

**Module**Web Solutions  
Development**Module Code**

IT5035FP

**Duration**

18 hours

# LABSHEET 9

**Title:**

Development of a web application using server-side scripting to access and process information from a database server.

**Objective(s):**

In this lab, students will learn to create a web application using server-side scripting and data processing framework upon completing the following:

- **Part (A):** Create MVC (Model-View-Controller) project
  - Install Entity Framework package
- **Part (B):** Create the Model
  - Create a Model class for a table
  - Create a Data Context class for all database operations
  - Define a database connection
- **Part (C):** Create the Controller
  - Define action for Create operation
  - Define action for Retrieve operation
  - Define action for Update operation
  - Define action for Delete operation
- **Part (D):** Create the View
  - Create a view to list all records from a table (using the Retrieve operation)
  - Create a view to display a specific record from a table (using the Retrieve operation)
  - Create a view to insert a record into a table (using the Create operation)
  - Create a view to edit a record in a table (using the Update operation)
  - Create a view to delete a record in a table (using the Delete operation)
- **Part (E):** Create a Login page for Administrator access

**Tools, Equipment and Materials:**

- 1 Personal Computer with Internet access
- 2 SQL Server
- 3 SQL Server Management Studio
- 4 Visual Studio

# LABSHEET 9

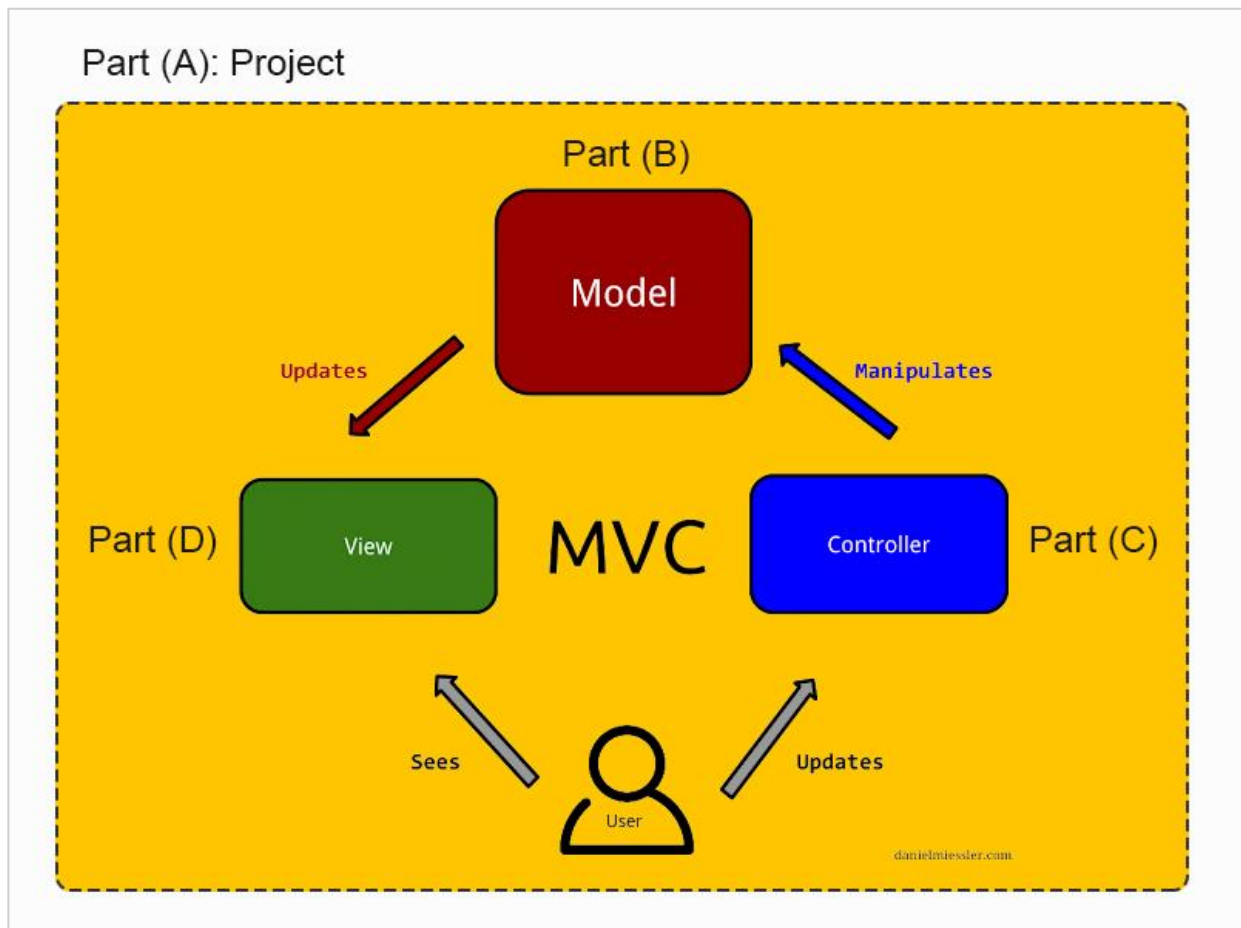
Module: Web Solutions Development

Module Code: IT5035FP

## Overview:

In this lab, students will start by setting up the project in Part (A). Once the project is set up, the MVC model is developed part by part. The Model is created first in Part (B) to connect to the database and handle the data in the application. The Controller is then created in Part (C) to define the CRUD (Create Retrieve Update Delete) operations. The View is then generated in Part (D) to provide the display function. With Part (D) completed, students will have the MVC web application running with the CRUD functions.

In Part (E), students will modify the MVC web application to include an Admin Login page. This addition will require students to modify the Model, Controller and View to complete the web application.



Development of MVC Web Application

# LABSHEET 9

Module: Web Solutions Development

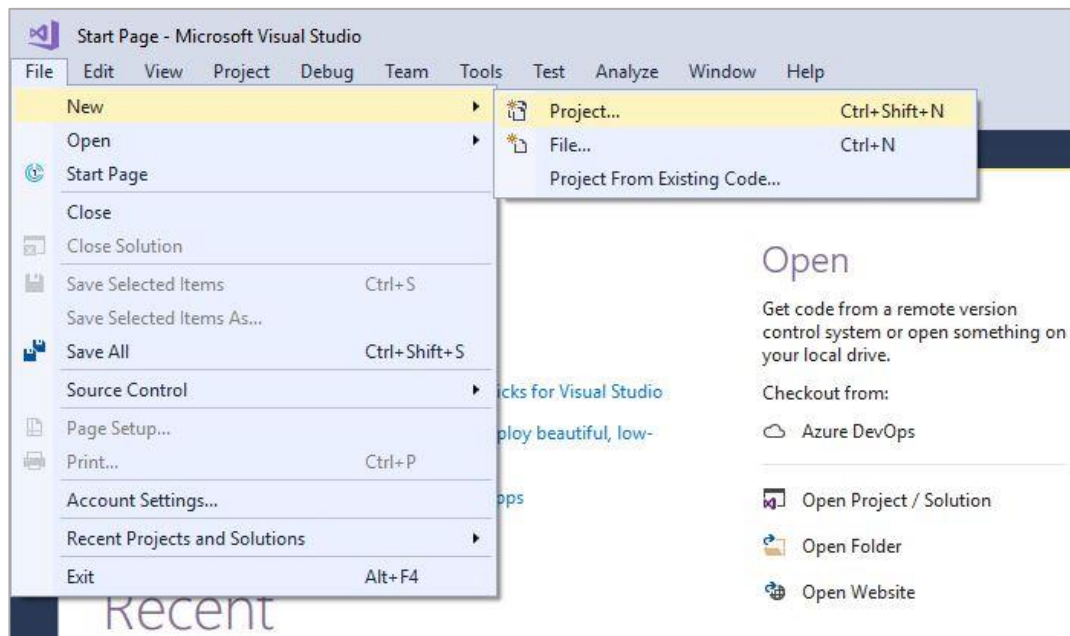
Module Code: IT5035FP

## Instructions:

### Part (A): Create MVC Project using EntityFramework Package

In Part (A), students will start the lab by creating an MVC project that uses Entity Framework. Entity Framework is an Object Relational Mapper (ORM), which is a tool that makes data access easy for developers.

- 1 Launch SQL Server Management Studio and Visual Studio.
- 2 Create a new project in Visual Studio.

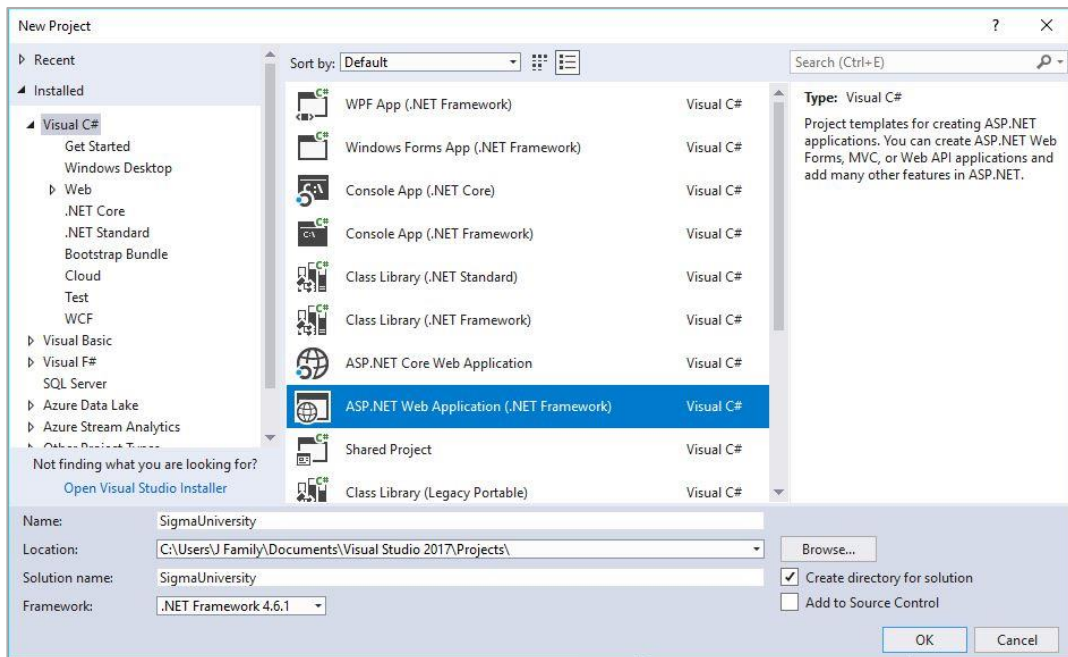


# LABSHEET 9

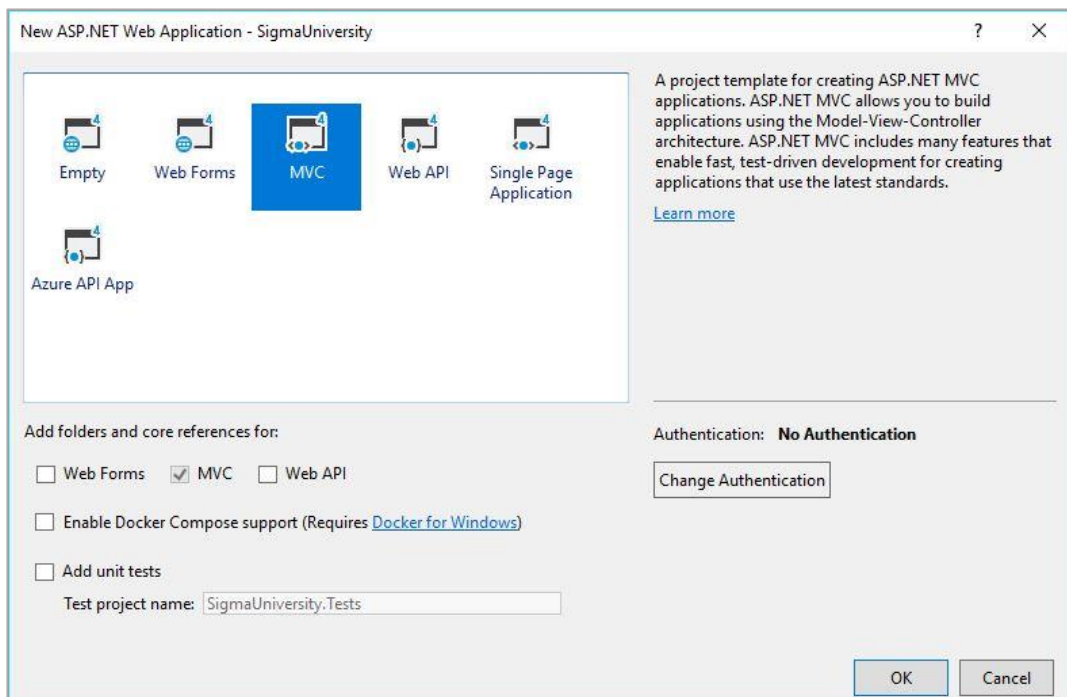
Module: Web Solutions Development

Module Code: IT5035FP

- 3 Select the **ASP.NET Web Application (.NET Framework) Visual C#** project template. Set the project **Name** as **SigmaUniversity**. Set the project **Location** to your preferred folder. Click on the **OK** button.



- 4 Select the **MVC** project template. Click on the **OK** button to generate the project.

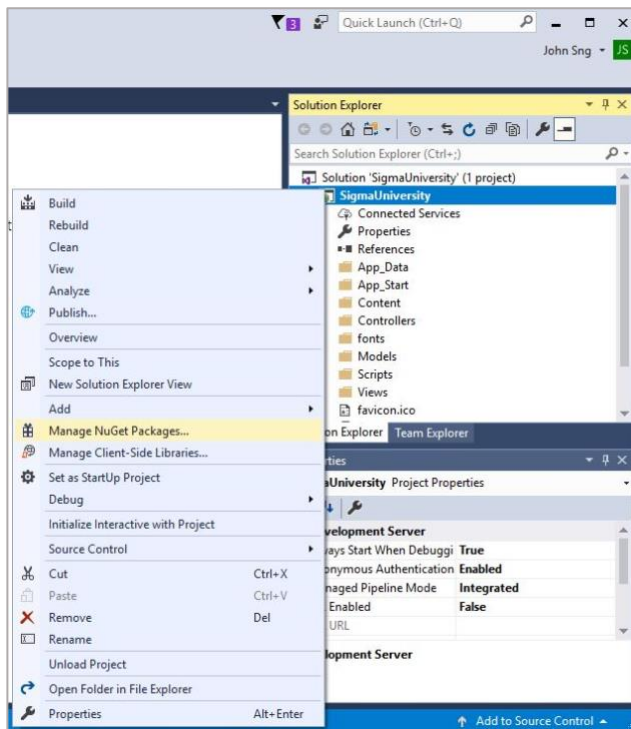


# LABSHEET 9

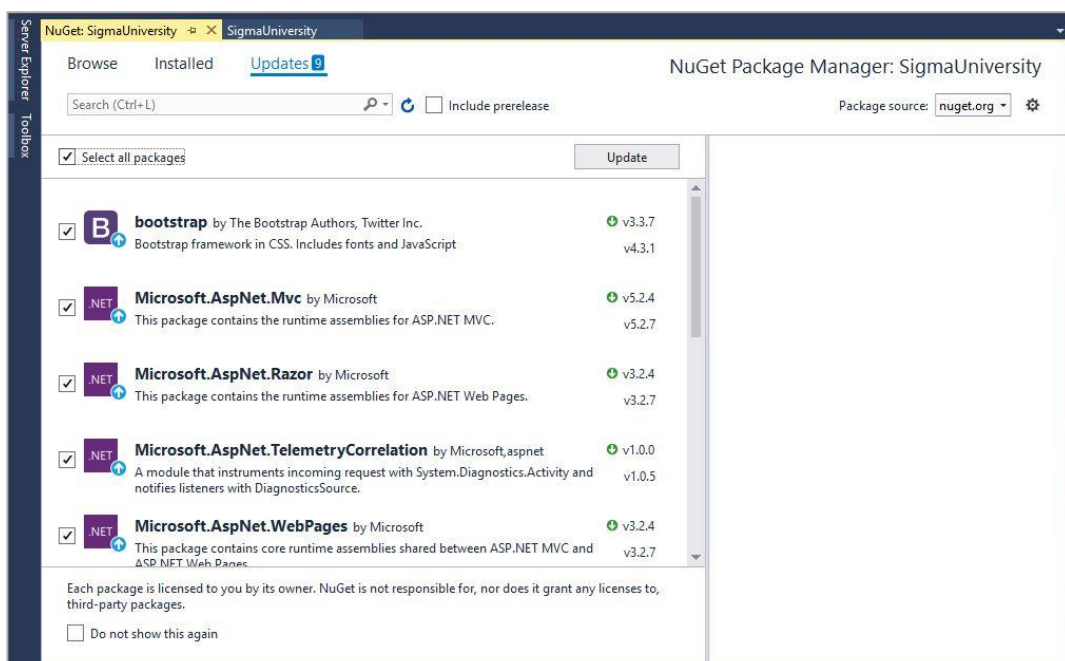
Module: Web Solutions Development

Module Code: IT5035FP

- 5 When the project has been generated successfully, go to the **Solution Explorer** panel and right-click on the project **SigmaUniversity**. Click on the **Manage NuGet Packages** option.



