FIT5032: Internet Applications Development

Studio Assessment Task 1: Studio #3 & #4

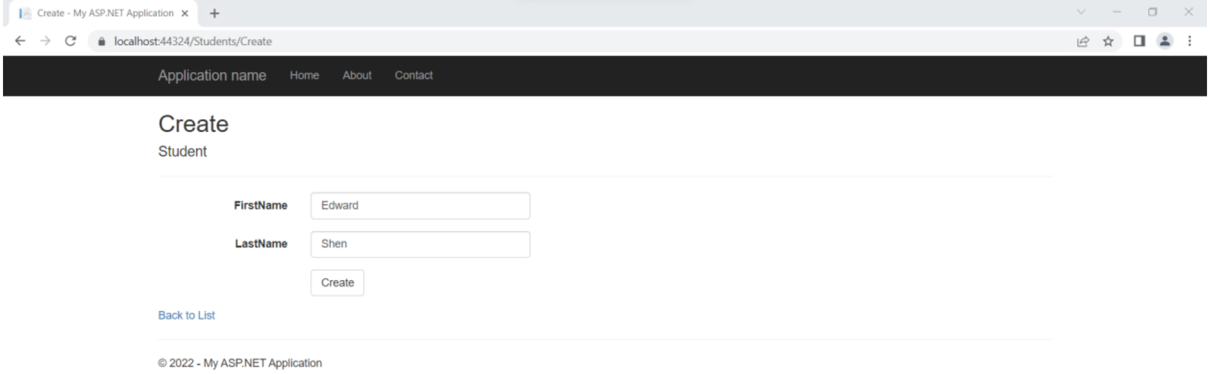
Name: Edward Shen

Student Id: 30594863

Date of submission: 21st August 2022

Self-Evaluation: High Distinction

Task 4.1: Model First Development Screenshots (Github):


Create - My ASP.NET Application
localhost:44324/Students/Create
Application name
Home
About
Contact
Create

Create

Student

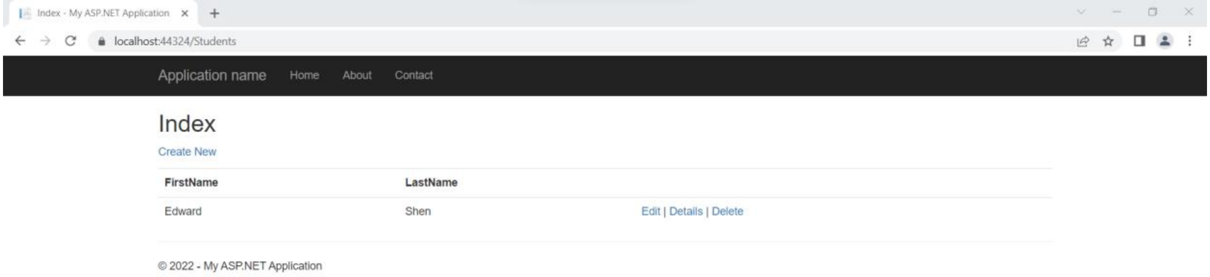
FirstName: Edward

LastName: Shen

Back to List

© 2022 - My ASP.NET Application

Figure 1: Create.cshtml (Running in localhost)



