetUsers [Data]
Object Browser
Max Rows: 1000
Connection Ready
Output
Application Insights Telemetry (unconfigured): {"name":"Microsoft.ApplicationInsights.Dev.Request","time":"2018-08-16T04:40:48.6798079Z","tags":{"ai.internal.
The thread 0x3e94 has exited with code 0 (0x0).
The thread 0x3920 has exited with code 0 (0x0).
The program '[5488] chrome.exe: WebKit' has exited with code -1 (0xffffffff).
The program '[13024] iisexpress.exe: Program Trace' has exited with code 0 (0x0).
The program '[13024] iisexpress.exe' has exited with code -1 (0xffffffff).
1 Rows
Ln 1
Col 1
Add to Source Control

Step 11

You can make use of this "Id" in which Microsoft Identity creates for you.

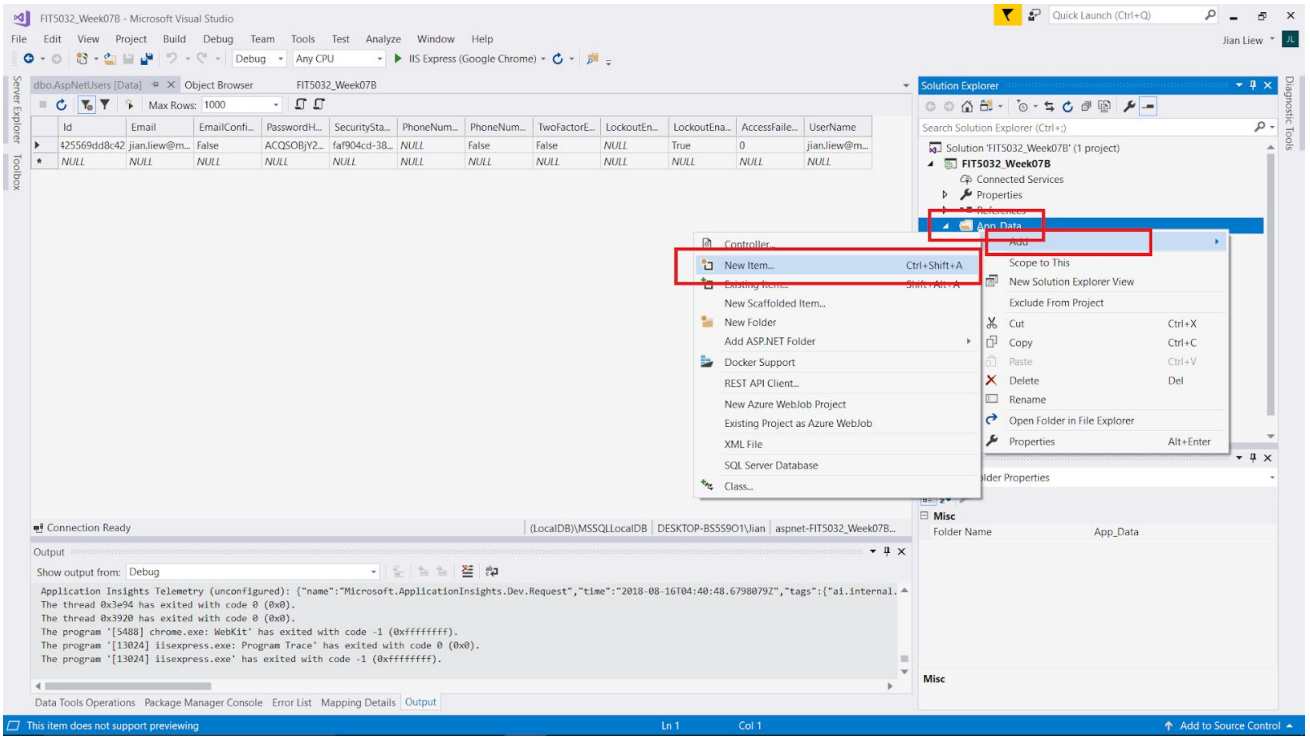
The CurrentUserId can be obtained via

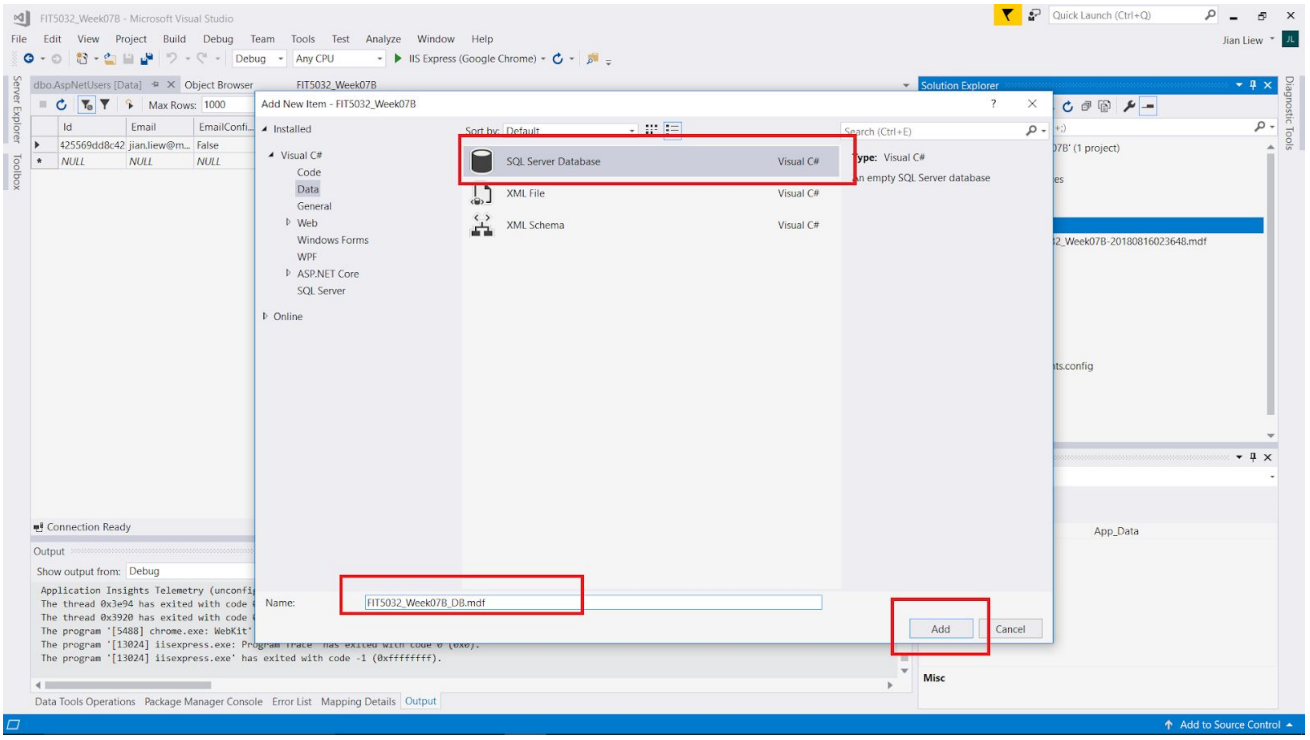
```
using Microsoft.AspNet.Identity;

string userId = User.Identity.GetUserId();
```