- 6 Select the **Updates** tab, and check the **Select all packages** option. Click on the **Update** button to update all packages.

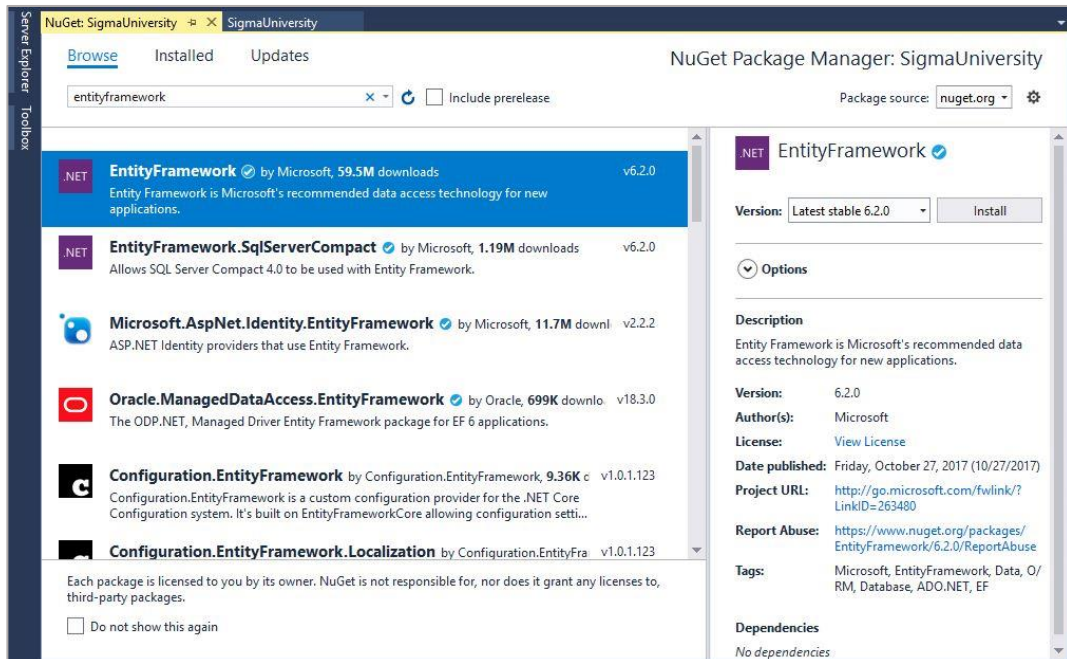


# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 7 Select the **Browse** tab, and enter **entityframework** into the **Search box**. If EntityFramework package is not installed, select the **EntityFramework** package, and click on the **Install** button to install the package. Close the **NuGet** tab after the installation has completed successfully.





# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## Part (B): Create the Model

In Part (B), students will learn to create the **Model** in the MVC (**Model**-View-Controller) project. Students will only be using the COLLEGE table in SigmaUniversity database created in the previous lab sheet. To ensure consistency in table design and data, students will drop all tables in SigmaUniversity database and recreate the COLLEGE table. Part (B) will be done in 4 subparts:

- Part (B1): Create COLLEGE table with data
- Part (B2): Create Model class for COLLEGE table
- Part (B3): Create Data Context class for all database operations
- Part (B4): Define database connection in Web.config

### Part (B1): Create COLLEGE Table with Data

- 1 To ensure students use the same COLLEGE table and data, execute the following SQL commands to drop all existing tables in **SigmaUniversity** database.

```

/* ***** */
/* 1. Drop all tables */
/* ***** */

/* 1.1: Drop PROFESSOR table */
ALTER TABLE PROFESSOR
    DROP CONSTRAINT DepartmentFK1;
DROP TABLE PROFESSOR

/* 1.2: Drop STUDENT table */
ALTER TABLE STUDENT
    DROP CONSTRAINT DepartmentFK2;
DROP TABLE STUDENT

/* 1.3: Drop DEPARTMENT table */
ALTER TABLE DEPARTMENT
    DROP CONSTRAINT CollegeFK;
DROP TABLE DEPARTMENT

/* 1.4: Drop COLLEGE table */
DROP TABLE COLLEGE
  
```

- 2 Execute the following SQL commands to create COLLEGE table. Note that the **constraint CollegeNameValues** has been removed in this lab.

```

/* ***** */
/* 2. Create COLLEGE table */
/* ***** */

/* 2.1: Create COLLEGE table */
CREATE TABLE COLLEGE (
    CollegeCode      INT          NOT NULL,
    CollegeName      NCHAR(100)   NOT NULL,
    DeanFirstName    NCHAR(50)    NOT NULL,
    DeanLastName     NCHAR(50)    NOT NULL,
    DeanEmail        NCHAR(50)    NOT NULL,
    CONSTRAINT       CollegePK     PRIMARY KEY (CollegeCode)
);
  
```

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 3 Execute the following SQL commands to insert data into COLLEGE table.

```
/* ***** */
/* 3. Populate COLLEGE table */
/* ***** */

/* 3.1: Insert data into COLLEGE table */
INSERT INTO COLLEGE (CollegeCode, CollegeName, DeanFirstName, DeanLastName, DeanEmail)
VALUES (100, 'College of Business', 'Mary', 'Tan', 'tan.mary@sigma.uni.edu');
INSERT INTO COLLEGE (CollegeCode, CollegeName, DeanFirstName, DeanLastName, DeanEmail)
VALUES (200, 'College of Engineering', 'Robert', 'Goh', 'goh.robert@sigma.uni.edu');
```



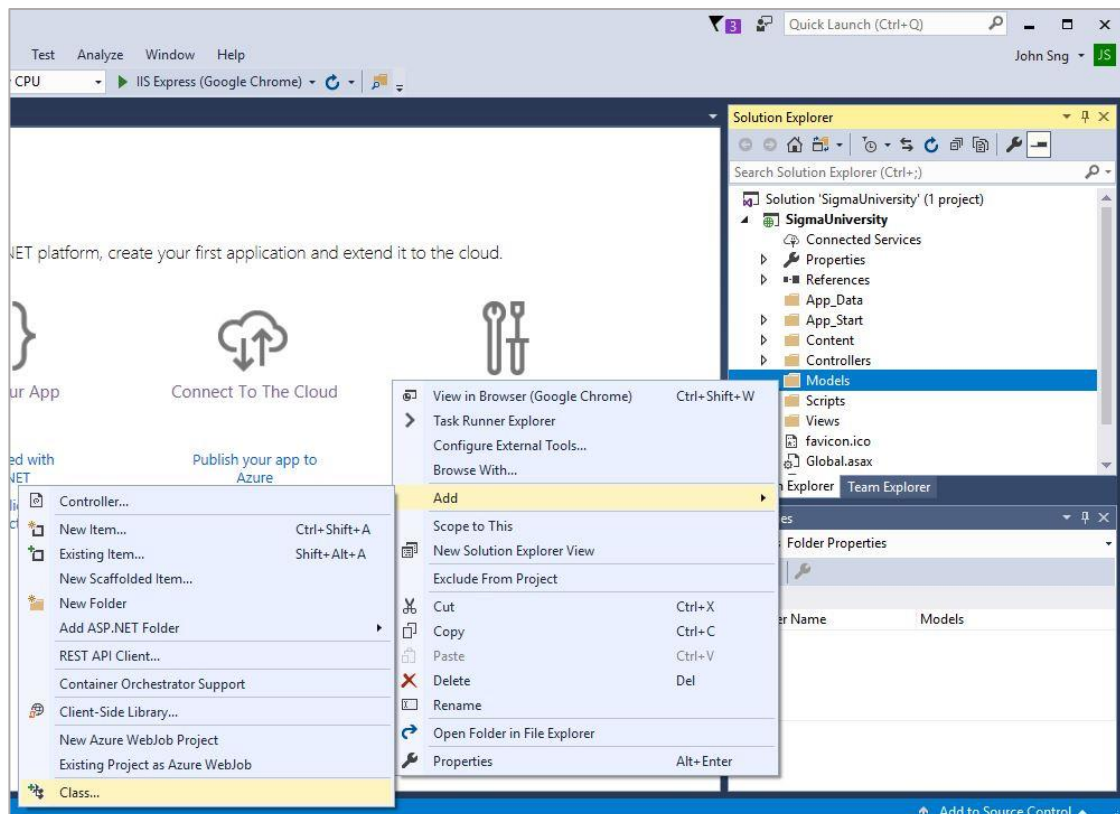
# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## Part (B2): Create Model Class for COLLEGE Table

- 1 To create a Model class, go to the Solution Explorer panel and right-click on the **Models** folder. Select **Add** → **Class**.

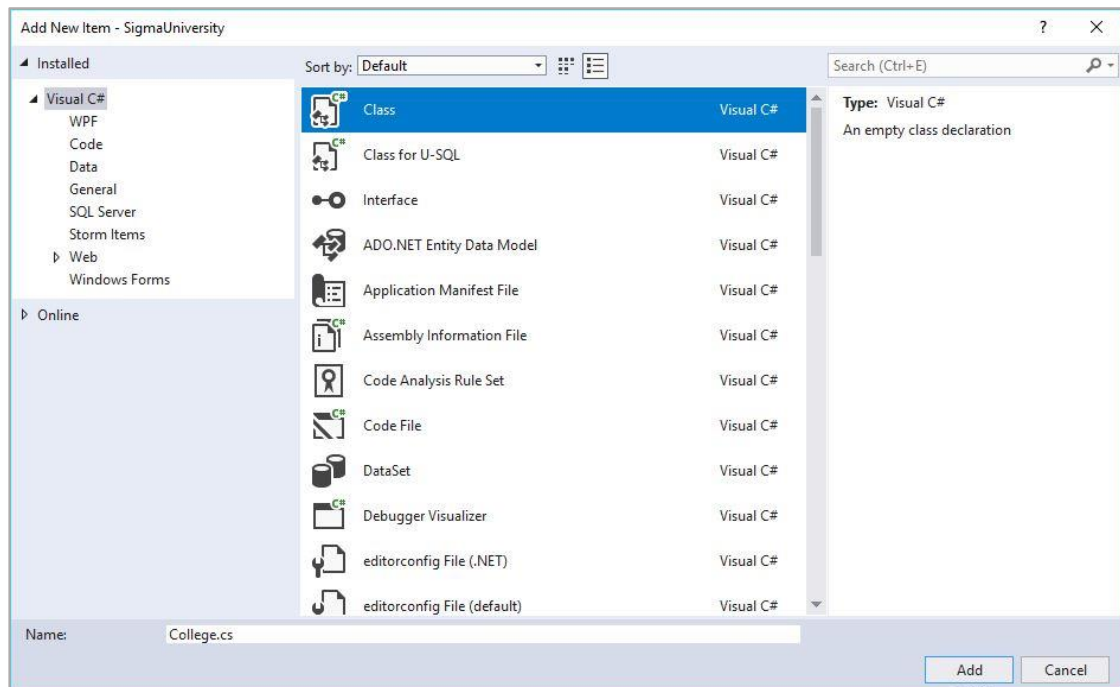


# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 2 In the pop-up **Add New Item** window, select **Visual C# -> Class** item. Set the **Name** to **College.cs**. Click on the **Add** button.



- 3 The following code will be generated by Visual Studio.

```

1  using System;
2      using System.Collections.Generic;
3      using System.Linq;
4      using System.Web;
5
6  namespace SigmaUniversity.Models
7  {
8      public class College
9      {
10     }
11 }

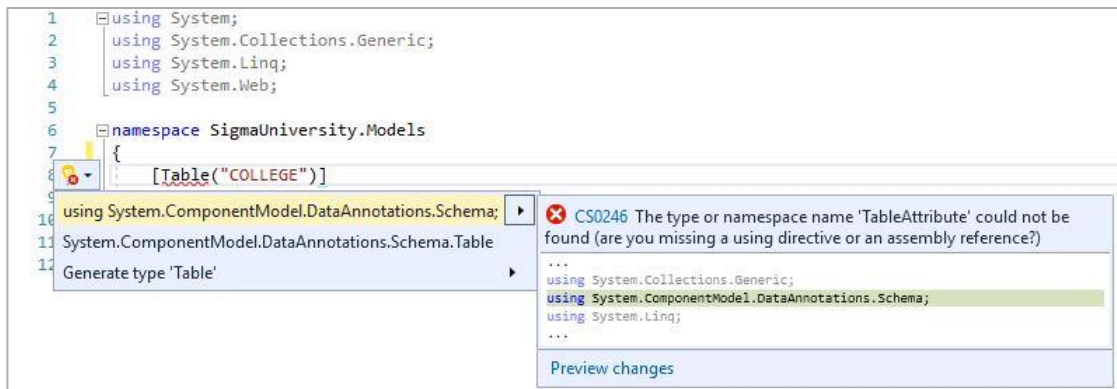
```

# LABSHEET 9

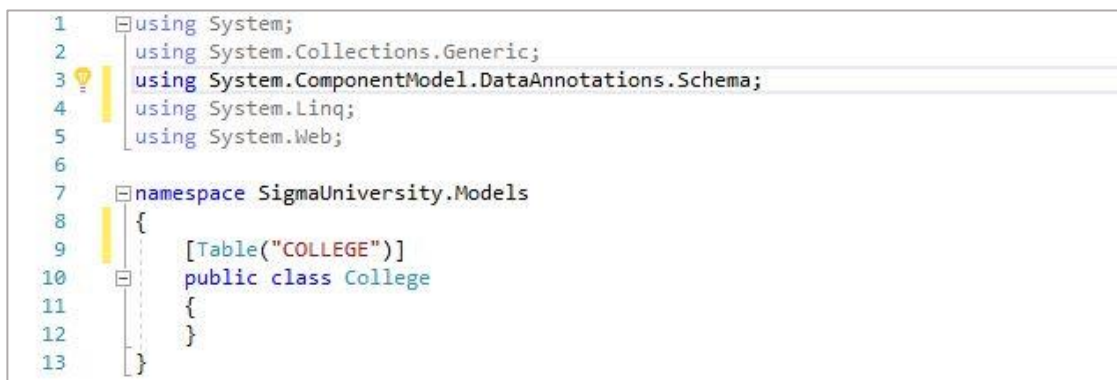
Module: Web Solutions Development

Module Code: IT5035FP

- 4 Enter the following code to add the Table Attribute. A Lightbulb icon appears on the left to suggest the required class that needs to be added. Click on the **Lightbulb** icon on the left and select the class (i.e. System.ComponentModel.DataAnnotations.Schema) as shown.