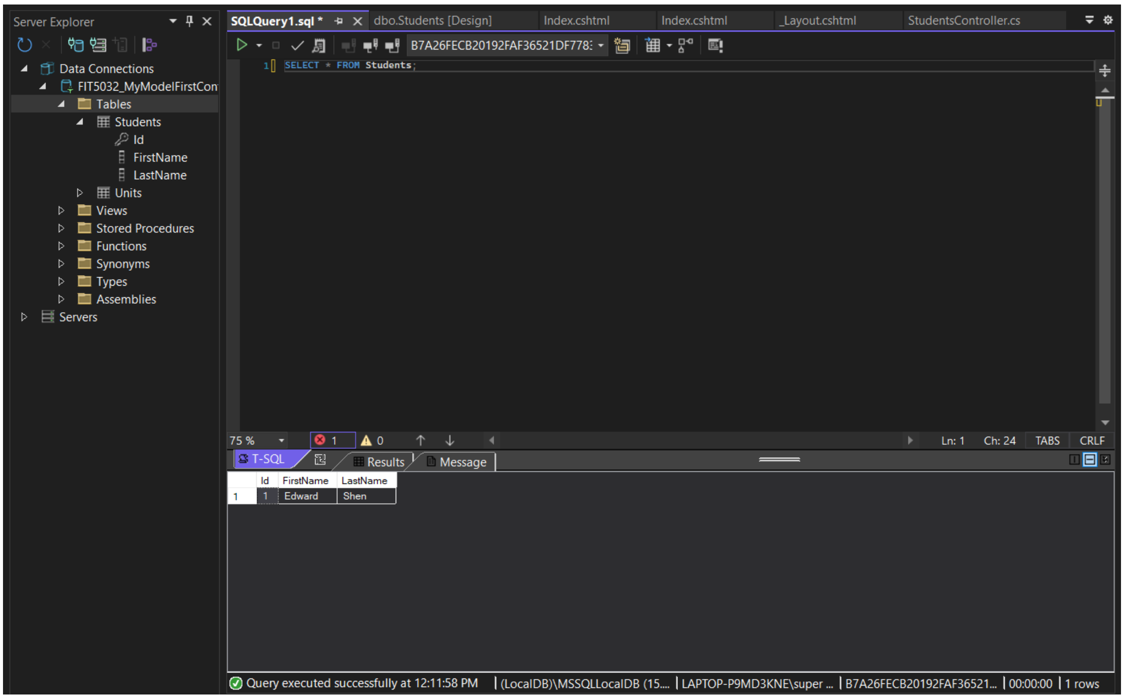
Index

Create New

FirstName	LastName	
Edward	Shen	Edit | Details | Delete

© 2022 - My ASP.NET Application

Figure 2: Index.cshtml after entering the student fields (Running in localhost)