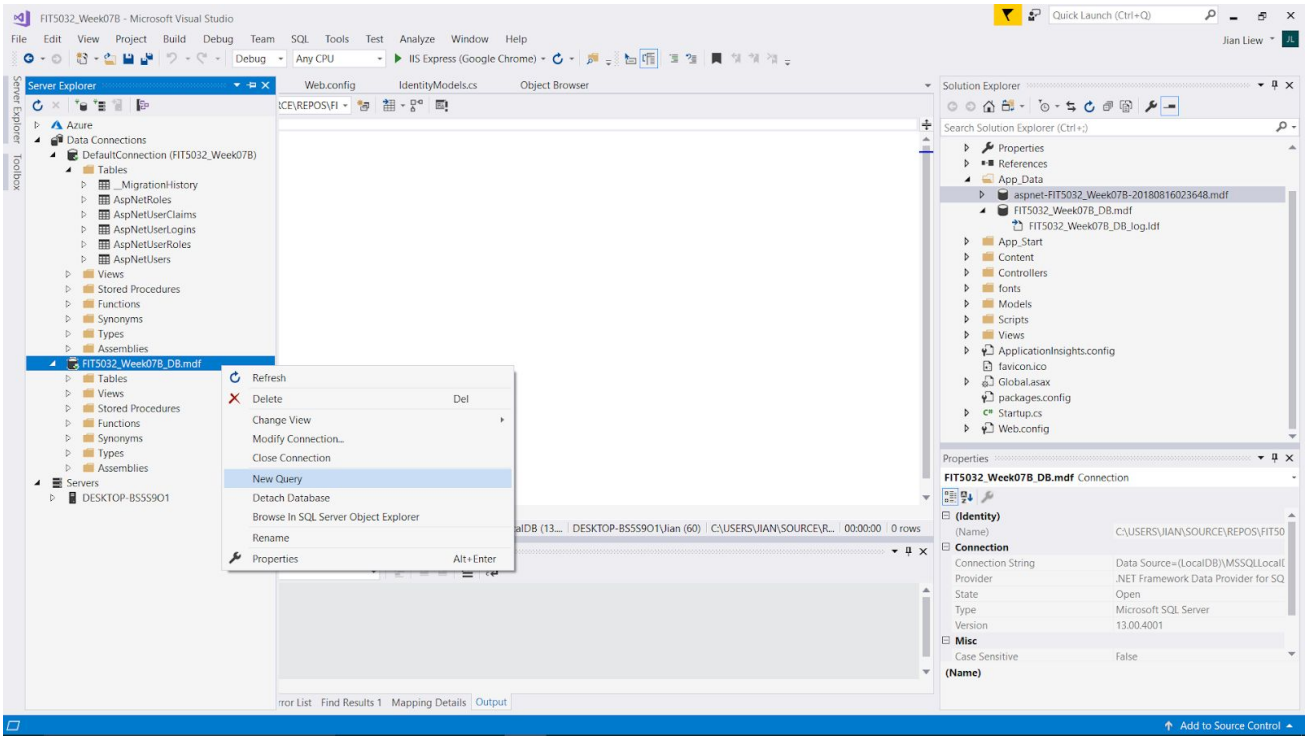
Step 12

Now I will go ahead and create another Database. The reason I am doing this is because I want a separate database for my authentication and authorization and a different database my business information. This is quite common practice. So, I actually have **2 databases or 2 mdf files.**

I will create another database with a very similar structure to what I have done on Week 4 but I will introduce another column to store the UserId. This UserId **will not** have any reference integrity across to my authorization and authentication database.



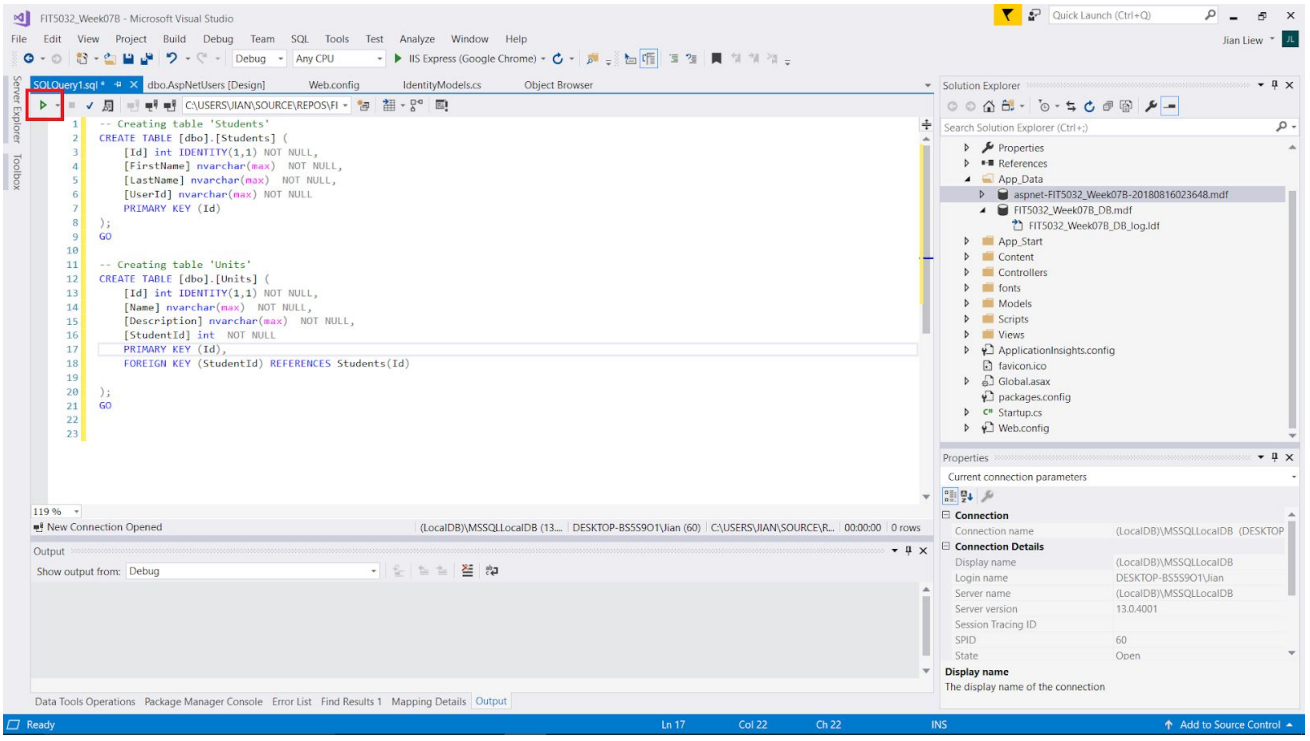

FIT5032_Week07B - Microsoft Visual Studio
Add New Item - FIT5032_Week07B
Installed
Visual C#
Code
Data
General
Web
Windows Forms
WPF
ASP.NET Core
SQL Server
Online
SQL Server Database
XML File
XML Schema
Name:
FIT5032_Week07B_DB.mdf
Add
Cancel
Solution Explorer
Connection Ready
Output
Data Tools Operations
Package Manager Console
Error List
Mapping Details
Output
Add to Source Control