- 5 The class will be added into the code:  
***using System.ComponentModel.DataAnnotations.Schema;***

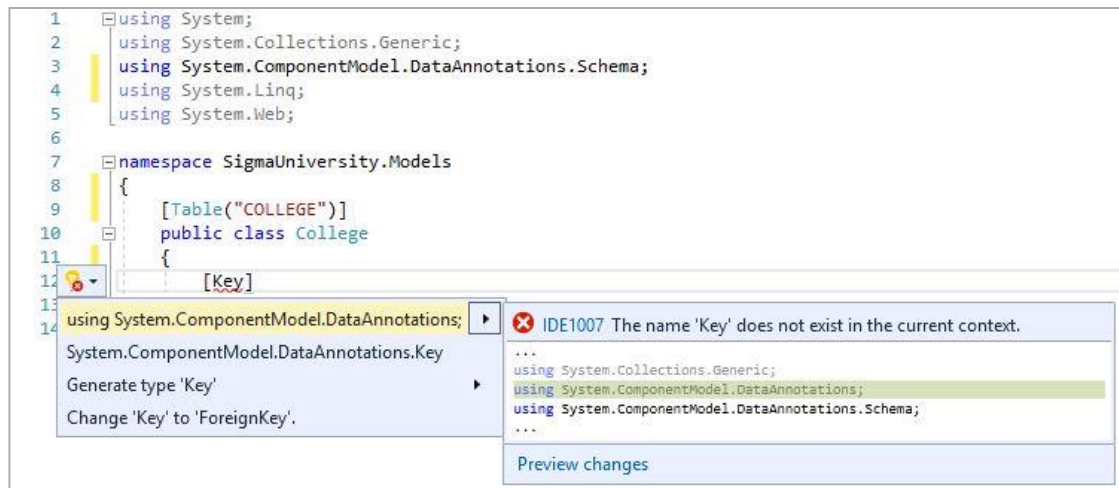


# LABSHEET 9

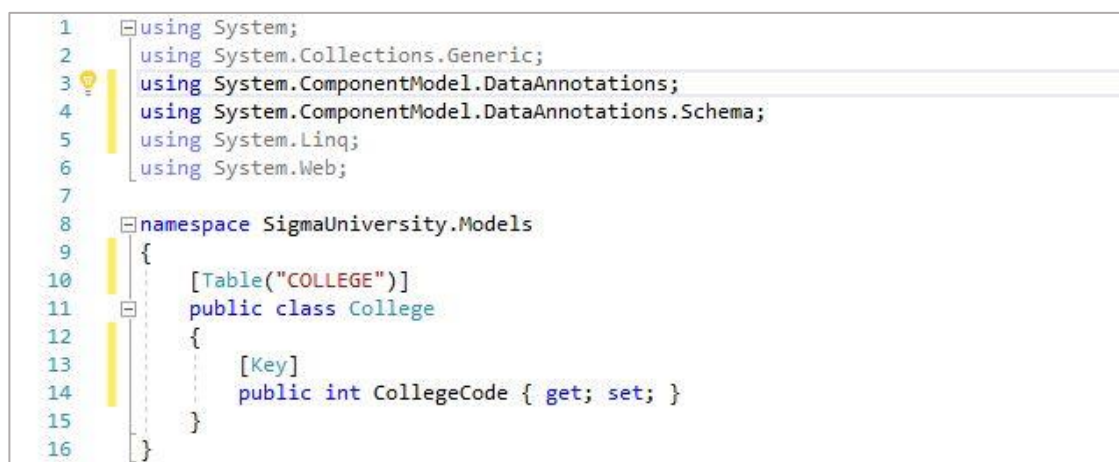
Module: Web Solutions Development

Module Code: IT5035FP

- 6 Enter the following code to add the Key Attribute. A Lightbulb icon appears on the left to suggest the required class that needs to be added. Click on the **Lightbulb** icon on the left and select the class (i.e. System.ComponentModel.DataAnnotations) as shown.



- 7 The class will be added into the code:  
**using System.ComponentModel.DataAnnotations;**



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 8 Enter the following code to define all the fields of the COLLEGE table.

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel.DataAnnotations;
4  using System.ComponentModel.DataAnnotations.Schema;
5  using System.Linq;
6  using System.Web;
7
8  namespace SigmaUniversity.Models
9  {
10     [Table("COLLEGE")]
11     public class College
12     {
13         [Key]
14         public int CollegeCode { get; set; }
15         public string CollegeName { get; set; }
16         public string DeanFirstName { get; set; }
17         public string DeanLastName { get; set; }
18         public string DeanEmail { get; set; }
19     }
20 }
```

- 9 Click on the **Save All** button in the main menu.

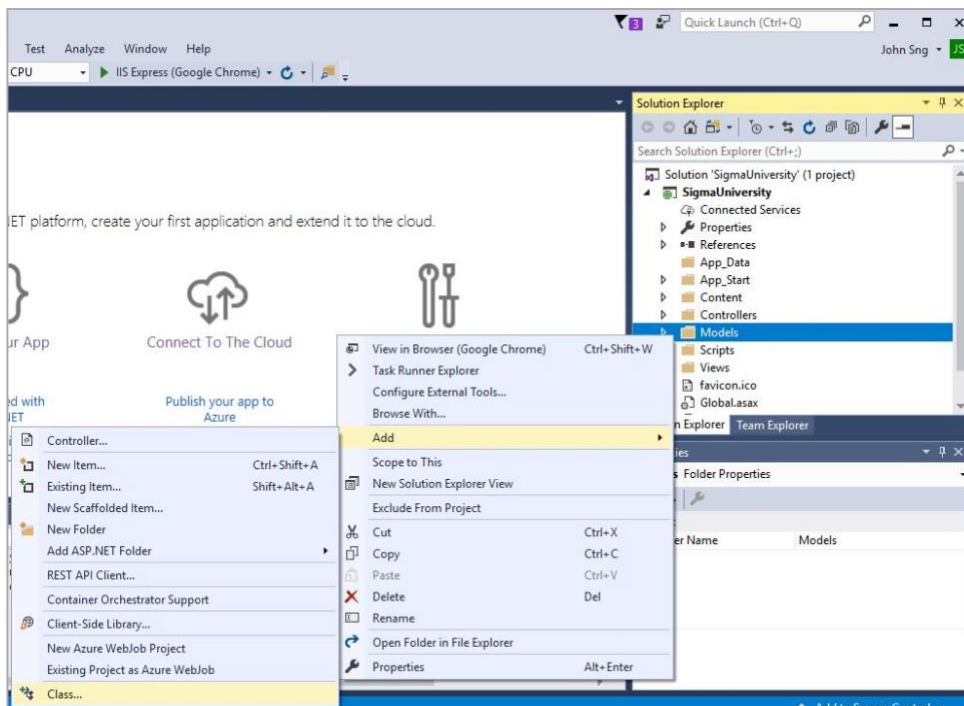
# LABSHEET 9

Module: Web Solutions Development

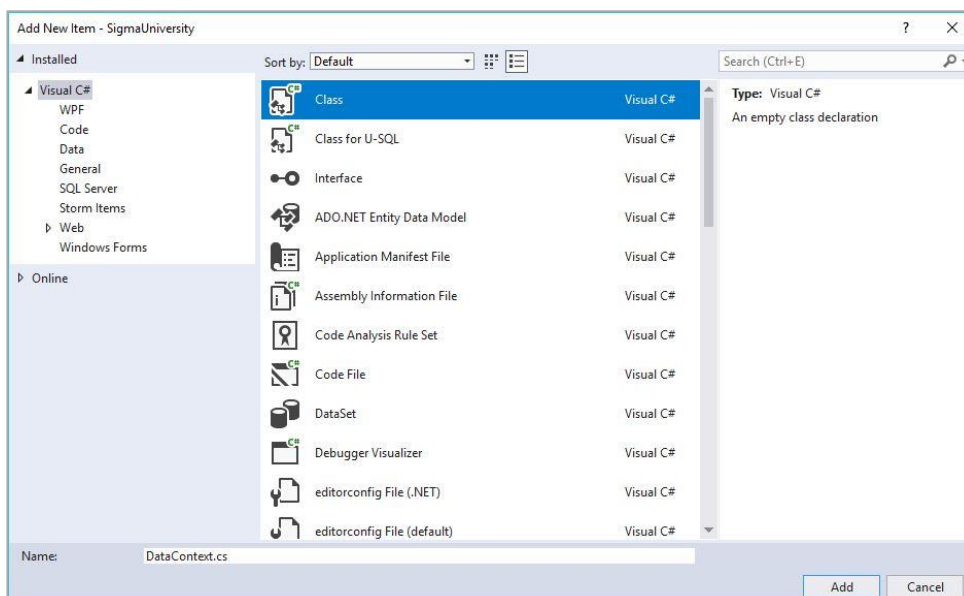
Module Code: IT5035FP

## Part (B3): Create Data Context Class for All Database Operations

- 1 To create the Data Context class, go to the Solution Explorer panel and right-click on the **Models** folder. Select **Add** → **Class**.



- 2 In the pop-up **Add New Item** window, select **Visual C#** -> **Class** item. Set the **Name** to **DataContext.cs**. Click on the **Add** button.





# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 3 Enter the following code to define the interaction with the database.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Data.Entity;
4  using System.Linq;
5  using System.Web;
6
7  namespace SigmaUniversity.Models
8  {
9      public class DataContext : DbContext
10     {
11         public DataContext(): base("conn") { }
12         public DbSet<College> COLLEGE { get; set; }
13     }
14 }
```

- 4 Click on the **Save All** button in the main menu.

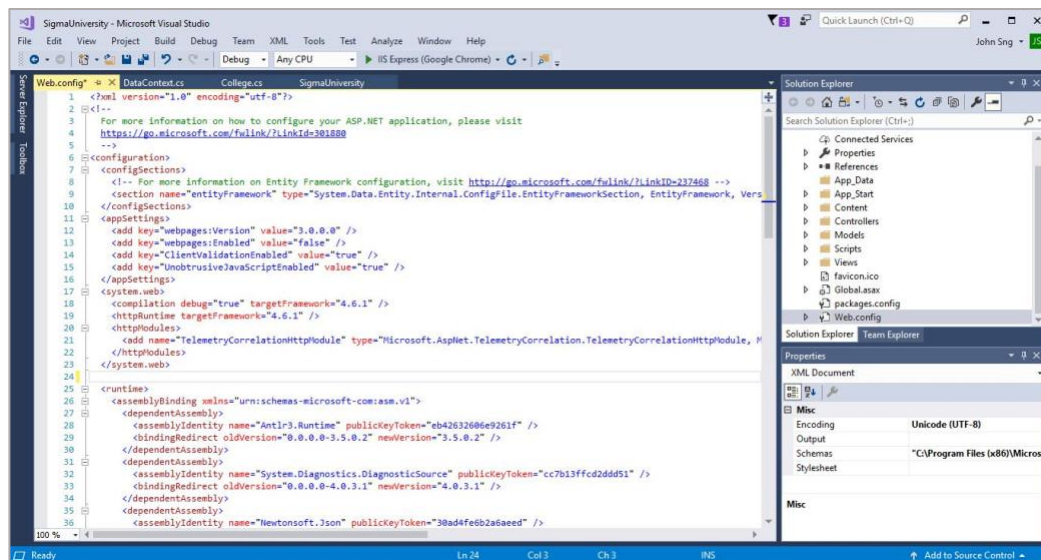
# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## Part (B4): Define Database Connection in Web.config

- 1 In ASP.NET, the Web.config file is responsible for controlling the behaviour of the application. Hence, the database connection defined in the Data Context class must be configured in Web.config. To access Web.config, go to the **Solution Explorer** panel and double-click on **Web.config**. The configuration code should be added after the **</system.web>** tag, and before the **<runtime>** tags.

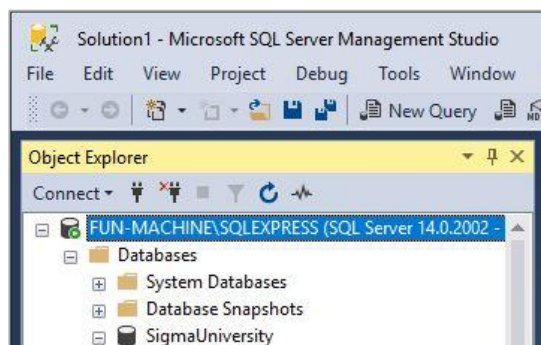


- 2 Add the following configuration code after the **</system.web>** tag, and before the **<runtime>** tags. **Note that you need to change the *server* setting in the example (e.g. FUN-MACHINE\SQLEXPRESS) to the server name in your SQL Server Management Studio.**

```

23 </system.web>
24 <connectionStrings>
25   <add name="conn" connectionString="server=FUN-MACHINE\SQLEXPRESS;database=SigmaUniversity;integrated security=true" providerName="System.Data.SqlClient"/>
26 </connectionStrings>
27 <runtime>

```



- 3 Click on the **Save All** button in the main menu.

# LABSHEET 9

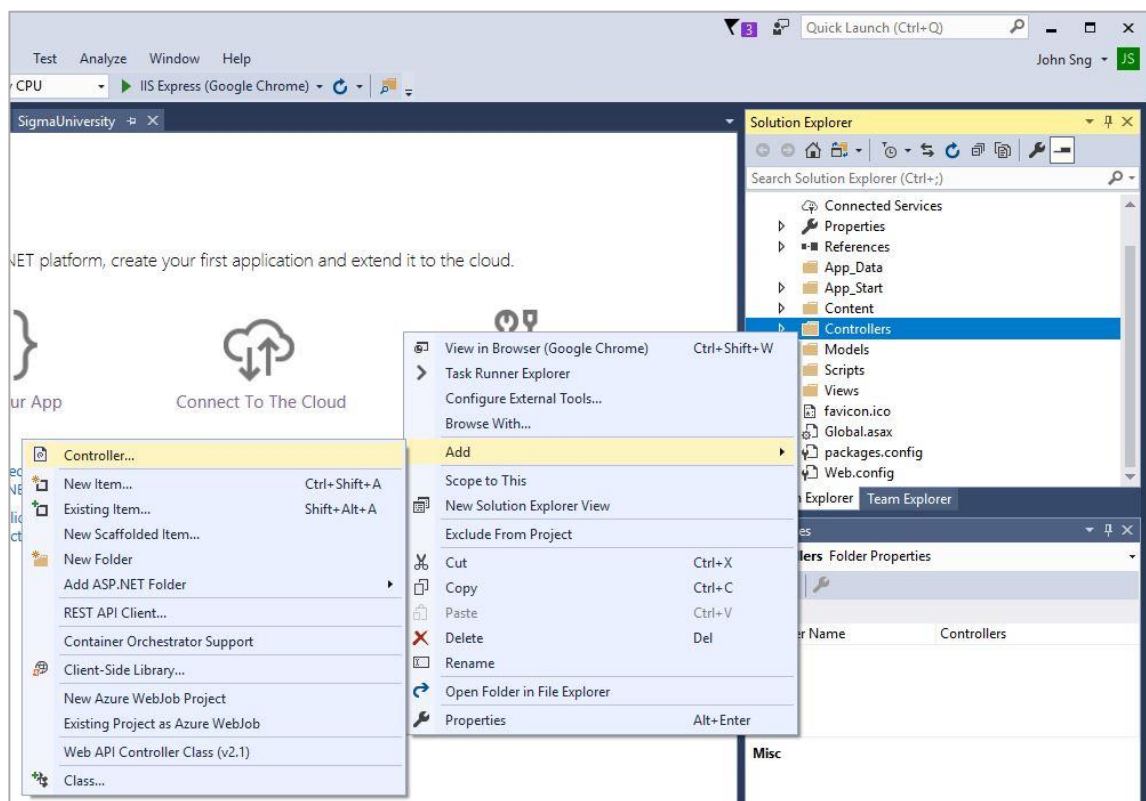
Module: Web Solutions Development

Module Code: IT5035FP

## Part (C): Create the Controller

In Part (C), students will learn to create the **Controller** in the MVC (Model-View-**Controller**) project. The Controller class created in Part (C) will be used for implementing the actual CRUD (Create Retrieve Update Delete) actions in Part (D).