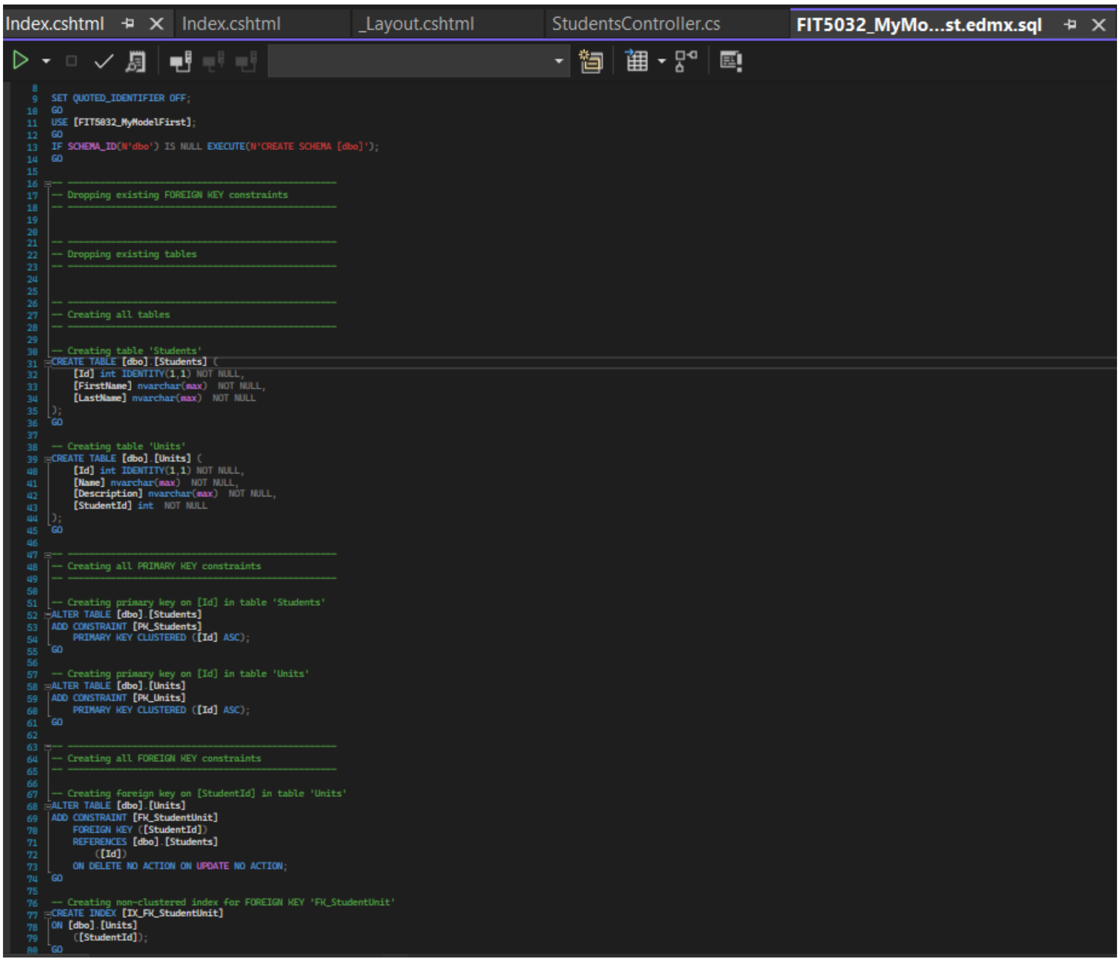


```
SELECT * FROM Students;
```

	Id	FirstName	LastName
1	1	Edward	Shen

Query executed successfully at 12:11:58 PM | (LocalDB)\MSSQLLocalDB (15.... | LAPTOP-P9MD3KNE\super ... | B7A26FECB20192FAF36521... | 00:00:00 | 1 rows

Figure 3: Querying the table after adding the student via the web application.

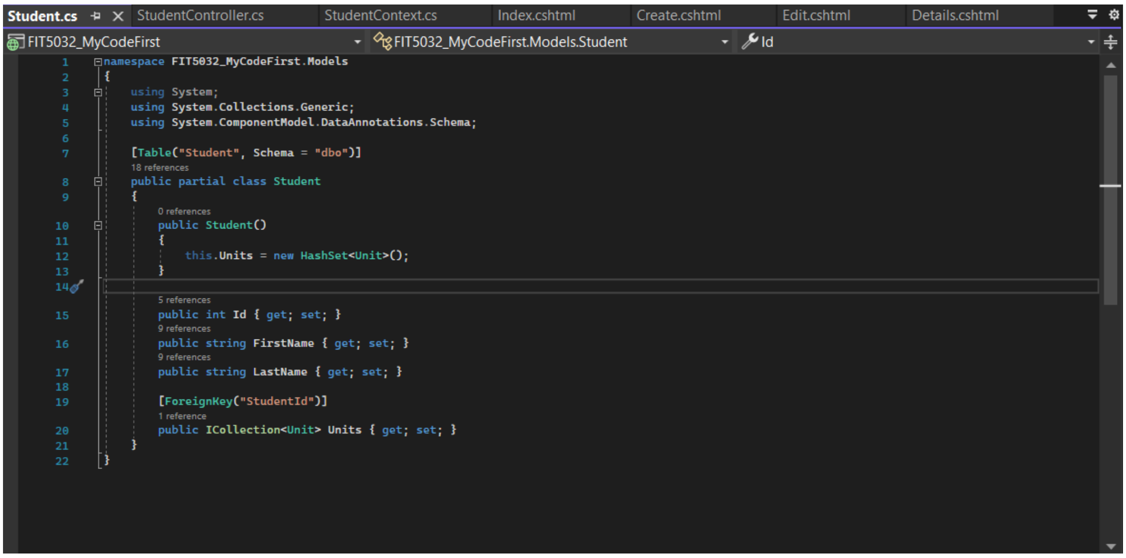


```
SET QUOTED_IDENTIFIER OFF;
GO
USE [FIT5032_MyModelFirst];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
GO

-- --------------------------------------------------
-- Dropping existing FOREIGN KEY constraints
-- --------------------------------------------------


-- --------------------------------------------------
-- Dropping existing tables
-- --------------------------------------------------


-- --------------------------------------------------
-- Creating all tables
-- --------------------------------------------------

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

-- --------------------------------------------------
-- Creating all PRIMARY KEY constraints
-- --------------------------------------------------

-- Creating primary key on [Id] in table 'Students'
ALTER TABLE [dbo].[Students]
ADD CONSTRAINT [PK_Students]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Units'
ALTER TABLE [dbo].[Units]
ADD CONSTRAINT [PK_Units]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- --------------------------------------------------
-- Creating all FOREIGN KEY constraints
-- --------------------------------------------------

-- Creating foreign key on [StudentId] in table 'Units'
ALTER TABLE [dbo].[Units]
ADD CONSTRAINT [FK_StudentUnit]
    FOREIGN KEY ([StudentId])
    REFERENCES [dbo].[Students]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_StudentUnit'
CREATE INDEX [IX_FK_StudentUnit]
ON [dbo].[Units]
    ([StudentId]);
GO
```