FIT5032_Week07B - Microsoft Visual Studio
Server Explorer
Azure
Data Connections
DefaultConnection (FIT5032_Week07B)
Tables
Views
Stored Procedures
Functions
Synonyms
Types
Assemblies
FIT5032_Week07B_DB.mdf
Servers
DESKTOP-BSSS9O1
Refresh
Delete
Change View
Modify Connection...
Close Connection
New Query
Detach Database
Browse In SQL Server Object Explorer
Rename
Properties
Solution Explorer
App_Data
aspnet-FIT5032_Week07B-20180816023648.mdf
FIT5032_Week07B_DB.mdf
FIT5032_Week07B_DB_log.ldf
App_Start
Content
Controllers
fonts
Models
Scripts
Views
ApplicationInsights.config
favicon.ico
Global.asax
packages.config
Startup.cs
Web.config
Properties
FIT5032_Week07B_DB.mdf Connection
(Identity)
Connection
Misc
Add to Source Control

Step 13

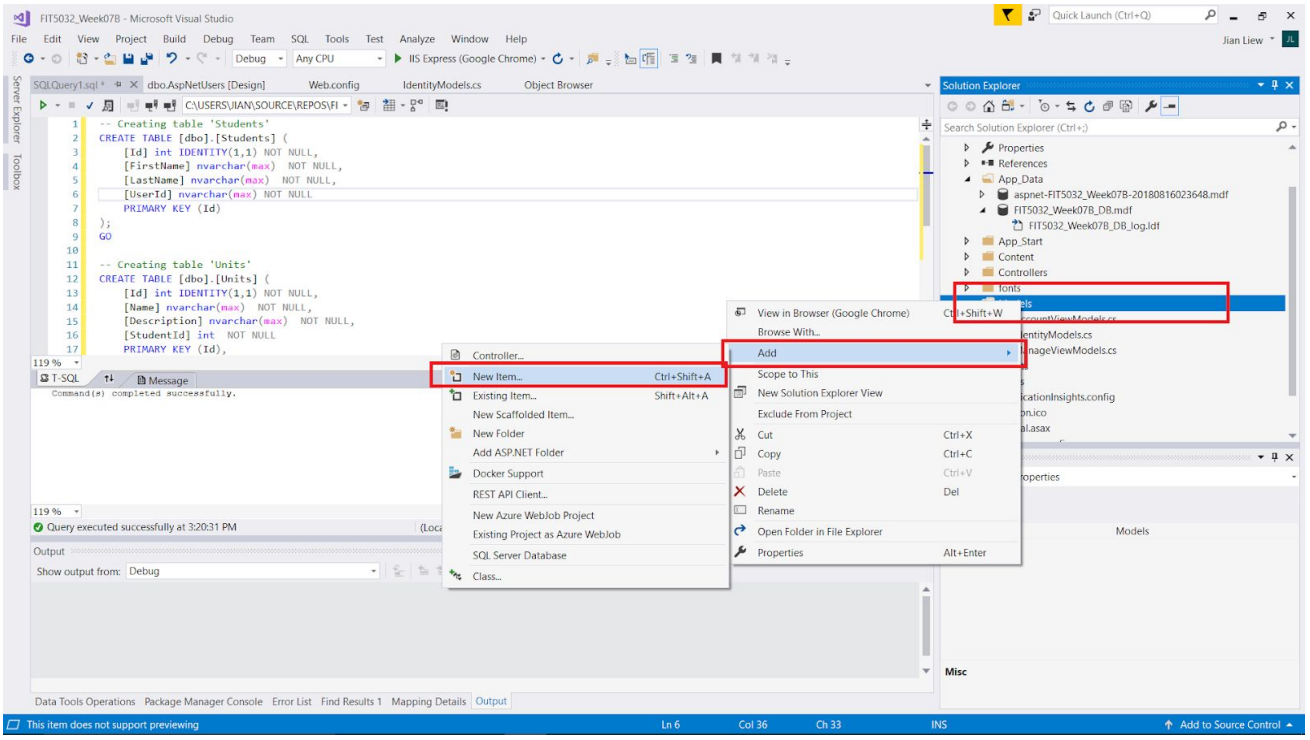
```
-- Creating table 'Students'
CREATE TABLE [dbo].[Students] (
      [Id] int IDENTITY(1,1) NOT NULL,
      [FirstName] nvarchar(max)  NOT NULL,
      [LastName] nvarchar(max)  NOT NULL,
      [UserId] nvarchar(max) NOT NULL
      PRIMARY KEY (Id)
);
GO

-- Creating table 'Units'
CREATE TABLE [dbo].[Units] (
      [Id] int IDENTITY(1,1) NOT NULL,
      [Name] nvarchar(max)  NOT NULL,
      [Description] nvarchar(max)  NOT NULL,
      [StudentId] int  NOT NULL
      PRIMARY KEY (Id),
      FOREIGN KEY (StudentId) REFERENCES Students(Id)

);
GO
```