- 1 To create a Controller class with actions for CRUD, go to the Solution Explorer panel and right-click on the **Controllers** folder. Select **Add** → **Controller**.

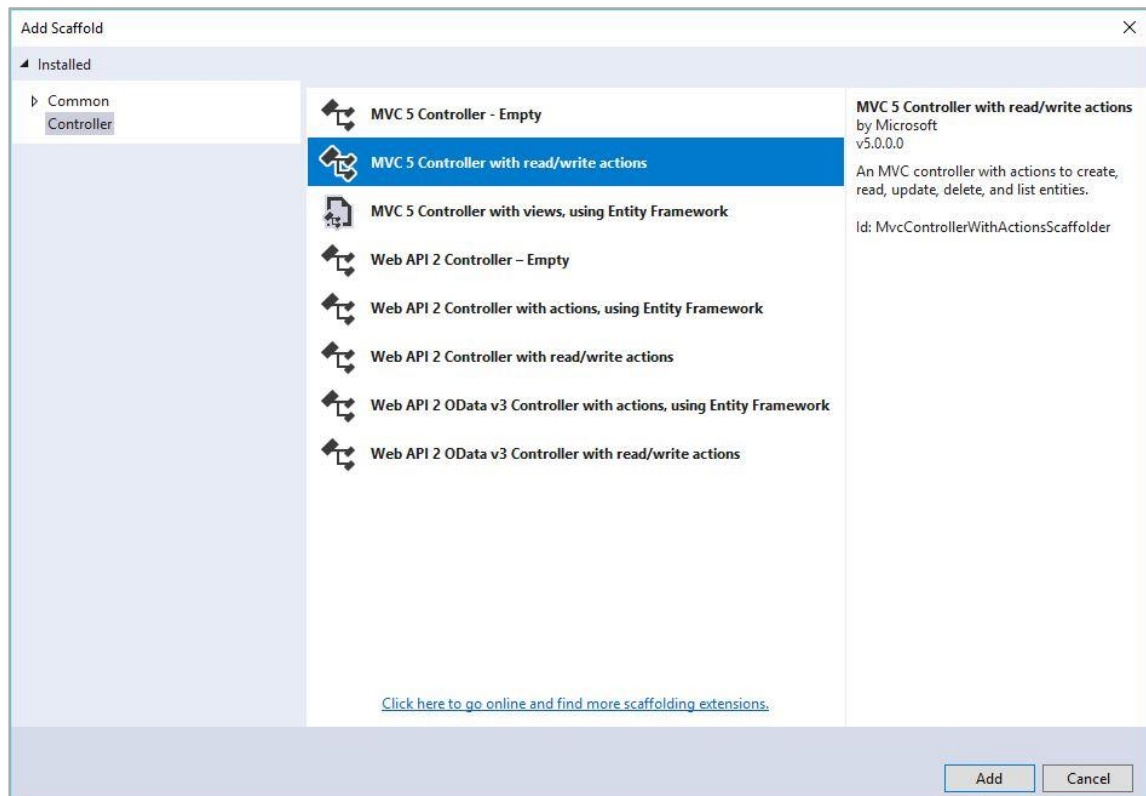


# LABSHEET 9

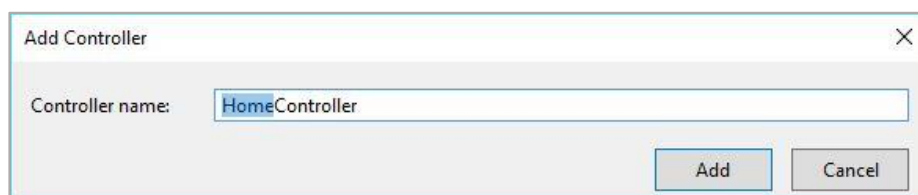
Module: Web Solutions Development

Module Code: IT5035FP

- 2 In the pop-up **Add Scaffold** window, select **MVC 5 Controller with read/write actions** to allow CRUD (Create, Retrieve, Update, Delete) and list operations. Click on the **Add** button.



- 3 Set the **Controller name** to **HomeController**. Click on the **Add** button.



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 4 Add the following code to create a **DataContext** object in order to access the database. Click on the **Lightbulb** icon to add the required class (i.e. SigmaUniversity.Models) into the code.

```
1  using SigmaUniversity.Models;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Web;
6  using System.Web.Mvc;
7
8  namespace SigmaUniversity.Controllers
9  {
10     public class HomeController : Controller
11     {
12         // GET: Home
13         DataContext db = new DataContext();
14         public ActionResult Index()
15         {
16             return View();
17         }
18     }
19 }
```

- 5 Click on the **Save All** button in the main menu.

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## Part (D): Create the View

In Part (D), students will learn to create the **View** in the MVC (Model-**View**-Controller) project. Students will be creating 5 different views and their corresponding actions in the Controller class to perform CRUD (Create Retrieve Update Delete) on the COLLEGE table. The 5 views will be done in 5 subparts:

- Part (D1): View to list all records (using the Retrieve operation)
- Part (D2): View to display details of a specific record (using the Retrieve operation)
- Part (D3): View to insert a new record (using the Create operation)
- Part (D4): View to edit a record (using the Update operation)
- Part (D5): View to delete a record (using the Delete operation)

## Part (D1): Create View to List All Records from COLLEGE Table

- 1 To list all records from COLLEGE table, add the following code to execute the SQL query in **HomeController.cs**. The **Index()** method executes the SQL query to retrieve the records (i.e. data), which are then passed back to the **View** to display.

```

1  using SigmaUniversity.Models;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Web;
6  using System.Web.Mvc;
7
8  namespace SigmaUniversity.Controllers
9  {
10     public class HomeController : Controller
11     {
12         // GET: Home
13         DataContext db = new DataContext();
14         public ActionResult Index()
15         {
16             var data = db.COLLEGE.SqlQuery("SELECT * FROM COLLEGE").ToList();
17             return View(data);
18         }
19     }
20 }

```

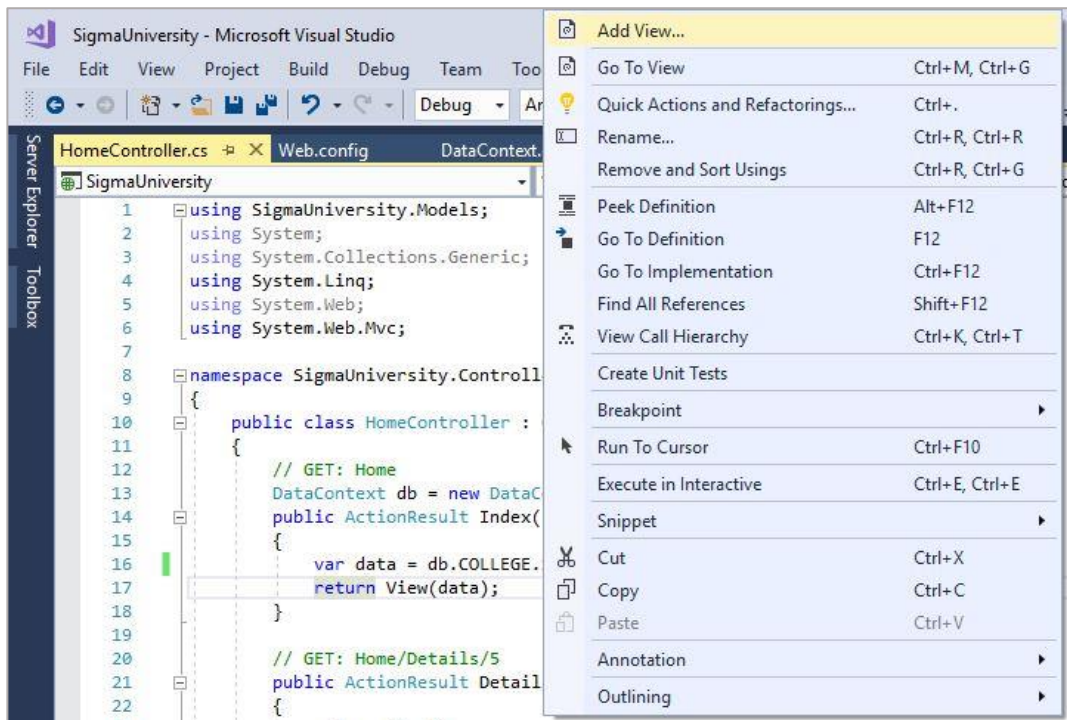


# LABSHEET 9

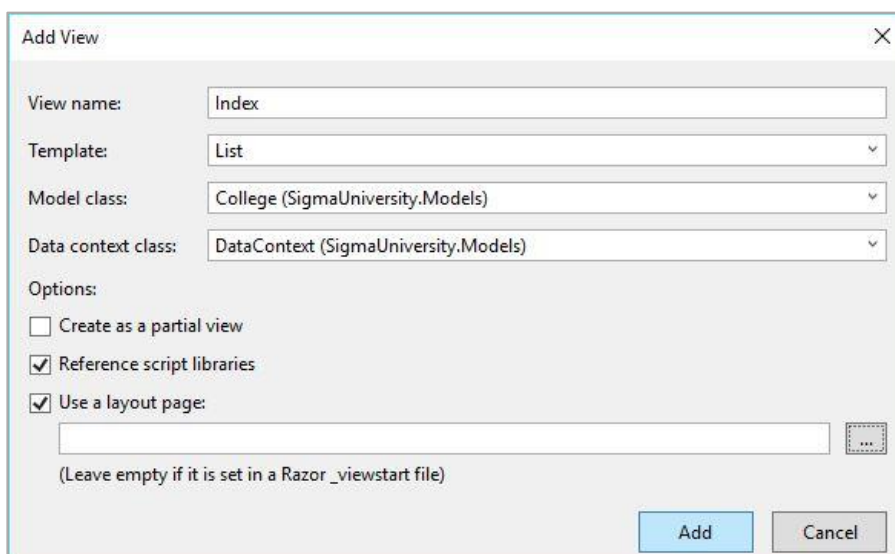
Module: Web Solutions Development

Module Code: IT5035FP

- 2 To create a View to list all records from COLLEGE table, right-click on the code editor window of **HomeController.cs** within the **Index() {...}** function. Select the **Add View** option.



- 3 In the pop-up **Add View** window, set the following fields:
- **Template:** List
  - **Model class:** College (SigmaUniversity.Models)
  - **Data context class:** DataContext (SigmaUniversity.Models)
- Click on the **Add** button.

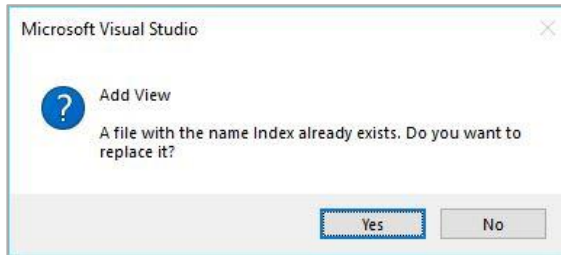


# LABSHEET 9

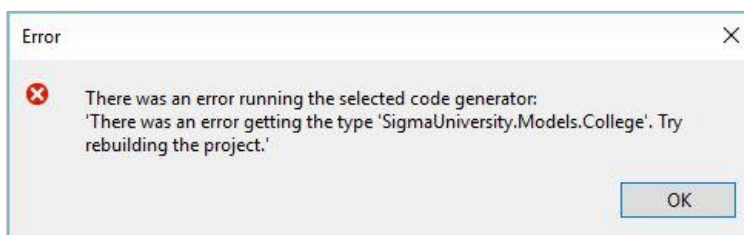
Module: Web Solutions Development

Module Code: IT5035FP

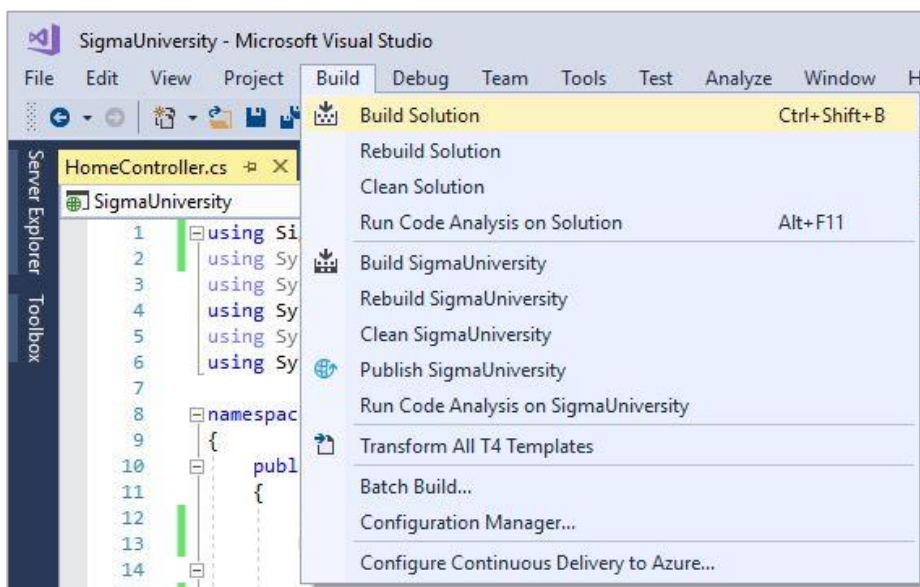
- 4 When you are prompted that an Index file already exists, click on the **Yes** button to replace it.



- 5 If you encounter an **Error** message that requests you to **try rebuilding the project**, click on the **OK** button. Then click on the **Cancel** button on the pop-up Add View window. (**Note: If you do not encounter this error message, proceed to Step 11 where an Index.cshtml file will be generated.**)



- 6 To solve the error, go to the main menu and select the **Build** option. Click on the **Build Solution** option.