Figure 4: Screenshot of Generated Student Schema. (Github)

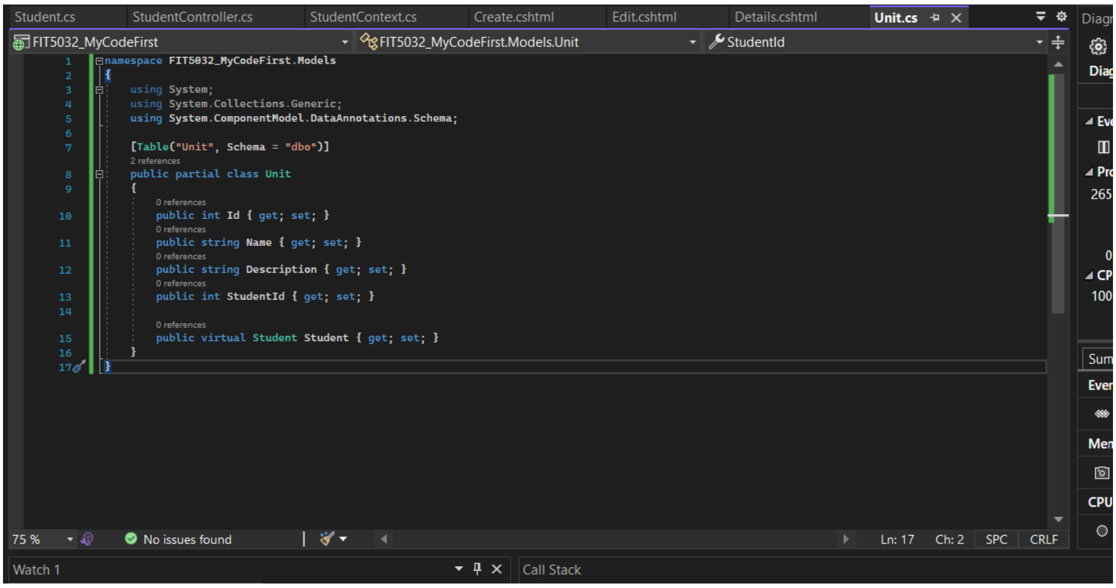
Task 4.2: Code First Development Screenshots (Github):

Note that the controller and cshtml code I used for this is identical to Task 4.1. The only major difference being the way the Domain classes under /Models is written.



```
namespace FIT5032_MyCodeFirst.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations.Schema;

    [Table("Student", Schema = "dbo")]
    public partial class Student
    {
        public Student()
        {
            this.Units = new HashSet<Unit>();
        }

        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }

        [ForeignKey("StudentId")]
        public ICollection<Unit> Units { get; set; }
    }
}
```

Figure 5: Domain Class for Student.cs (Github)