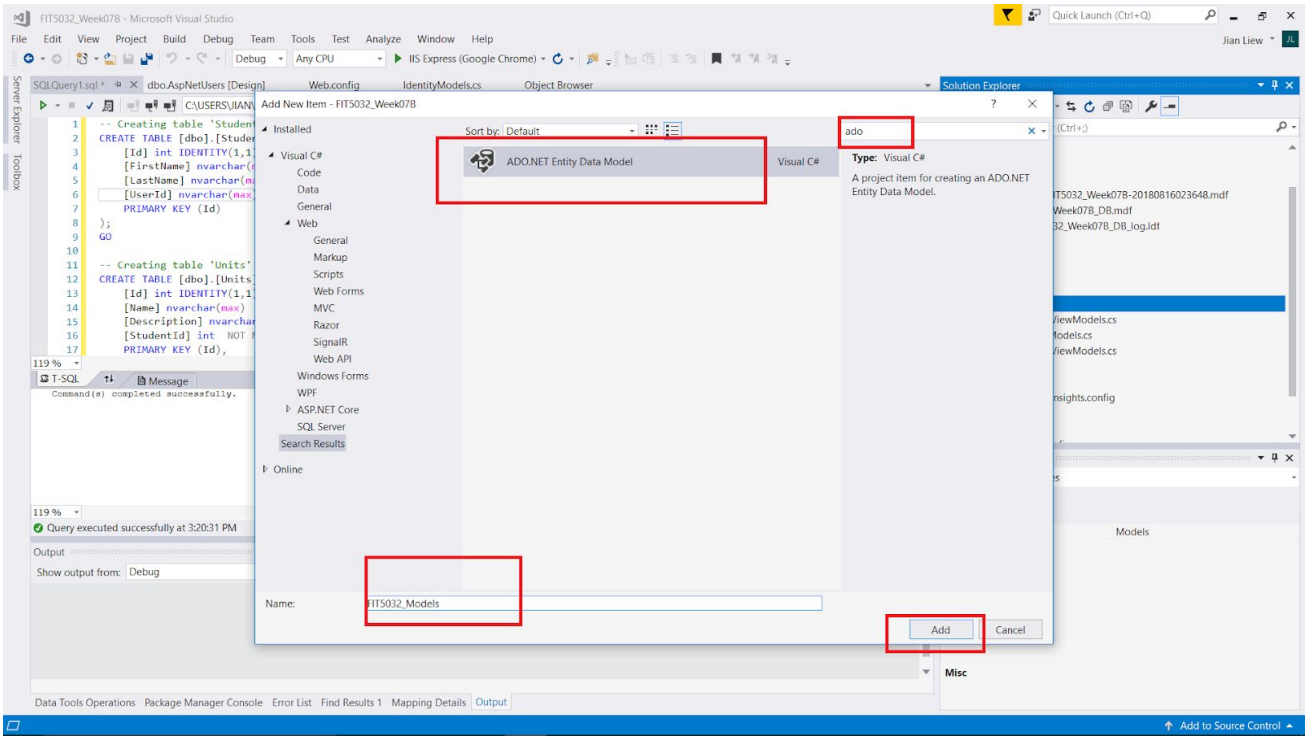
Step 14



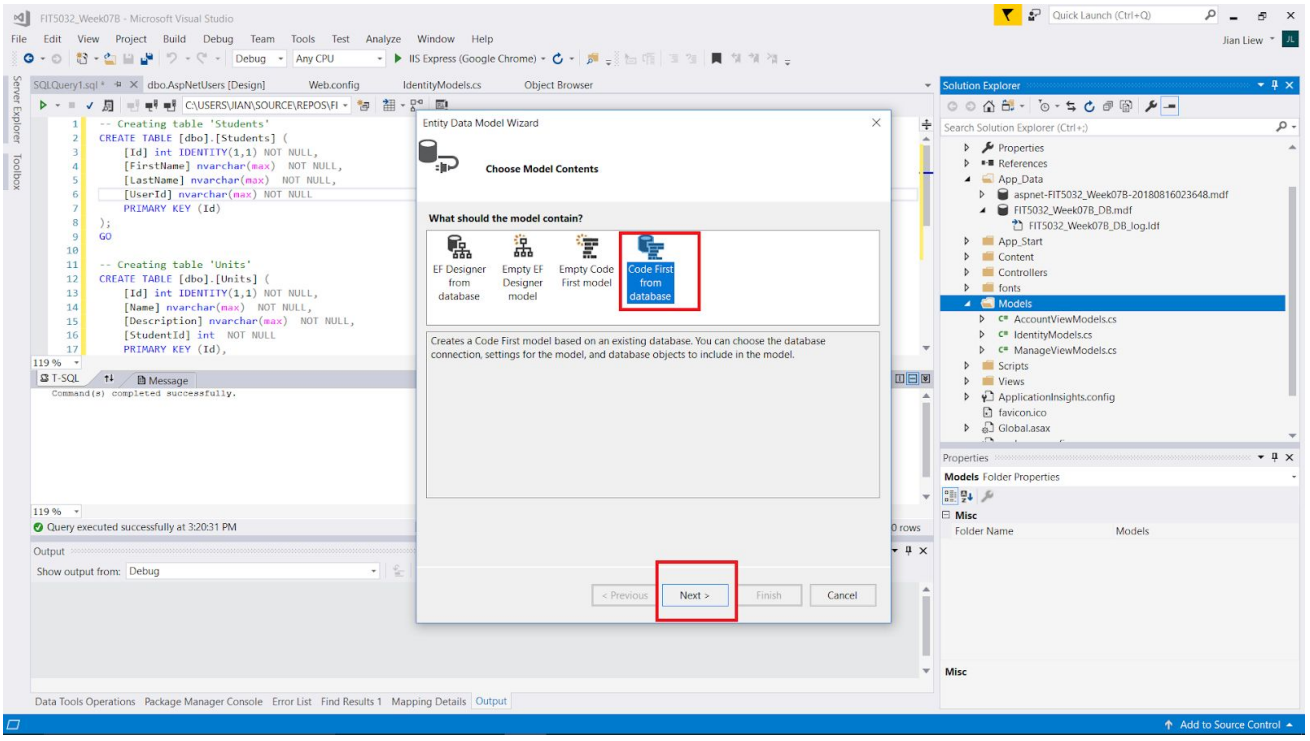
Step 15

After you have done that, you can now introduce your Models, Controllers & Views.