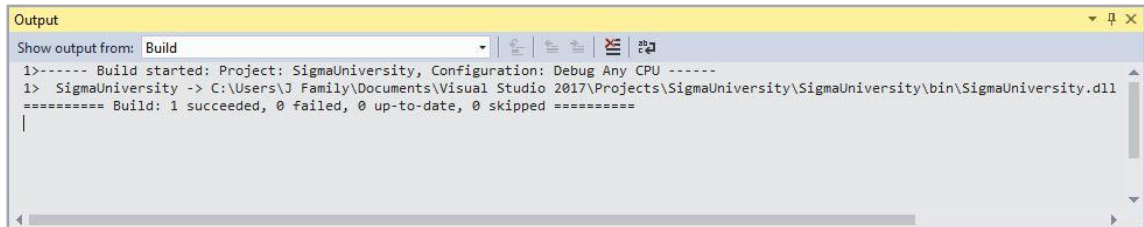


# LABSHEET 9

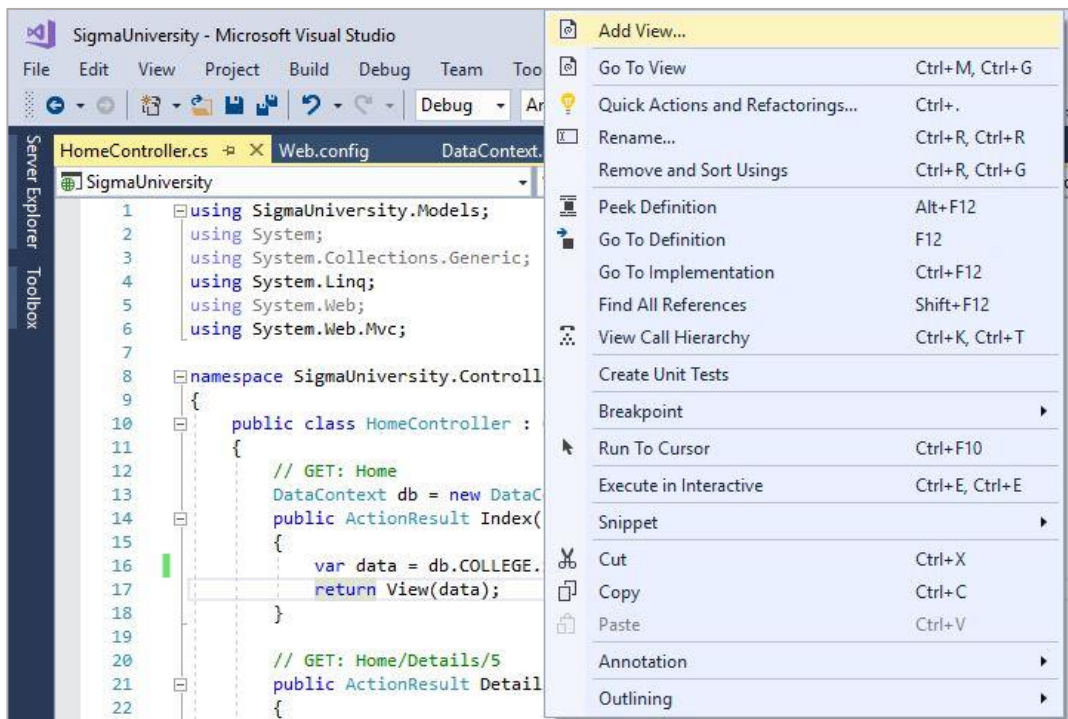
Module: Web Solutions Development

Module Code: IT5035FP

- 7 If the code build is successful, the following message will be show in the **Output** panel.



- 8 To create a View to list all records from COLLEGE table, right-click on the code editor window of **HomeController.cs** (e.g. within **Index() {...}** function). Select the **Add View** option.

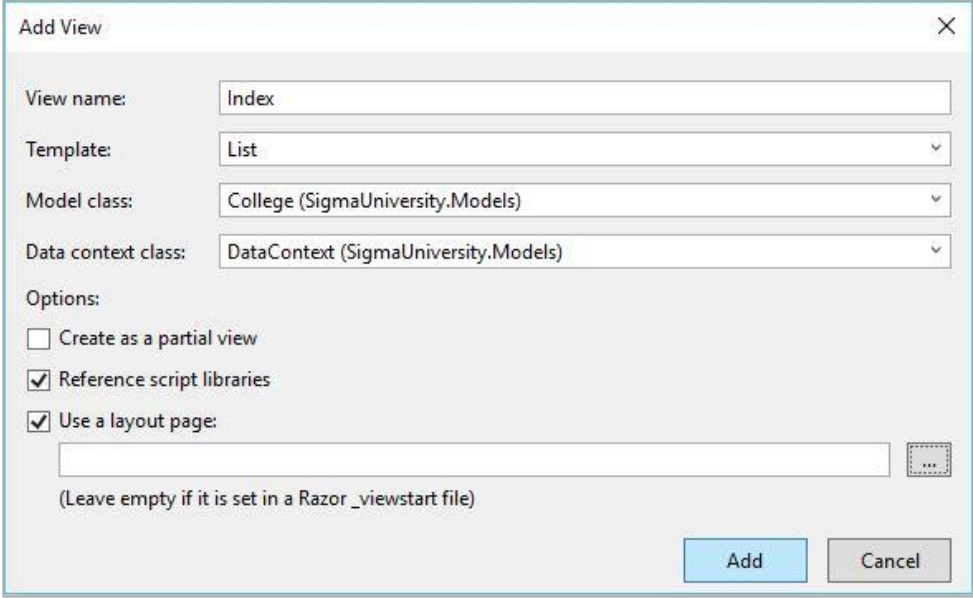


# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 9 In the pop-up **Add View** window, set the following fields:
- **Template:** List
  - **Model class:** College (SigmaUniversity.Models)
  - **Data context class:** DataContext (SigmaUniversity.Models)
- Click on the **Add** button.



Add View

View name: Index

Template: List

Model class: College (SigmaUniversity.Models)

Data context class: DataContext (SigmaUniversity.Models)

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

(Leave empty if it is set in a Razor \_viewstart file)

Add Cancel

- 10 When you are prompted that an Index file already exists, click on the **Yes** button to replace it.



Microsoft Visual Studio

Add View

A file with the name Index already exists. Do you want to replace it?

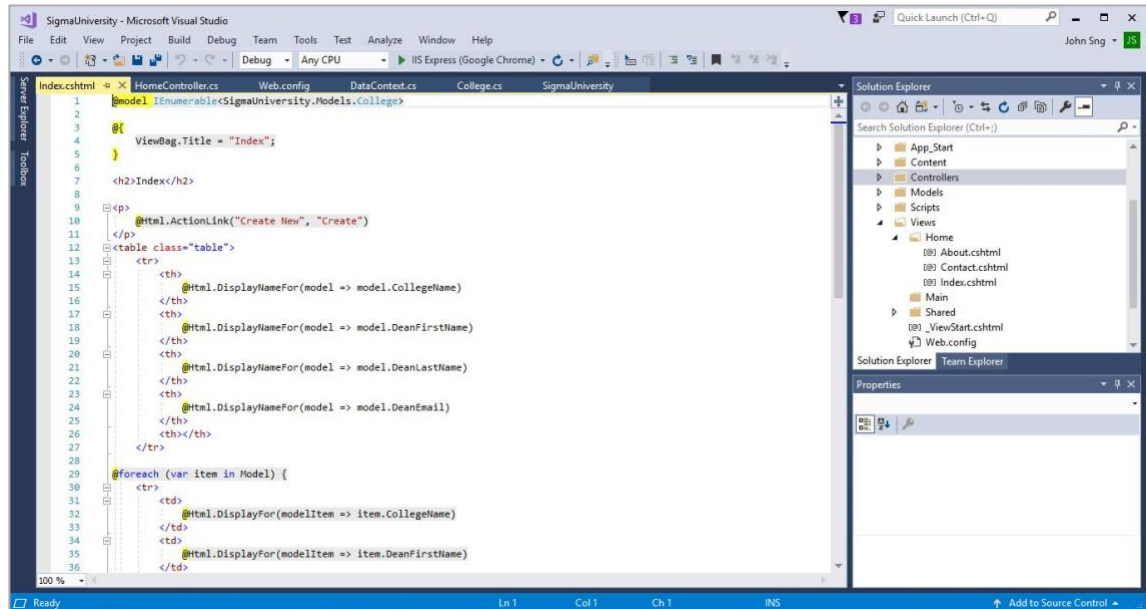
Yes No

# LABSHEET 9

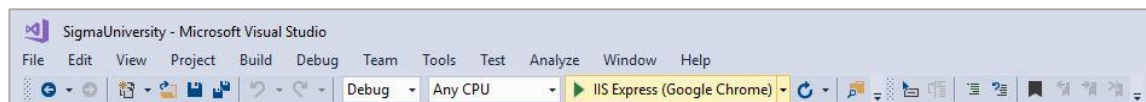
Module: Web Solutions Development

Module Code: IT5035FP

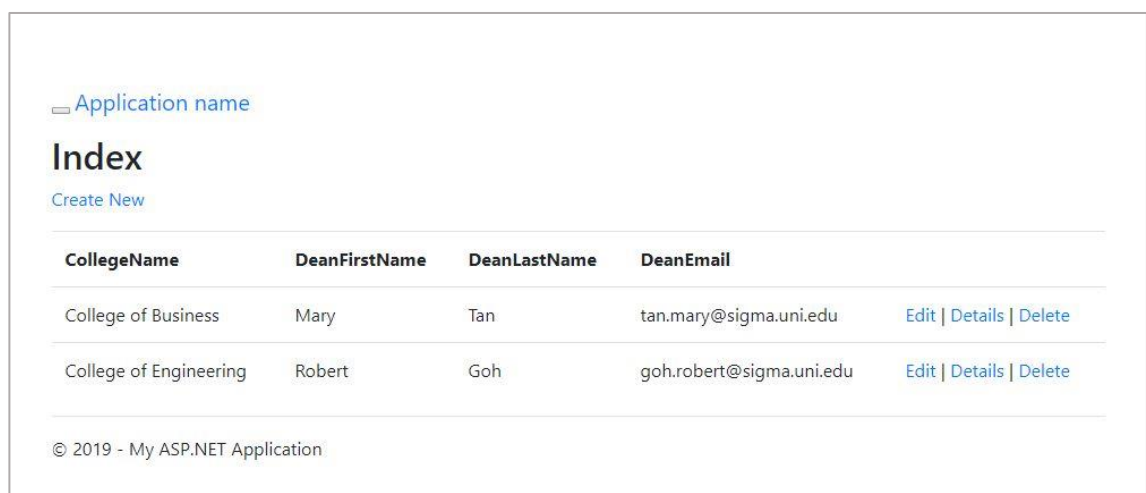
- 11 An ***Index.cshtml*** file will be generated.



- 12 To build and test ***Index.cshtml*** file, click on the **IIS Express (Google Chrome)** button.



- 13 If the build is successful, the Index page will be launched in your browser.



- 14 Click on the **Save All** button in the main menu.



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## Part (D2): Create View to Display Details of a Specific Record from COLLEGE Table

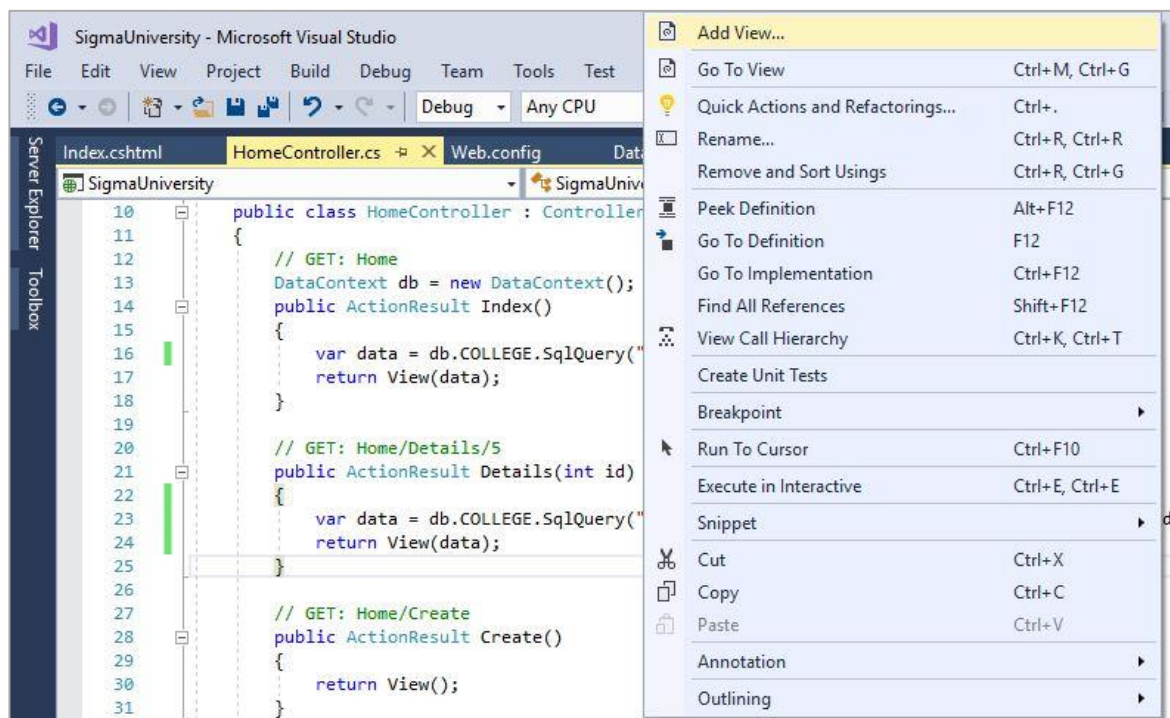
- 1 To display the details of specific records from COLLEGE Table, add the following code to execute the SQL query in **HomeController.cs**. The **Details(int id)** method receives the CollegeCode (i.e. int id) from the View, and uses it to execute the SQL query to retrieve the record based on the CollegeCode. The record (i.e. data) is then passed back to the **View** to display.

```

20 // GET: Home/Details/5
21 public ActionResult Details(int id)
22 {
23     var data = db.COLLEGE.SqlQuery("SELECT * FROM COLLEGE WHERE CollegeCode=@p0", id).SingleOrDefault();
24     return View(data);
25 }

```

- 2 To create a View to display details of a specific record from COLLEGE table, right-click on the code editor window of **HomeController.cs** within the **Details(int id) {...}** function. Select the **Add View** option.





# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 3 In the pop-up **Add View** window, set the following fields:
- **Template:** Details
  - **Model class:** College (SigmaUniversity.Models)
  - **Data context class:** DataContext (SigmaUniversity.Models)
- Click on the **Add** button.

**Add View**

View name:

Template:

Model class:

Data context class:

Options:

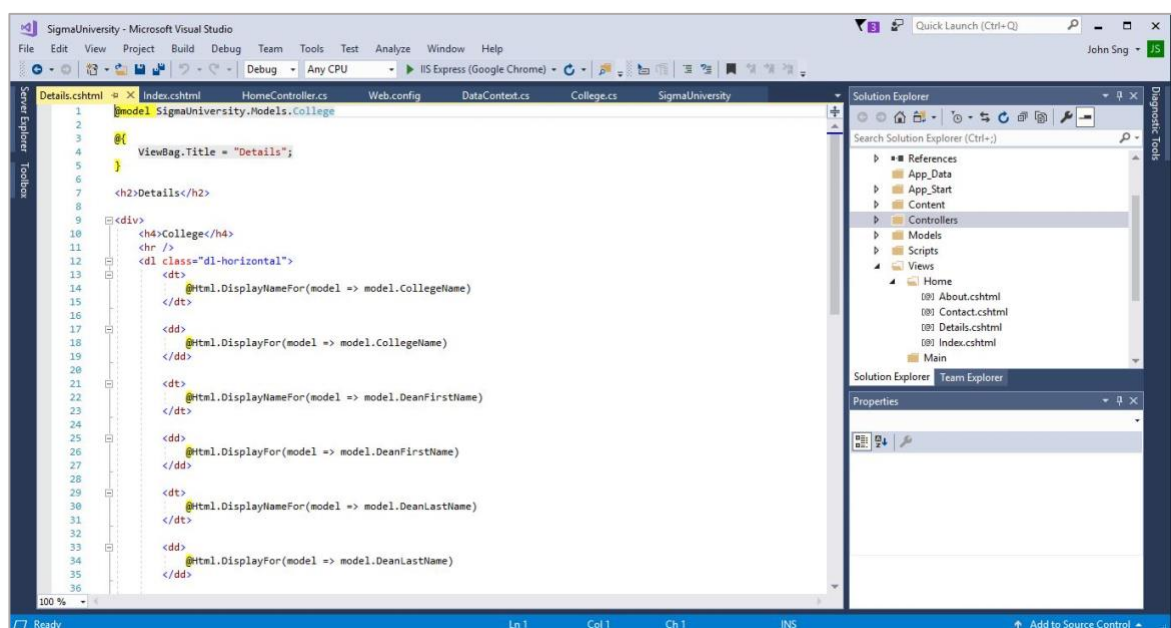
☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

(Leave empty if it is set in a Razor \_viewstart file)

- 4 A **Details.cshtml** file will be generated.