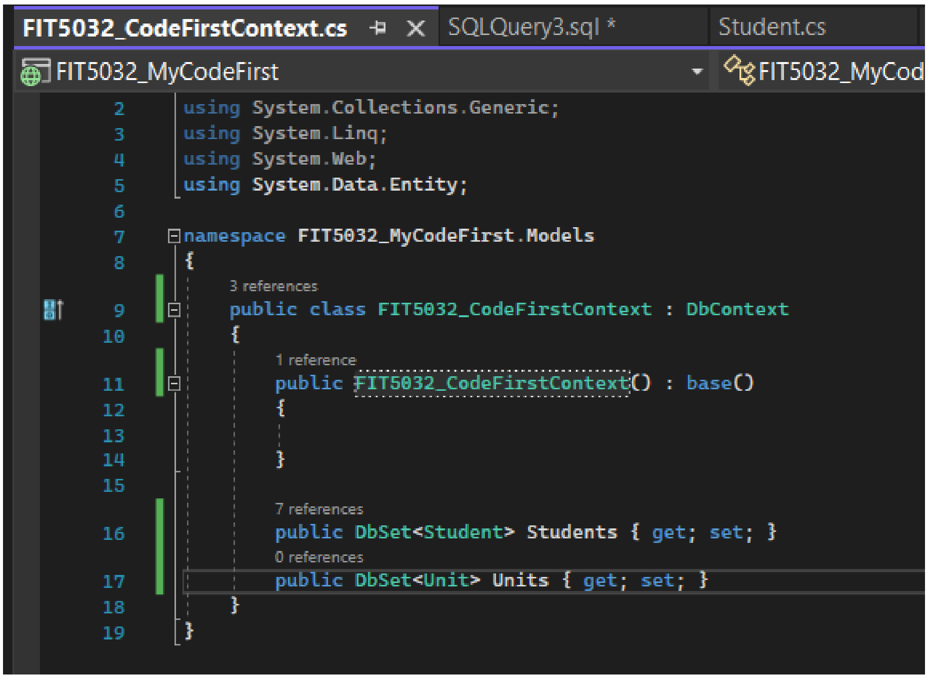


```
namespace FIT5032_MyCodeFirst.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations.Schema;

    [Table("Unit", Schema = "dbo")]
    public partial class Unit
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public int StudentId { get; set; }

        public virtual Student Student { get; set; }
    }
}
```

Figure 6: Domain Class for Unit.cs (Github)

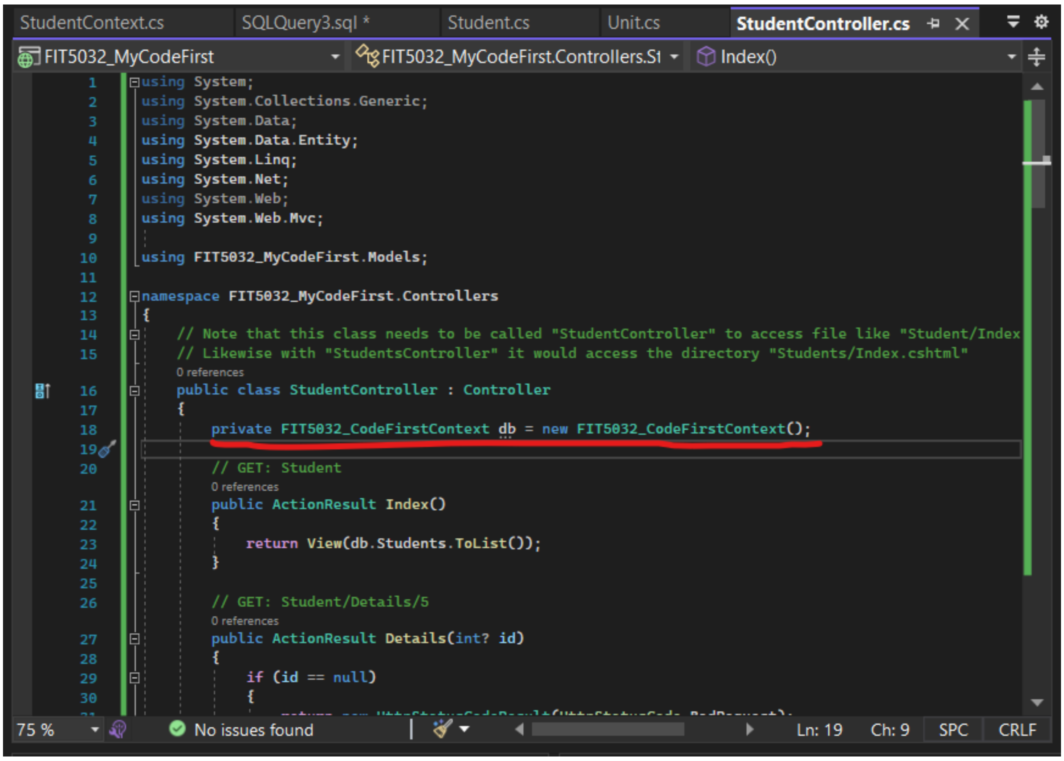
The only major difference being the [Table(“table_name”, Schema=”db_name”)] annotation at the top along with the [ForeignKey(“fk_name”)] annotation.