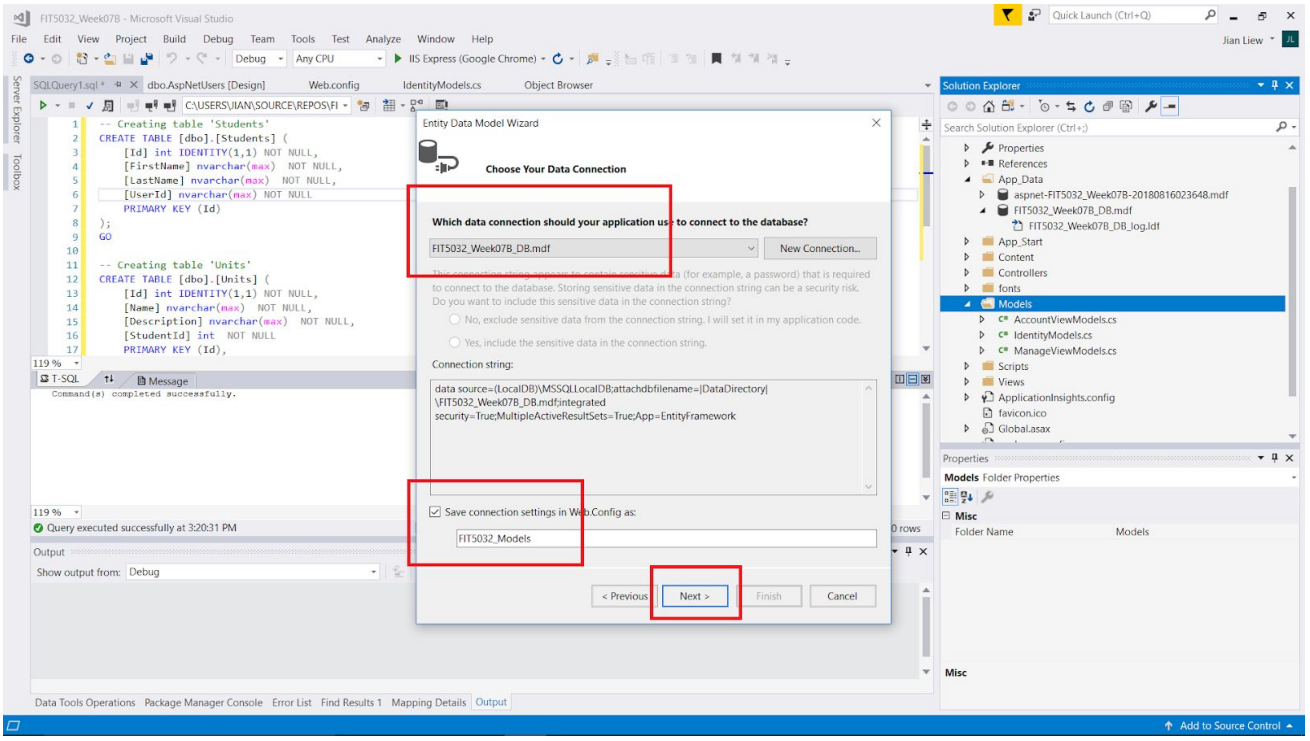




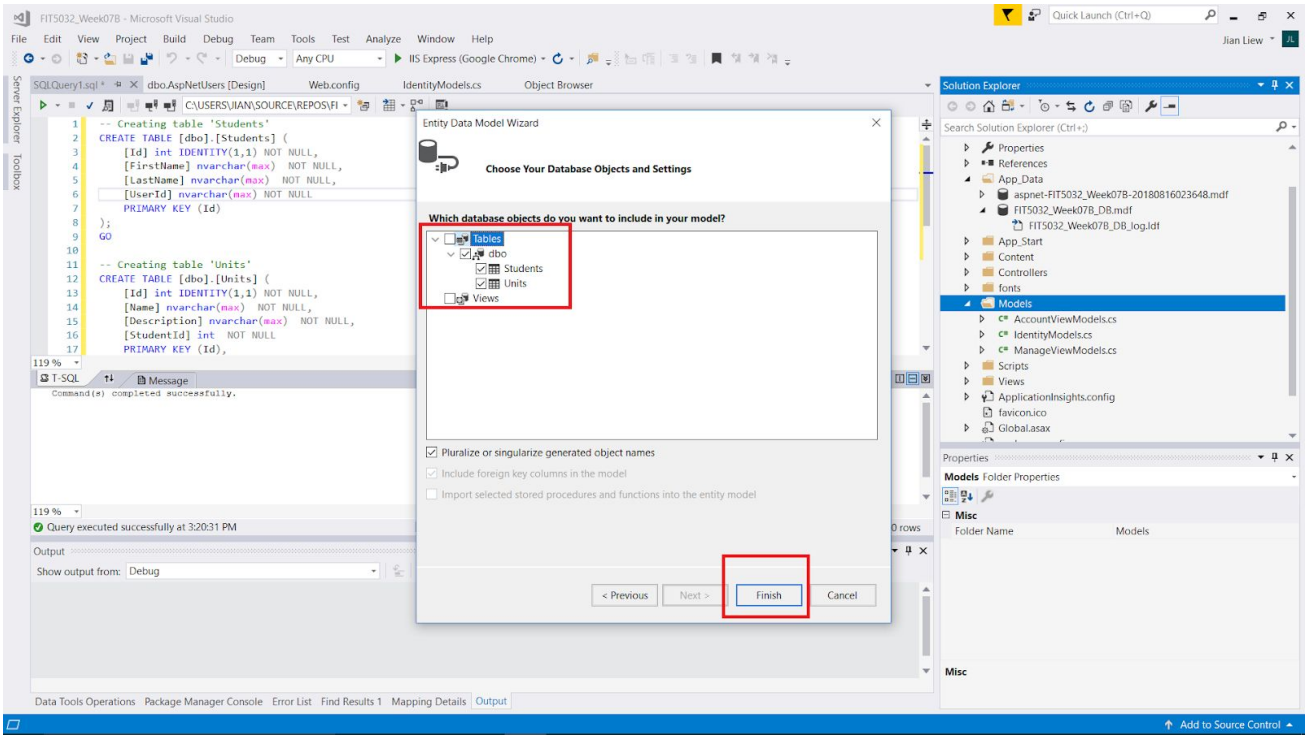
Step 16



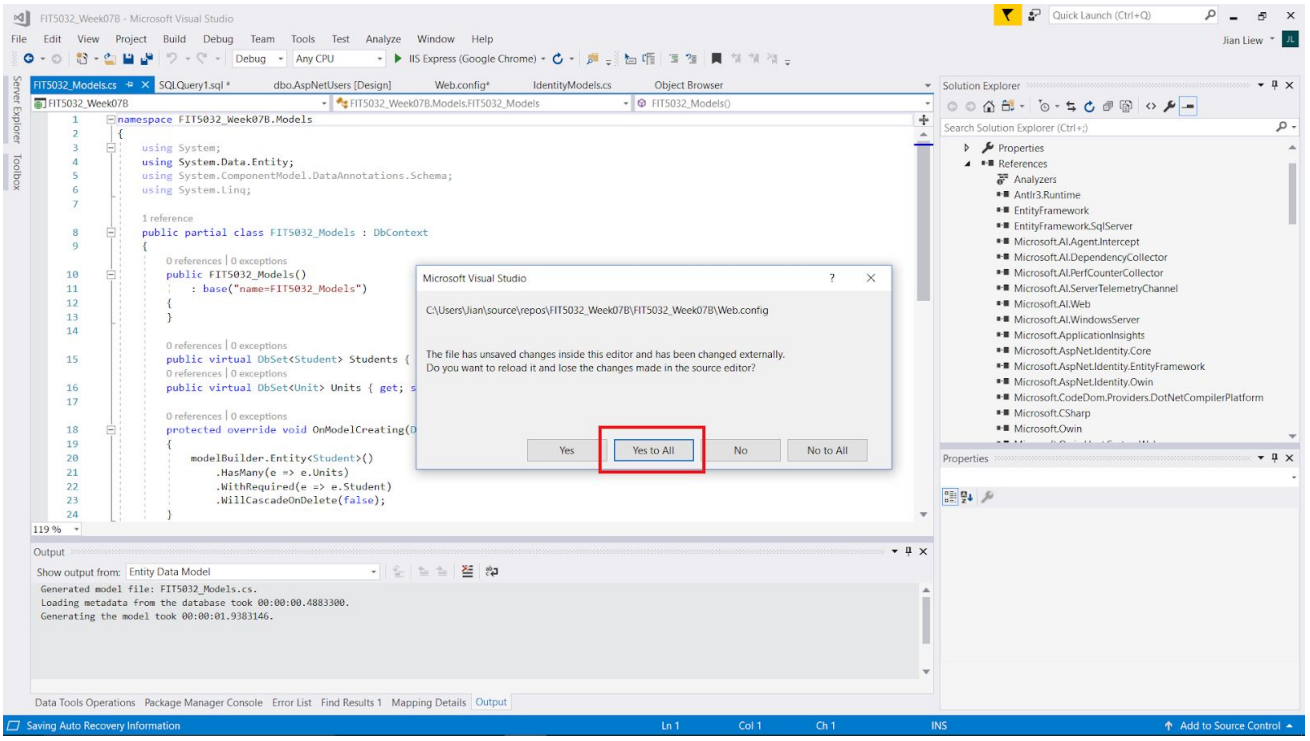
Please remember to select the correct data connection.