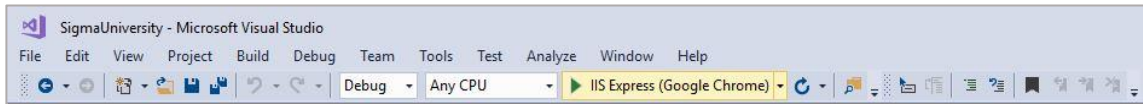


# LABSHEET 9

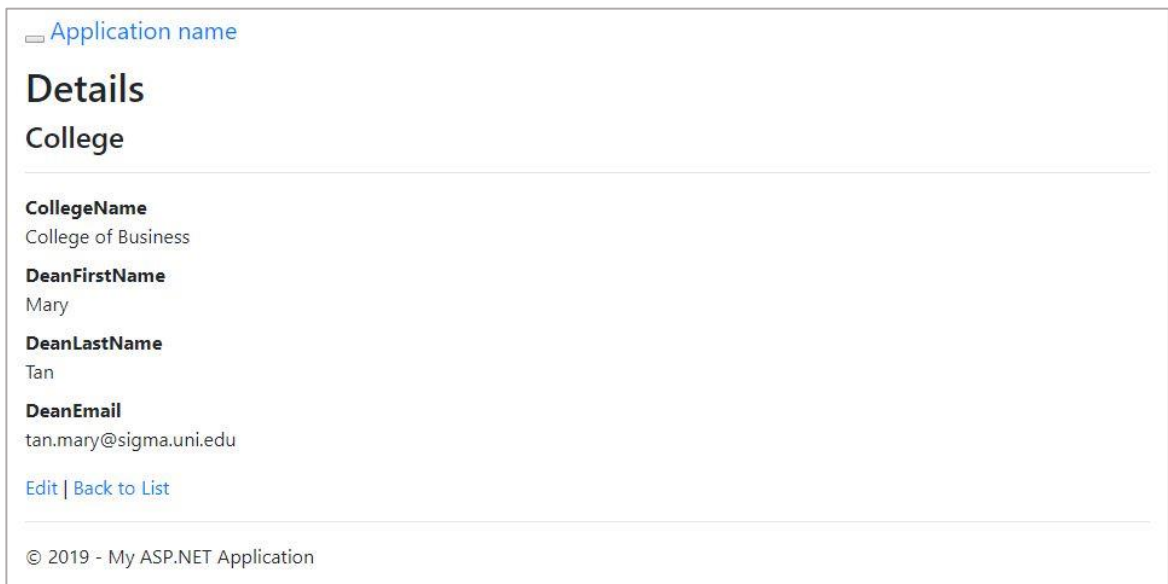
Module: Web Solutions Development

Module Code: IT5035FP

- 5 To build and test ***Details.cshtml*** file, click on the **IIS Express (Google Chrome)** button.



- 6 If the build is successful, the Index page will be launched in your browser. Click on the **Details** link of each record and you will see the Details page. Click on the **Back to List** link to return to the Index page.



- 7 Click on the **Save All** button in the main menu.

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## Part (D3): Create View to Insert a Record into COLLEGE Table

- 1 To insert a record into COLLEGE Table, add the following code to execute the SQL query in **HomeController.cs**. The **Create(FormCollection collection)** method receives the form data (i.e. FormCollection collection) from the View, and uses it to execute the SQL query to insert the record. Remember to modify the method to Create(**College** collection).

```

33 // POST: Home/Create
34 [HttpPost]
35 public ActionResult Create(College collection)
36 {
37     try
38     {
39         // Create a new list to store details of new record to be inserted
40         List<object> newRecord = new List<object>();
41         newRecord.Add(collection.CollegeCode);
42         newRecord.Add(collection.CollegeName);
43         newRecord.Add(collection.DeanFirstName);
44         newRecord.Add(collection.DeanLastName);
45         newRecord.Add(collection.DeanEmail);
46
47         // Copy record items into an array for easy addition to SQL statement
48         object[] recordItems = newRecord.ToArray();
49         int result = db.Database.ExecuteSqlCommand("INSERT INTO COLLEGE " +
50             "(CollegeCode, CollegeName, DeanFirstName, DeanLastName, DeanEmail) " +
51             "VALUES (@p0, @p1, @p2, @p3, @p4)", recordItems);
52
53         if (result > 0)
54         {
55             ViewBag.msg = "College record is added.";
56         }
57         return View();
58     }
59     catch
60     {
61         return View();
62     }
63 }

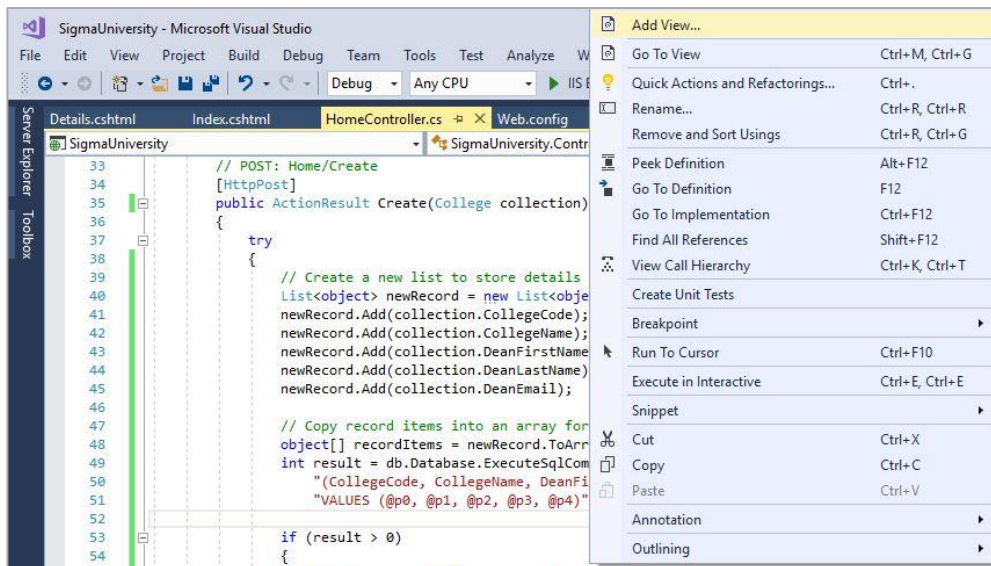
```

# LABSHEET 9

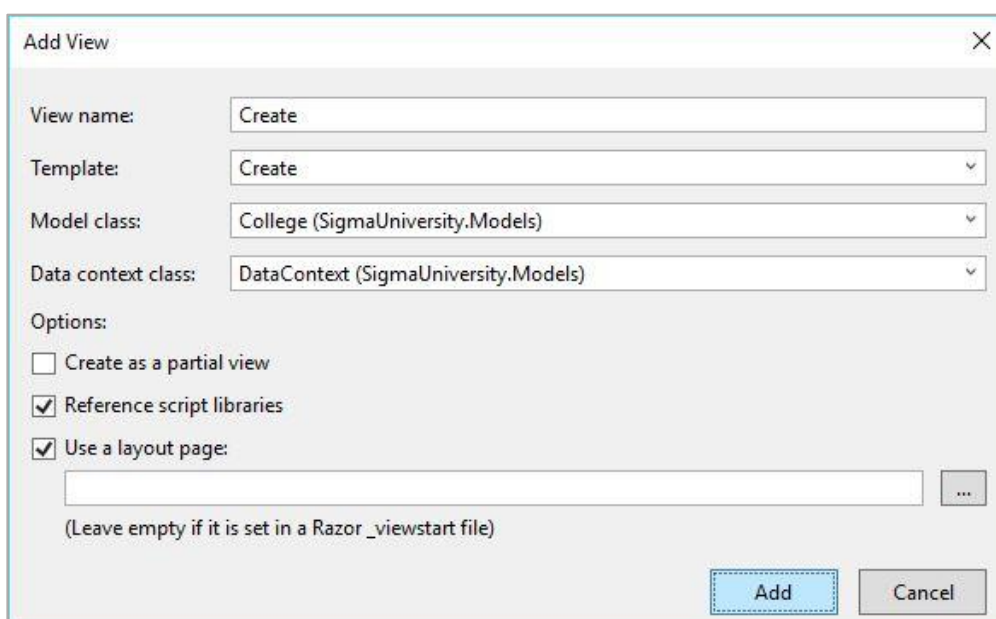
Module: Web Solutions Development

Module Code: IT5035FP

- 2 To create a View to insert a record into COLLEGE table, right-click on the code editor window of **HomeController.cs** within the **Create(College collection) {...}** function. Select the **Add View** option.



- 3 In the pop-up **Add View** window, set the following fields:
- **Template:** Create
  - **Model class:** College (SigmaUniversity.Models)
  - **Data context class:** DataContext (SigmaUniversity.Models)
- Click on the **Add** button.

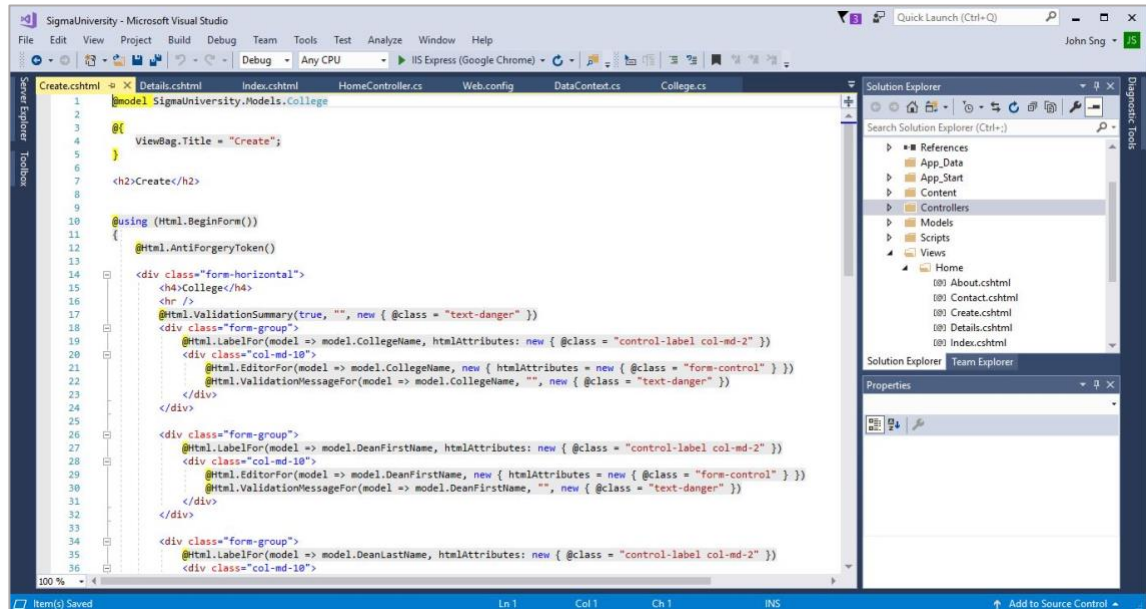


# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 4 A **Create.cshtml** file will be generated.



- 5 When you run the Create.cshtml file, you will observe that there are only 4 fields available for you to enter record details. The CollegeCode field is not available because it is set as the Primary Key. To add this field, add the following code into **Create.cshtml** after the line **@Html.ValidationSummary**, and before the **CollegeName** code block.

```

17 @Html.ValidationSummary(true, "", new { @class = "text-danger" })
18 <div class="form-group">
19     @Html.LabelFor(model => model.CollegeCode, htmlAttributes: new { @class = "control-label col-md-2" })
20     <div class="col-md-10">
21         @Html.EditorFor(model => model.CollegeCode, new { htmlAttributes = new { @class = "form-control" } })
22         @Html.ValidationMessageFor(model => model.CollegeCode, "", new { @class = "text-danger" })
23     </div>
24 </div>
25
26 <div class="form-group">
27     @Html.LabelFor(model => model.CollegeName, htmlAttributes: new { @class = "control-label col-md-2" })

```



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 6 To display the message “**College record is added.**” after a record has been created in COLLEGE table, add the following code into **Create.cshtml** before the **Create** button code block.

```

58 <div class="form-group">
59   <div class="col-md-10">
60     @if (ViewBag.msg != null)
61     {
62       <h5>@ViewBag.msg</h5>
63     }
64   </div>
65 </div>
66
67 <div class="form-group">
68   <div class="col-md-offset-2 col-md-10">
69     <input type="submit" value="Create" class="btn btn-default" />

```

- 7 To build and test **Create.cshtml** file, click on the **IIS Express (Google Chrome)** button.



- 8 If the build is successful, the Create page will be launched in your browser. Enter the details of the new record to be created. Click on the **Create** link to create the new record.

 A screenshot of a web browser displaying a form titled 'Create College'. The form has several input fields: 'CollegeCode' with the value '300', 'CollegeName' with 'College of Arts', 'DeanFirstName' with 'Joseph', 'DeanLastName' with 'Lee', and 'DeanEmail' with 'lee.joseph@sigma.uni.edu'. Below the fields is a 'Create' button and a 'Back to List' link. At the bottom of the page, it says '© 2019 - My ASP.NET Application'.



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 9 The Create page will display the message “***College record is added.***” to show that the record has been successfully created.

Application name

## Create College

CollegeCode  
300

CollegeName  
College of Arts

DeanFirstName  
Joseph

DeanLastName  
Lee

DeanEmail  
lee.joseph@sigma.uni.edu

**College record is added.**

Create

[Back to List](#)

© 2019 - My ASP.NET Application

- 10 Click the **Back to List** link to display the COLLEGE table records.

Application name

## Index

[Create New](#)

CollegeName	DeanFirstName	DeanLastName	DeanEmail	
College of Business	Mary	Tan	tan.mary@sigma.uni.edu	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
College of Engineering	Robert	Goh	goh.robert@sigma.uni.edu	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
College of Arts	Joseph	Lee	lee.joseph@sigma.uni.edu	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

© 2019 - My ASP.NET Application

- 11 Click on the **Save All** button in the main menu.