Note that the FIT5032_CodeFirstContext.cs class was used to connect to the database (Github):



```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;

namespace FIT5032_MyCodeFirst.Models
{
    3 references
    public class FIT5032_CodeFirstContext : DbContext
    {
        1 reference
        public FIT5032_CodeFirstContext() : base()
        {

        }

        7 references
        public DbSet<Student> Students { get; set; }
        0 references
        public DbSet<Unit> Units { get; set; }
    }
}
```

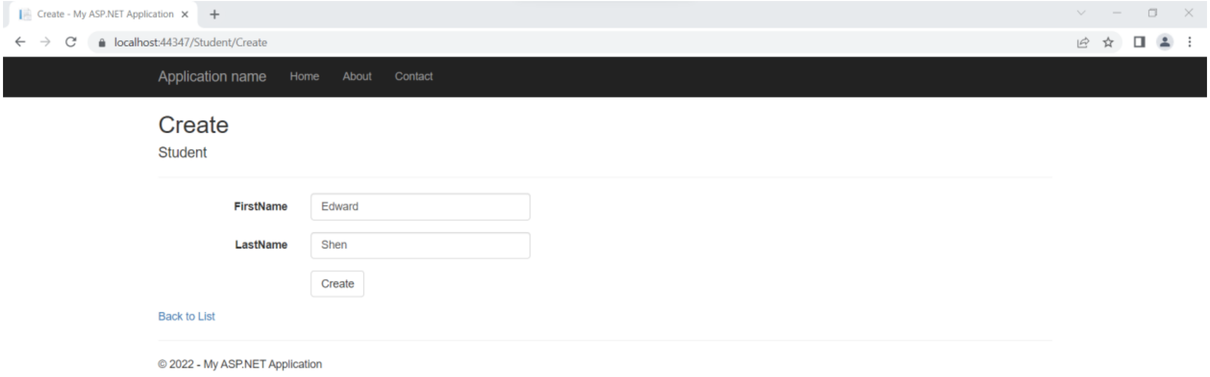
Figure 7: The DbContext subclass I am using to interact w/ SQL Server database.



```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;

using FIT5032_MyCodeFirst.Models;

namespace FIT5032_MyCodeFirst.Controllers
{
    // Note that this class needs to be called "StudentController" to access file like "Student/Index
    // Likewise with "StudentsController" it would access the directory "Students/Index.cshtml"
    0 references
    public class StudentController : Controller
    {
        private FIT5032_CodeFirstContext db = new FIT5032_CodeFirstContext();

        // GET: Student
        0 references
        public ActionResult Index()
        {
            return View(db.Students.ToList());
        }

        // GET: Student/Details/5
        0 references
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
```

Figure 8: Dependency Injection of the Context class from the previous figure. Also note that the controller code is (basically) the same generated code from the model first approach (I couldn't be bothered to rewrite it from scratch… but usually we would want to modify the code in the controller to adhere to business requirements… but this should be enough for a quick POC). (Github)

Demo of Entering A row into our SQL Server Database:


Create - My ASP.NET Application
localhost:44347/Student/Create
Application name
Home
About
Contact
Create

Create

Student

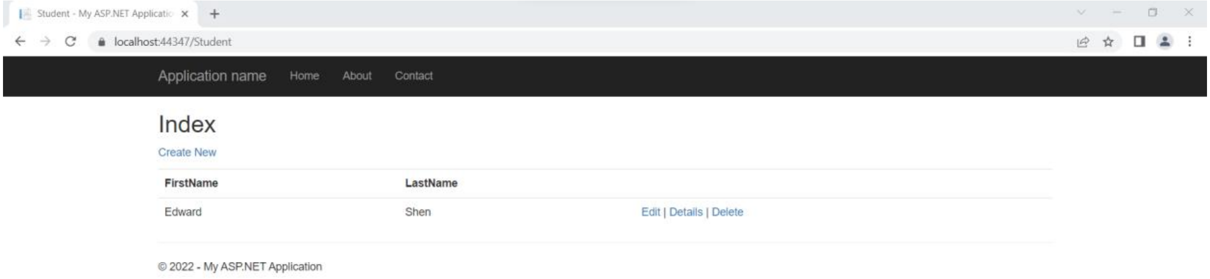
FirstName: Edward

LastName: Shen

Back to List

© 2022 - My ASP.NET Application

Figure 9: Entering it in via our web application:



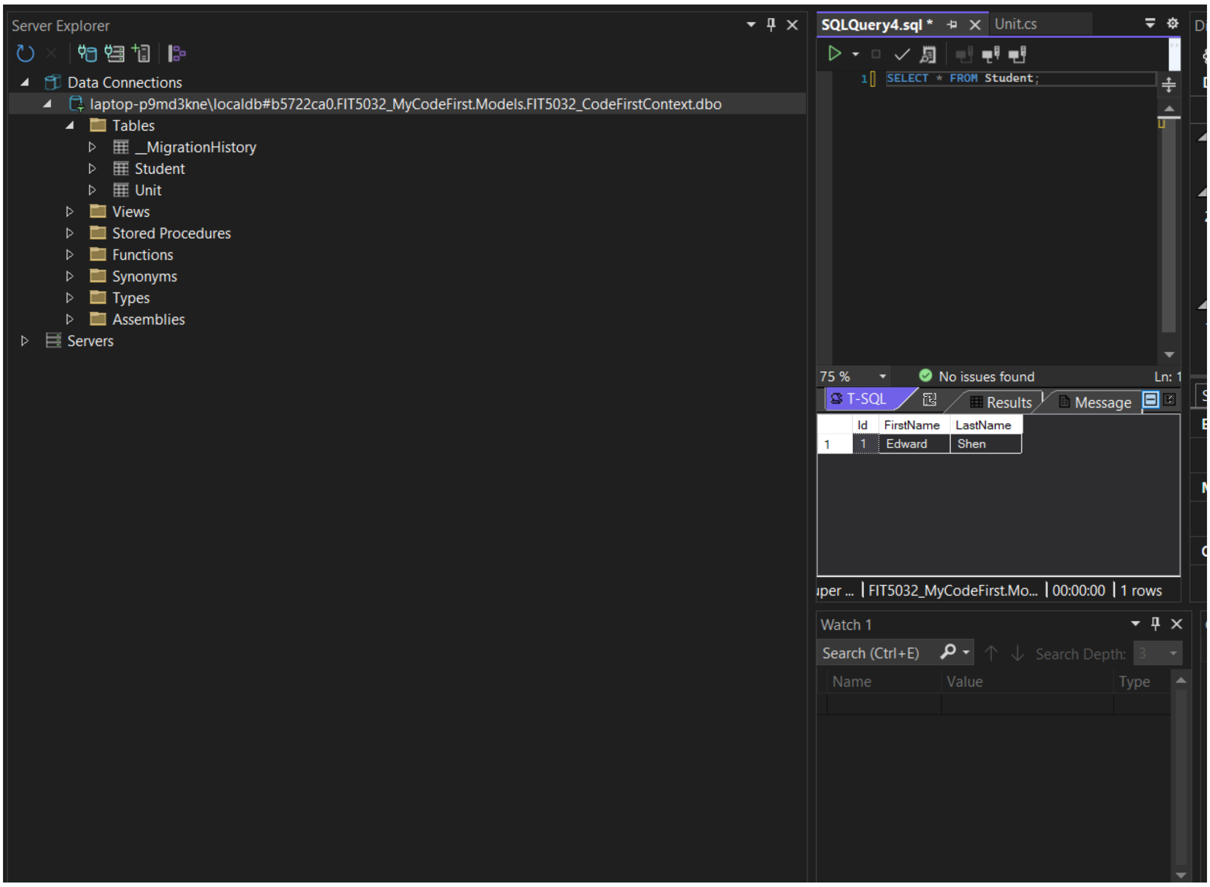
Index

Create New

FirstName	LastName	
Edward	Shen	Edit | Details | Delete

© 2022 - My ASP.NET Application

Figure 10: It now shows on our Student/Index.cshtml page.



```
SELECT * FROM Student;
```

	Id	FirstName	LastName
1	1	Edward	Shen

```
per ... | FIT5032_MyCodeFirst.Mo... | 00:00:00 | 1 rows
```

Figure 11: Directly Query the database that the context created to make sure it saved the record. (Note that the name of the data connection itself “FIT5032_CodeFirstContext.dbo” which was specified in our Student.cs domain class earlier.)

Note I won’t do a demonstration for CRUD on the Unit.cs Domain class just to keep this short (I’m pretty sure this is enough to cover the efolio task anyway and it shouldn’t be too difficult to figure out since we’ve already done it with the student class).