Step 17

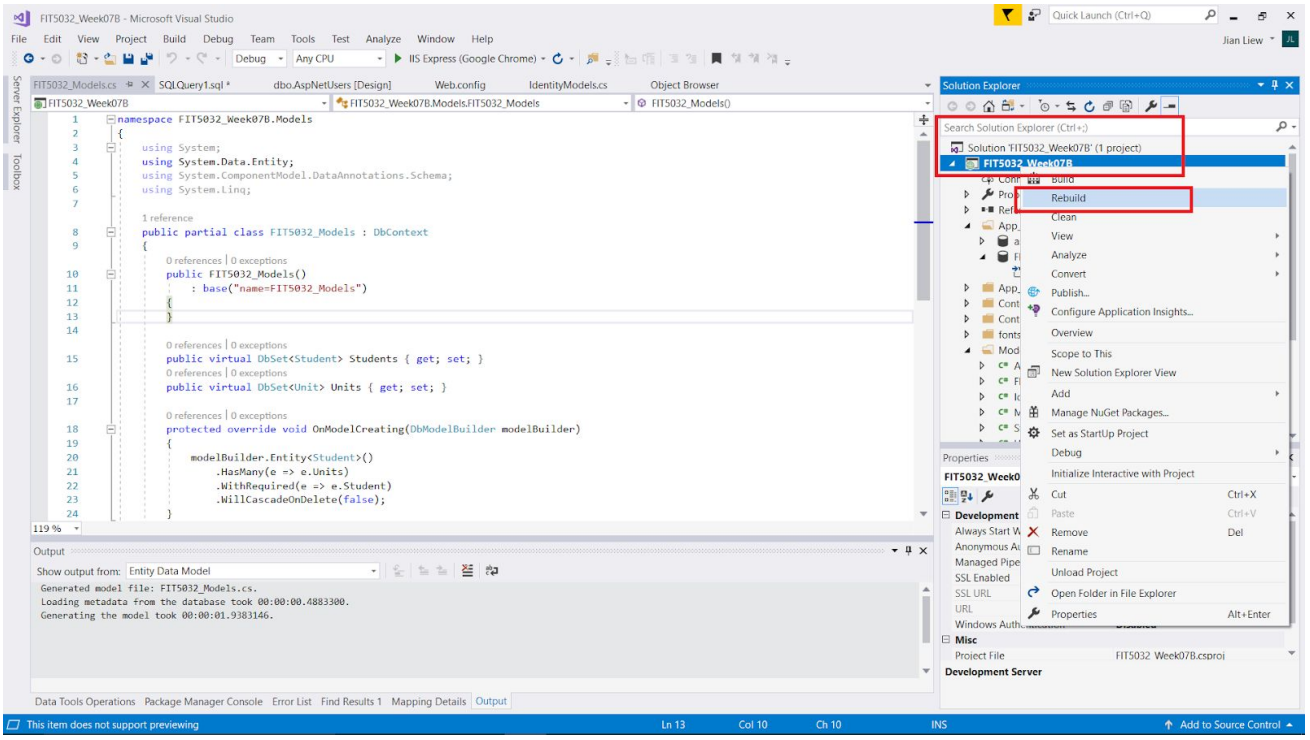


If you see any warnings just hit "Yes to All"