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## Part (D4): Create View to Edit a Record in COLLEGE Table

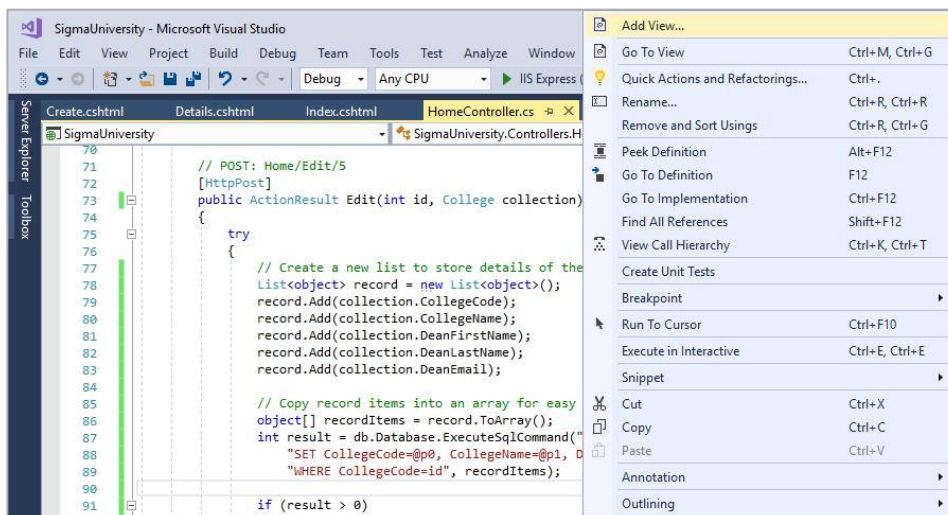
- 1 To edit a record in COLLEGE Table, add the following code to execute the SQL query in **HomeController.cs**. The **Edit(int id, FormCollection collection)** method receives the form data (i.e. int id, FormCollection collection) from the View, and uses it to execute the SQL query to update the record. Remember to modify the method to Edit(int id, **College** collection).

```

72 // POST: Home/Edit/5
73 [HttpPost]
74 public ActionResult Edit(int id, College collection)
75 {
76     try
77     {
78         // Create a new list to store details of the record to be updated
79         List<object> record = new List<object>();
80         record.Add(collection.CollegeCode);
81         record.Add(collection.CollegeName);
82         record.Add(collection.DeanFirstName);
83         record.Add(collection.DeanLastName);
84         record.Add(collection.DeanEmail);
85
86         // Copy record items into an array for easy addition to SQL statement
87         object[] recordItems = record.ToArray();
88         int result = db.Database.ExecuteSqlCommand("UPDATE COLLEGE " +
89             "SET CollegeCode=@p0, CollegeName=@p1, DeanFirstName=@p2, DeanLastName=@p3, DeanEmail=@p4 " +
90             "WHERE CollegeCode=" + id, recordItems);
91
92         if (result > 0)
93         {
94             ViewBag.msg = "College record is updated.";
95         }
96         return View();
97     }
98     catch
99     {
100         return View();
101     }
102 }

```

- 2 To create a View to edit a record in COLLEGE table, right-click on the code editor window of **HomeController.cs** within the **Edit(int id, College collection) {...}** function. Select the **Add View** option.

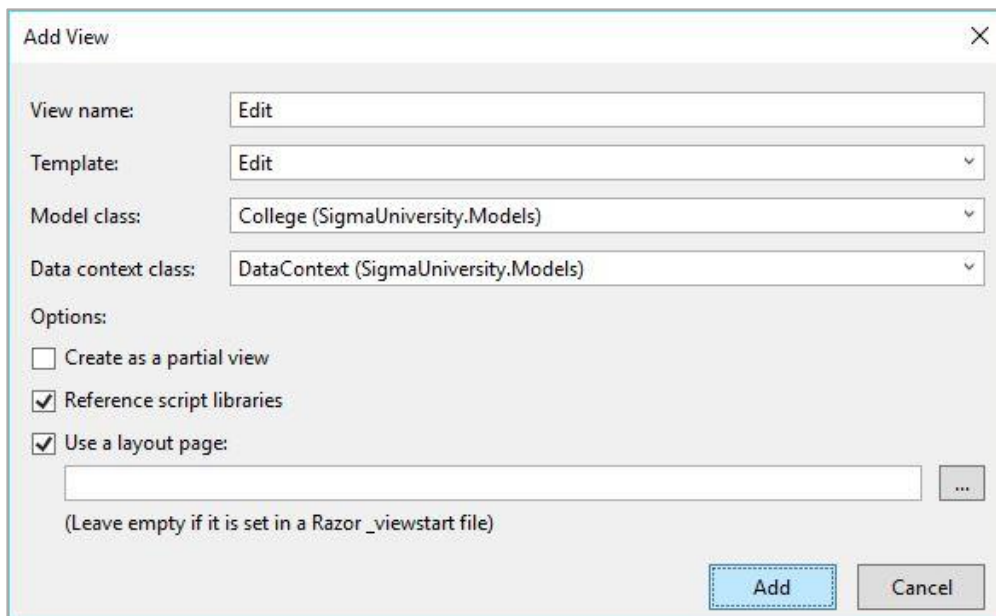


# LABSHEET 9

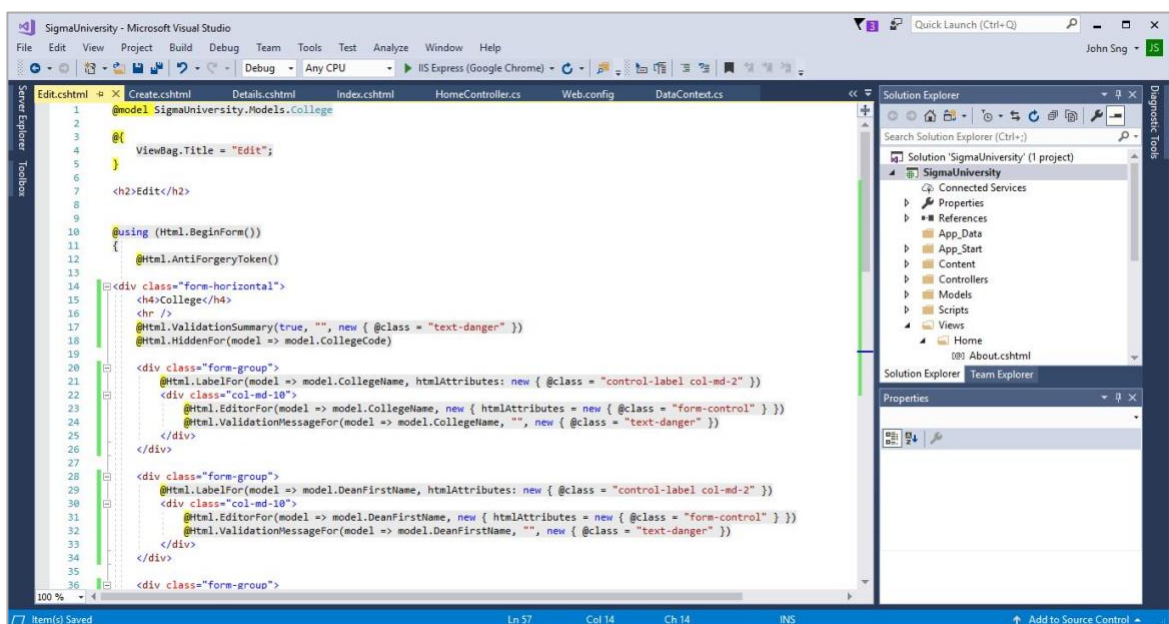
Module: Web Solutions Development

Module Code: IT5035FP

- 3 In the pop-up **Add View** window, set the following fields:
- **Template:** Edit
  - **Model class:** College (SigmaUniversity.Models)
  - **Data context class:** DataContext (SigmaUniversity.Models)
- Click on the **Add** button.



- 4 An **Edit.cshtml** file will be generated.



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 5 When you run the `Edit.cshtml` file, you will observe that there are only 4 fields available for you to edit record details. The `CollegeCode` field is not available because it is set as the Primary Key. To add this field, add the following code into ***Edit.cshtml*** after the line ***@Html.ValidationSummary***, and before the ***CollegeName*** code block.

```

17 @Html.ValidationSummary(true, "", new { @class = "text-danger" })
18
19 <div class="form-group">
20     @Html.LabelFor(model => model.CollegeCode, htmlAttributes: new { @class = "control-label col-md-2" })
21     <div class="col-md-10">
22         @Html.EditorFor(model => model.CollegeCode, new { htmlAttributes = new { @class = "form-control" } })
23         @Html.ValidationMessageFor(model => model.CollegeCode, "", new { @class = "text-danger" })
24     </div>
25 </div>
26
27 <div class="form-group">
28     @Html.LabelFor(model => model.CollegeName, htmlAttributes: new { @class = "control-label col-md-2" })

```

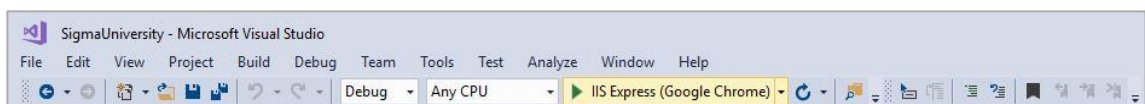
- 6 To display the message "***College record is updated.***" after a record has been updated in COLLEGE table, add the following code into ***Edit.cshtml*** before the ***Save*** button code block.

```

52 <div class="form-group">
53     <div class="col-md-10">
54         @if (ViewBag.msg != null)
55         {
56             <h5>@ViewBag.msg</h5>
57         }
58     </div>
59 </div>
60
61 <div class="form-group">
62     <div class="col-md-offset-2 col-md-10">
63         <input type="submit" value="Save" class="btn btn-default" />

```

- 7 To build and test ***Edit.cshtml*** file, select the ***HomeController.cs*** tab and click on the ***IIS Express (Google Chrome)*** button.



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 8 If the build is successful, the Index page will be launched in your browser. Click on the **Edit** link of any record to launch the Edit page.

Application name

## Edit College

CollegeCode

CollegeName

DeanFirstName

DeanLastName

DeanEmail

Save

[Back to List](#)

© 2019 - My ASP.NET Application

- 9 Observe that the text boxes on the Edit page are empty. To fill the text boxes with record details, add the following code to **HomeController.cs**.

```

65 // GET: Home/Edit/5
66 public ActionResult Edit(int id)
67 {
68     var data = db.COLLEGE.SqlQuery("SELECT * FROM COLLEGE WHERE CollegeCode=@p0", id).SingleOrDefault();
69     return View(data);
70 }

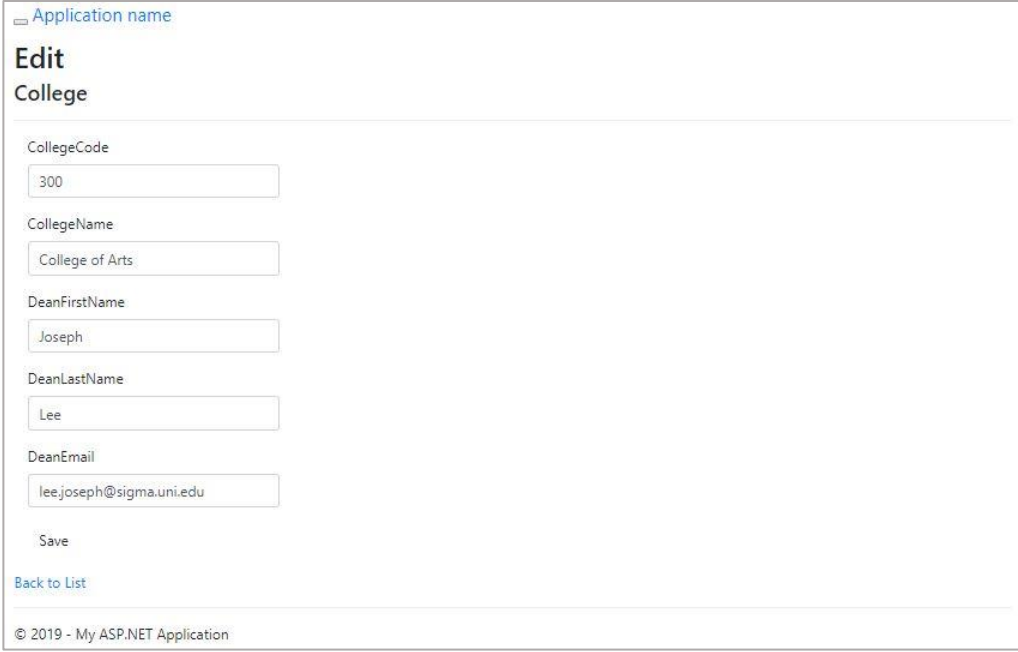
```

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 10 Click on the **IIS Express (Google Chrome)** button. The Index page will be launched. Click on the **Edit** link for the **College of Arts** record. The Edit page will be launched with record details.



Application name

## Edit College

CollegeCode  
300

CollegeName  
College of Arts

DeanFirstName  
Joseph

DeanLastName  
Lee

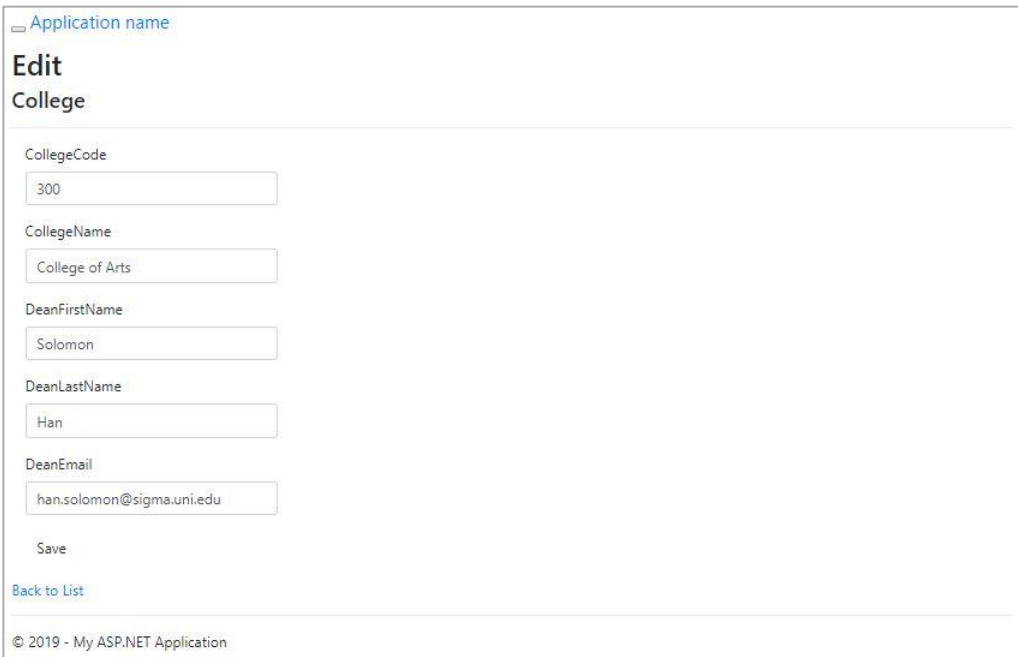
DeanEmail  
lee.joseph@sigma.uni.edu

Save

[Back to List](#)

© 2019 - My ASP.NET Application

- 11 Edit the fields for **DeanFirstName**, **DeanLastName**, and **DeanEmail** as shown. **Note that if you edit CollegeName, you must update the table constraints for COLLEGE table first.** Click on the **Save** link.



Application name

## Edit College

CollegeCode  
300

CollegeName  
College of Arts

DeanFirstName  
Solomon

DeanLastName  
Han

DeanEmail  
han.solomon@sigma.uni.edu

Save

[Back to List](#)

© 2019 - My ASP.NET Application



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 12 The Edit page will display the message “**College record is updated.**” to show that the record has been successfully updated.

The screenshot shows the 'Edit College' page. At the top, there is a header with 'Application name' and a hamburger menu icon. Below the header, the title 'Edit College' is displayed. The form contains several input fields: 'CollegeCode' with the value '300', 'CollegeName' with 'College of Arts', 'DeanFirstName' with 'Solomon', 'DeanLastName' with 'Han', and 'DeanEmail' with 'han.solomon@sigma.uni.edu'. Below these fields, a green message box states 'College record is updated.' There is a 'Save' button and a 'Back to List' link. At the bottom, the footer reads '© 2019 - My ASP.NET Application'.

- 13 Click the **Back to List** link to display the updated COLLEGE table records.