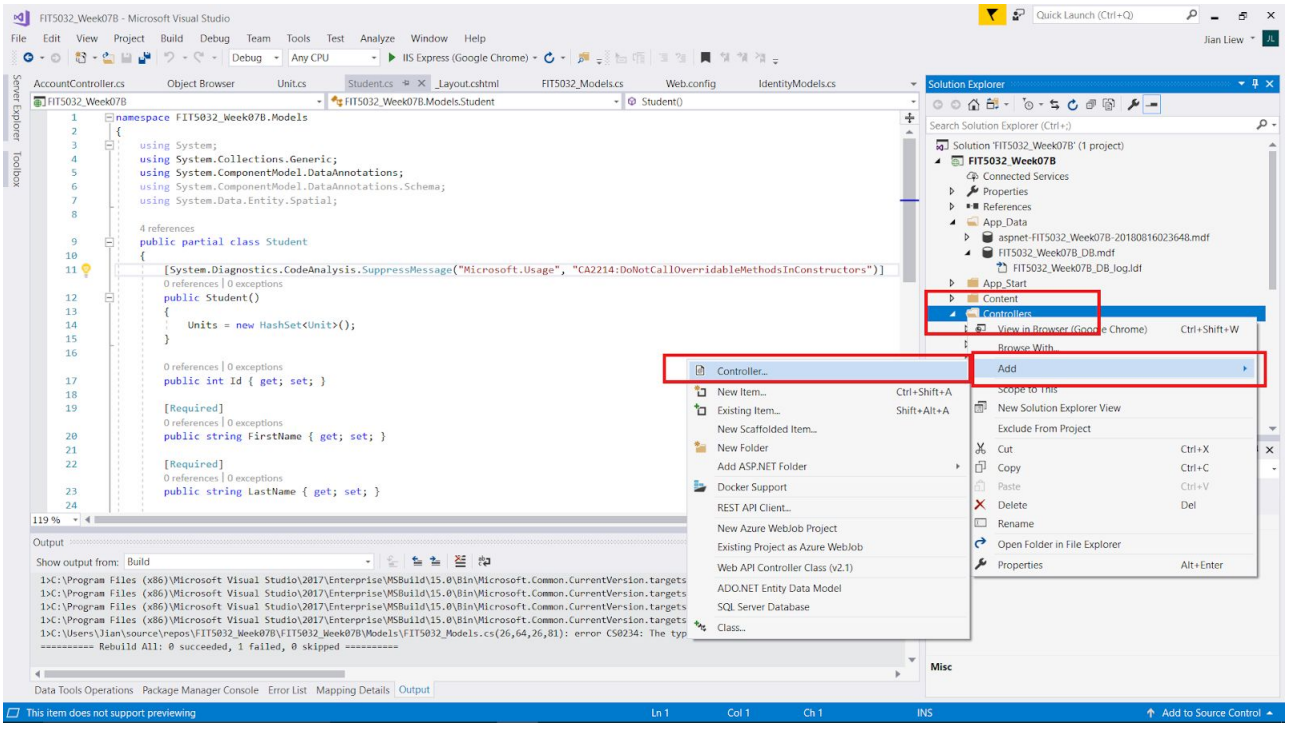
Step 18

After that, **please "Rebuild" your project.**

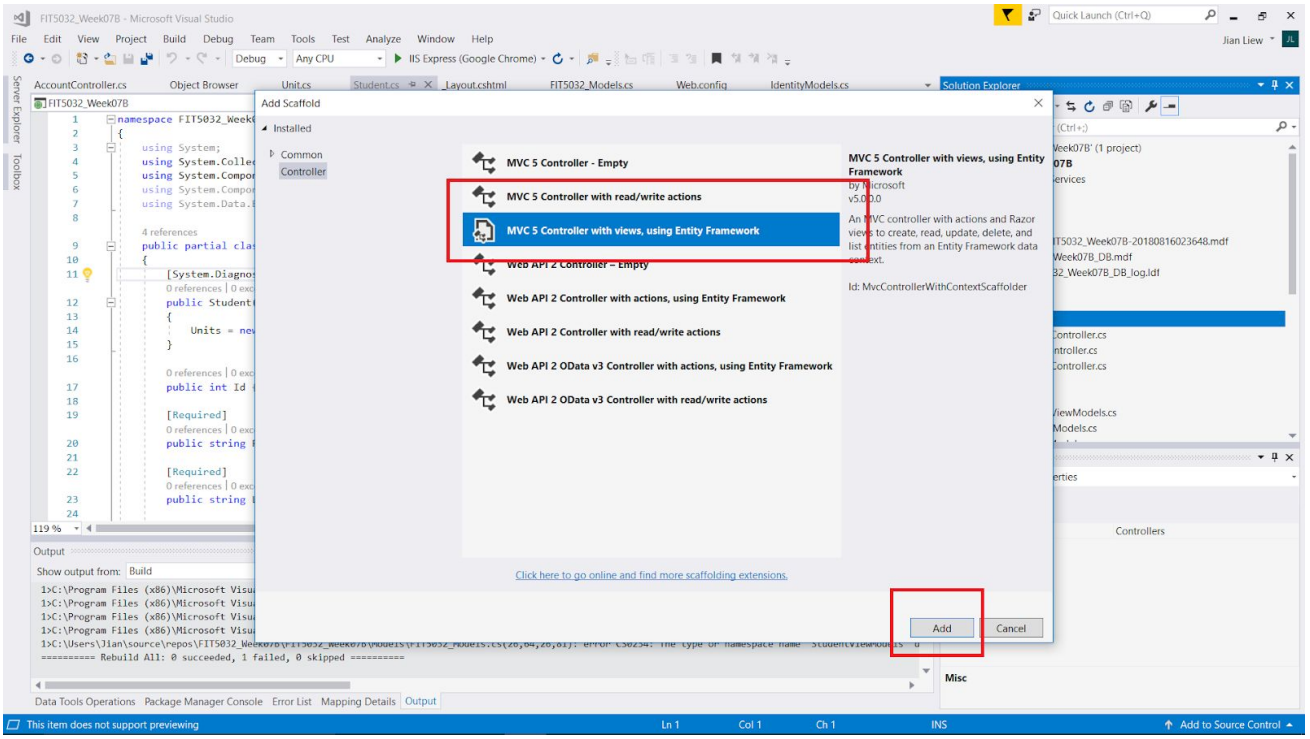


Step 14

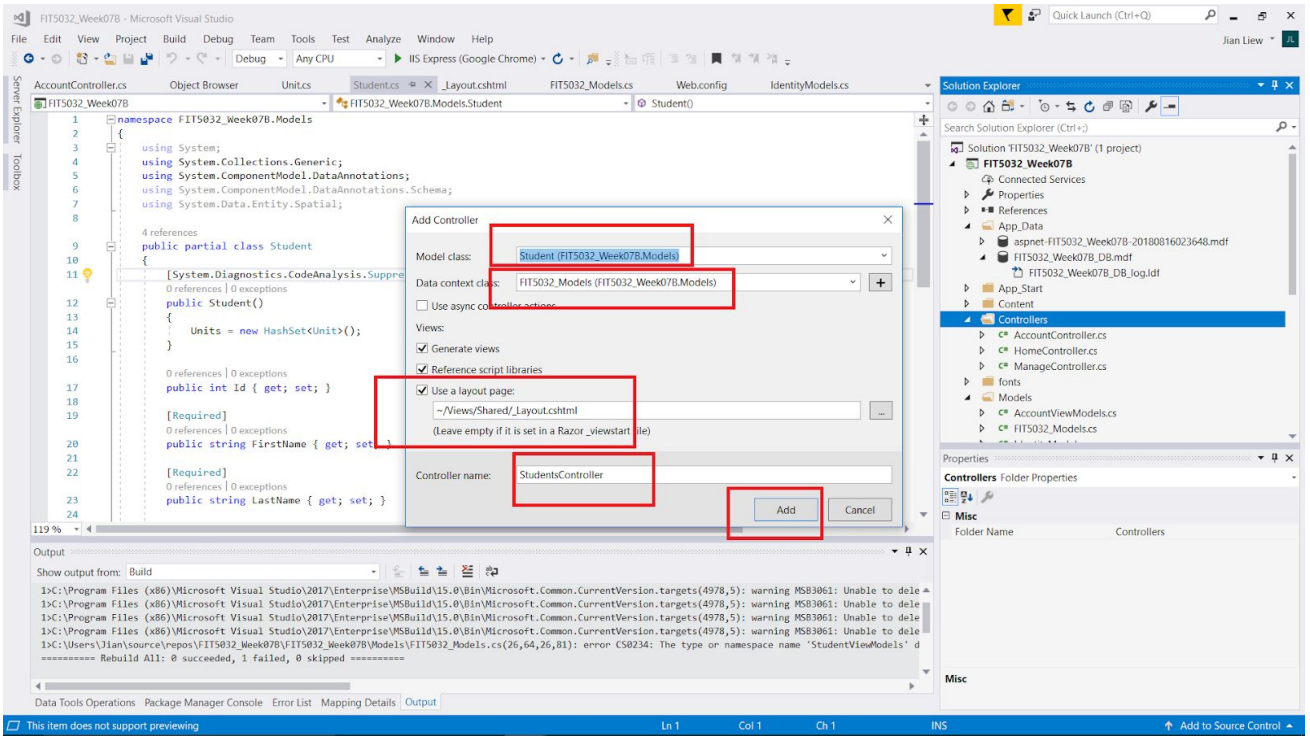
You can introduce more Attribute validations if you want to.



Step 19



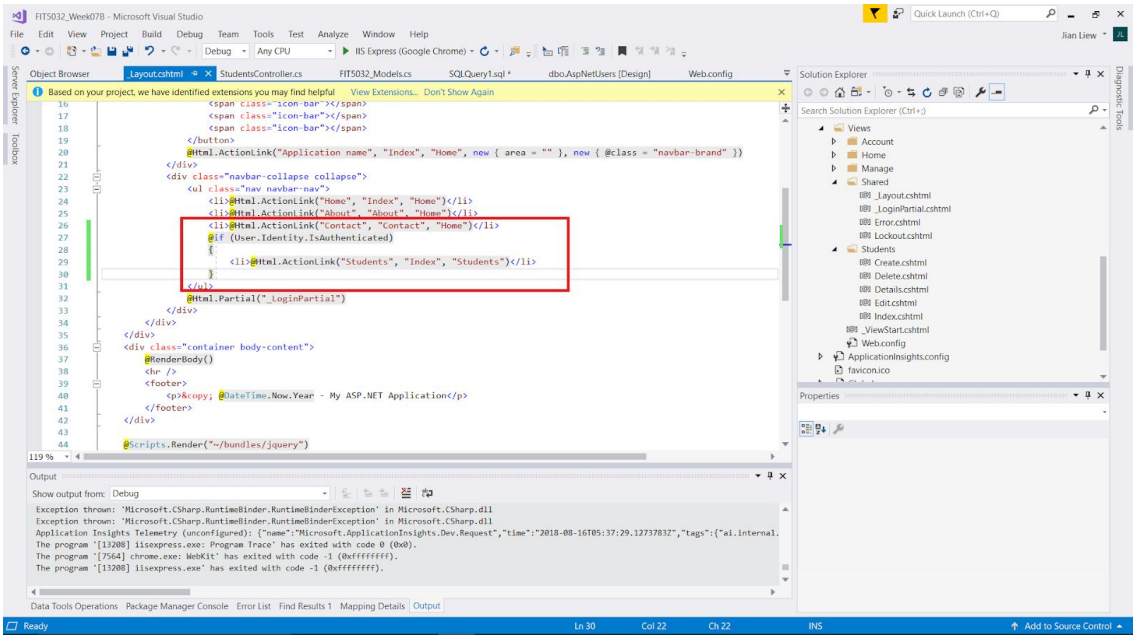
Step 20



Step 21

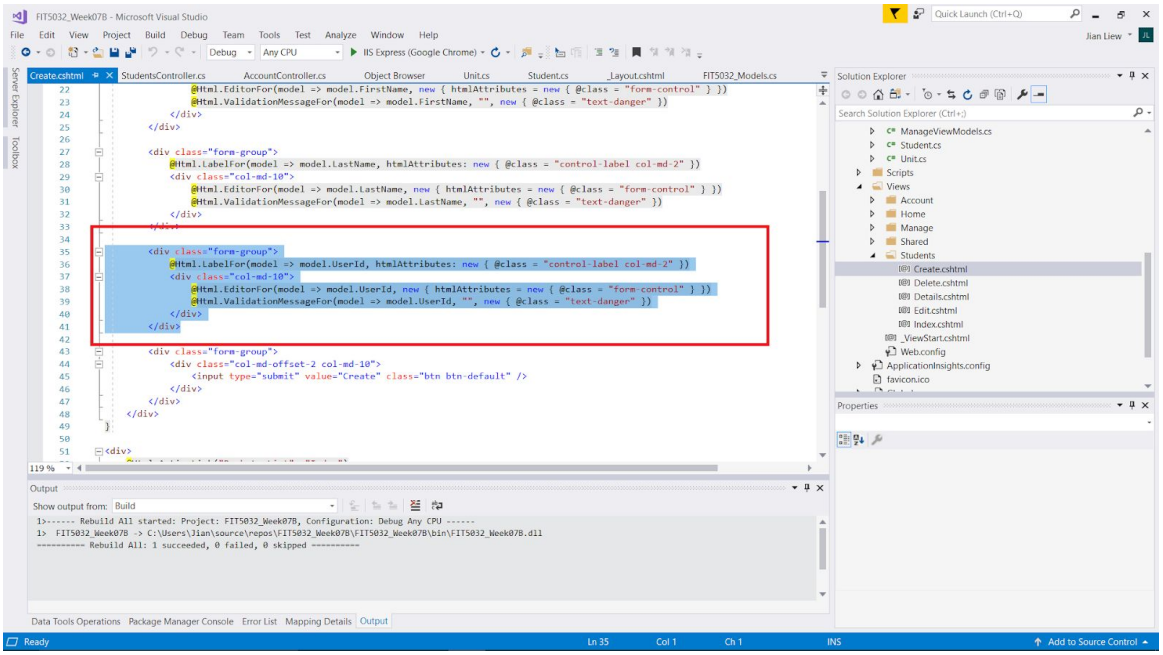
Now, because we have used to automatic scaffolding features. I will need to make some modifications.

I will first introduce some codes in the shared layout. **So, only an authenticated user will be able to see the link for the Students page.**



Step 22

Remove the following at the "Create.cshtml" for the Student Views.



Step 23

I will also make some changes to the StudentsController.cs at the Create method and the Index method. Here I will introduce an Authorize so that only authorized users can enter the Create page. I will also make minor changes to the model binding mechanism because I want the UserId from MS Identity to be binded to the Model in the Controller.

```
// POST: Students/Create
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize]
public ActionResult Create([Bind(Include = "Id,FirstName,LastName")] Student
student)
{
      student.UserId = User.Identity.GetUserId();

      ModelState.Clear();
      TryValidateModel(student);

      if (ModelState.IsValid)
      {
            db.Students.Add(student);
            db.SaveChanges();
            return RedirectToAction("Index");
      }

      return View(student);
}
```

Also make changes to the Index method. Here at the Index, I will only query information that belongs to that user. So, he or she can only see the entries made by themselves.

```
// GET: Students
[Authorize]
public ActionResult Index()
{
      var userId = User.Identity.GetUserId();

      var students = db.Students.Where(s => s.UserId == userId).ToList();

      return View(students);
}
```

Step 24

Now you can go ahead and test your application. **You should create multiple users so that you can see it in action. You will notice that each of the users can only see the entries they made.**

Explanation

What have I done was a really simple method of using MS Identity. However it can be way more complex when you start to introduce roles into the picture. **For example, you can create a role in the database called the Admin role.** With that, you can also use more attribute authorization to permit and disallow certain features for certain users.

Conclusion

Upon the completion of this tutorial

- You would understand how MS Identity works at a basic level
- How to use the Authorize attribute
- How to query specific data using the userId

Change Log

Date	Summary
16 August 2018	Initial draft of document.
06 September 2018	Added clarification if "Database is in use error is seen"