The screenshot shows the 'Index' page. At the top, there is a header with 'Application name' and a hamburger menu icon. Below the header, the title 'Index' is displayed, followed by a 'Create New' link. The main content is a table with the following columns: 'CollegeName', 'DeanFirstName', 'DeanLastName', 'DeanEmail', and a set of action links ('Edit | Details | Delete'). The table contains three rows of data:

CollegeName	DeanFirstName	DeanLastName	DeanEmail	
College of Business	Mary	Tan	tan.mary@sigma.uni.edu	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
College of Engineering	Robert	Goh	goh.robert@sigma.uni.edu	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
College of Arts	Solomon	Han	han.solomon@sigma.uni.edu	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

At the bottom, the footer reads '© 2019 - My ASP.NET Application'.

- 14 Click on the **Save All** button in the main menu.

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## Part (D5): Create View to Delete a Record in COLLEGE Table

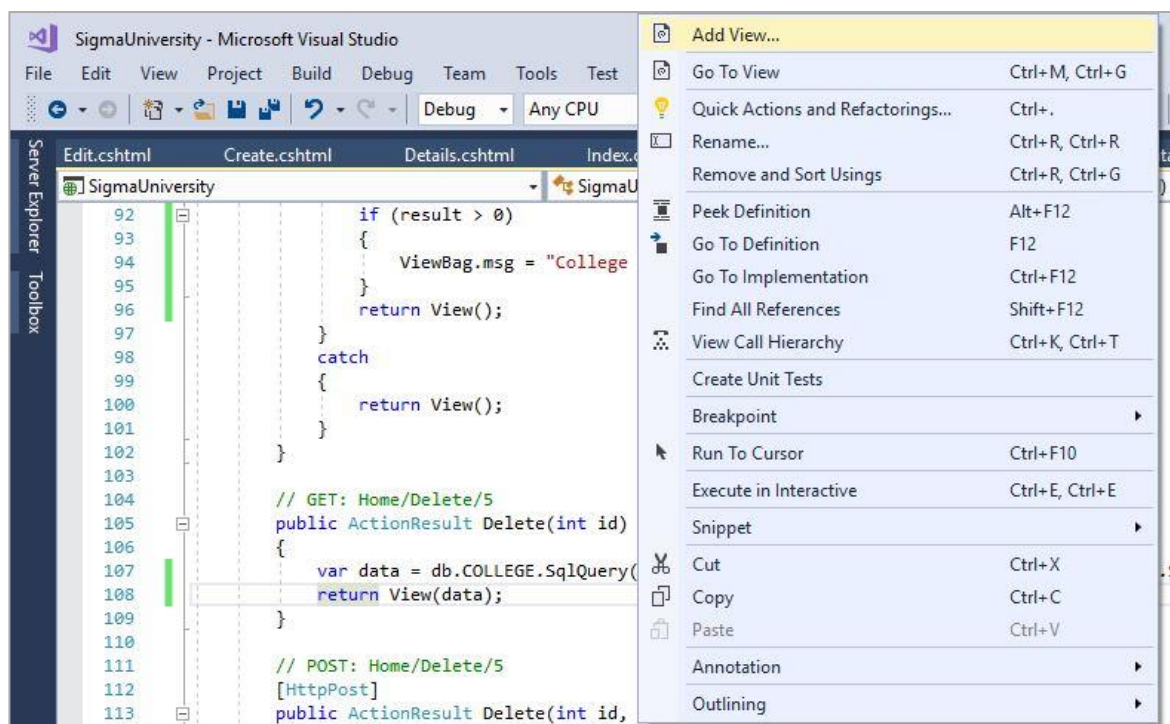
- 1 To view the record in COLLEGE table that is to be deleted, add the following code to execute the SQL query in **HomeController.cs**. The **Delete(int id)** method receives the form data (i.e. int id) from the View, and uses it to execute the SQL query to display the record that is to be deleted.

```

104 // GET: Home/Delete/5
105 public ActionResult Delete(int id)
106 {
107     var data = db.COLLEGE.SqlQuery("SELECT * FROM COLLEGE WHERE CollegeCode=@p0", id).SingleOrDefault();
108     return View(data);
109 }

```

- 2 To create a View to delete a record in COLLEGE table, right-click on the code editor window of **HomeController.cs** within the **Delete(int id) {...}** function. Select the **Add View** option.



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 3 In the pop-up **Add View** window, set the following fields:
- **Template:** Delete
  - **Model class:** College (SigmaUniversity.Models)
  - **Data context class:** DataContext (SigmaUniversity.Models)
- Click on the **Add** button.

**Add View**

View name:

Template:

Model class:

Data context class:

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

(Leave empty if it is set in a Razor \_viewstart file)

- 4 A **Delete.cshtml** file will be generated.

```

1 @model SigmaUniversity.Models.College
2
3 @{
4     ViewBag.Title = "Delete";
5 }
6
7 <h2>Delete</h2>
8
9 <h3>Are you sure you want to delete this?</h3>
10
11 <div>
12     <h4>College</h4>
13     <hr />
14     <dl class="dl-horizontal">
15         <dt>
16             @Html.DisplayNameFor(model => model.CollegeName)
17         </dt>
18         <dd>
19             @Html.DisplayFor(model => model.CollegeName)
20         </dd>
21     </dl>
22     <dt>
23         @Html.DisplayNameFor(model => model.DeanFirstName)
24     </dt>
25     <dd>
26         @Html.DisplayFor(model => model.DeanFirstName)
27     </dd>
28     <dt>
29         @Html.DisplayNameFor(model => model.DeanLastName)
30     </dt>
31     <dd>
32         @Html.DisplayFor(model => model.DeanLastName)
33     </dd>
34     <dt>
35         @Html.DisplayNameFor(model => model.DeanLastName)
36     </dt>
37     <dd>
38         @Html.DisplayFor(model => model.DeanLastName)
39     </dd>
40 </dl>
41 </div>

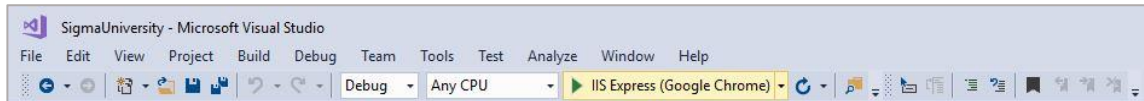
```

# LABSHEET 9

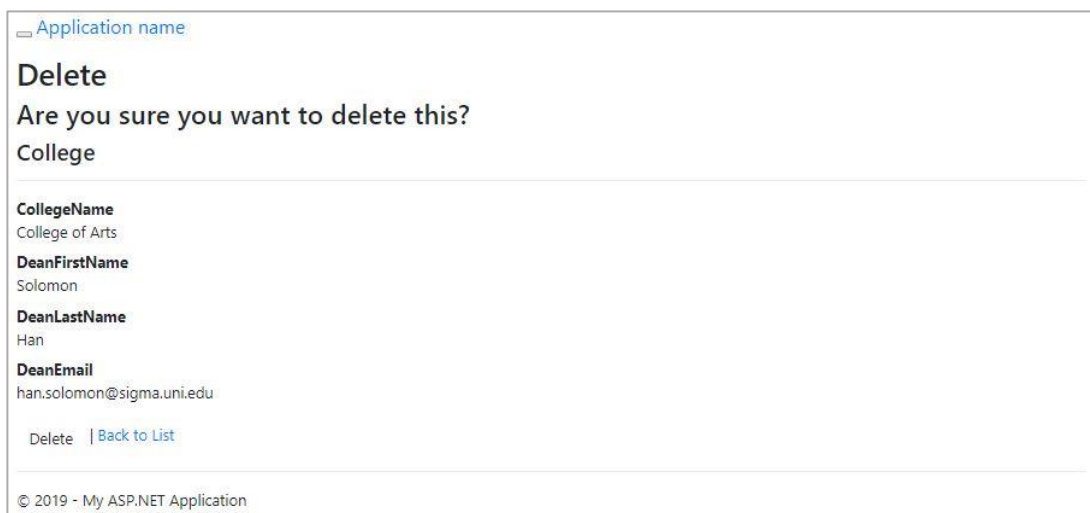
Module: Web Solutions Development

Module Code: IT5035FP

- 5 To build and test **Delete.cshtml** file, select the **HomeController.cs** tab and click on the **IIS Express (Google Chrome)** button.



- 6 If the build is successful, the Index page will be launched in your browser. Click on the **Delete** link of any record to launch the Delete page. The details of the record to be deleted is displayed on the page. If you click on the **Delete** link, the record will not be deleted.



- 7 To delete the record in COLLEGE Table, add the following code to execute the SQL query in **HomeController.cs**. The **Delete(int id, FormCollection collection)** method receives the form data (i.e. int id, FormCollection collection) from the View, and uses it to execute the SQL query to display the record that is to be deleted. Remember to modify the method to Delete(int id, **College** collection).

```

111 // POST: Home/Delete/5
112 [HttpPost]
113 public ActionResult Delete(int id, College collection)
114 {
115     try
116     {
117         // SQL statement to delete a record
118         int result = db.Database.ExecuteSqlCommand("DELETE FROM COLLEGE WHERE CollegeCode=@p0", id);
119
120         // If record is deleted successfully, return to Index page to show updated COLLEGE table
121         if (result > 0)
122         {
123             return RedirectToAction("Index");
124         }
125         return View();
126     }
127     catch
128     {
129         return View();
130     }
131 }

```

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 8 Click on the **IIS Express (Google Chrome)** button. The Index page will be launched. Click on the **Delete** link for the **College of Arts** record. The Delete page will be launched with record details. Click on the **Delete** link to delete the record.

[Application name](#)

## Delete

Are you sure you want to delete this?

College

---

**CollegeName**  
College of Arts

**DeanFirstName**  
Solomon

**DeanLastName**  
Han

**DeanEmail**  
han.solomon@sigma.uni.edu

[Delete](#) | [Back to List](#)

---

© 2019 - My ASP.NET Application

- 9 The updated Index page will show that the record has been deleted.

[Application name](#)

## Index

[Create New](#)

CollegeName	DeanFirstName	DeanLastName	DeanEmail	
College of Business	Mary	Tan	tan.mary@sigma.uni.edu	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
College of Engineering	Robert	Goh	goh.robert@sigma.uni.edu	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

---

© 2019 - My ASP.NET Application

# LABSHEET 9

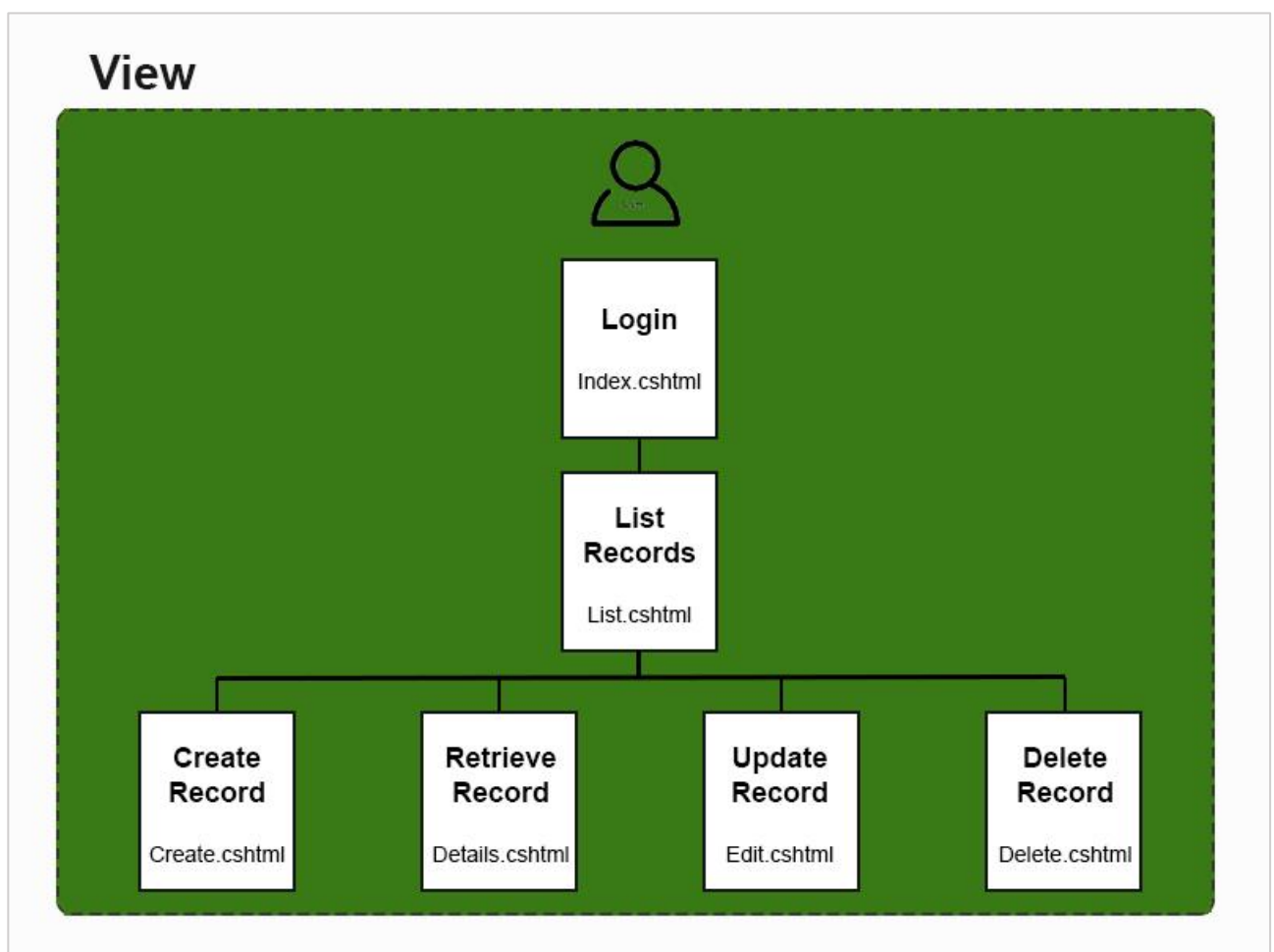
Module: Web Solutions Development

Module Code: IT5035FP

## Part (E): Create an Administrator Login Page

In Part (E), students will learn to create an Administrator login page. This login page will be the home page (i.e. Index page) where the Administrator will login with a Staff ID and Password in order to access the CRUD pages created in Part (D). Part (E) will be done in 4 subparts:

- Part (E1): Create ITADMIN table for Administrator accounts
- Part (E2): Create the Model for ITADMIN table
- Part (E3): Update the Controller to include the Login action
- Part (E4): Create the View for Login page



Addition of Login Page



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## Part (E1): Create ITADMIN Table for Administrator Accounts

- To create Administrator accounts, add a new **ITADMIN** table in SigmaUniversity database using the following data dictionary. You can create the table using the GUI tools in Microsoft SQL Server Management Studio.

ITADMIN Table

Column Name	Data Type	Key	Null Status
StaffID	int	Primary Key	NOT NULL
Password	nvarchar(MAX)		NOT NULL

Alternatively, you can create the ITADMIN table using the following SQL commands.

```

/* ***** */
/* 1. Create ITADMIN table */
/* ***** */
CREATE TABLE ITADMIN (
    StaffID          INT          NOT NULL,
    "Password"       NVARCHAR(MAX) NOT NULL,
    CONSTRAINT StaffIDPK PRIMARY KEY (StaffID)
);

```

- Create the following entries in the **ITADMIN** table. You can create the entries using the GUI tools in Microsoft SQL Server Management Studio.

ITADMIN Table

StaffID	Password
90001	Ironman
90002	Spiderman

Alternatively, you can create the entries using the following SQL commands.

```

/* ***** */
/* 2. Populate ITADMIN table */
/* ***** */
INSERT INTO ITADMIN (StaffID, "Password")
VALUES (90001, 'Ironman');
INSERT INTO ITADMIN (StaffID, "Password")
VALUES (90002, 'Spiderman');

```

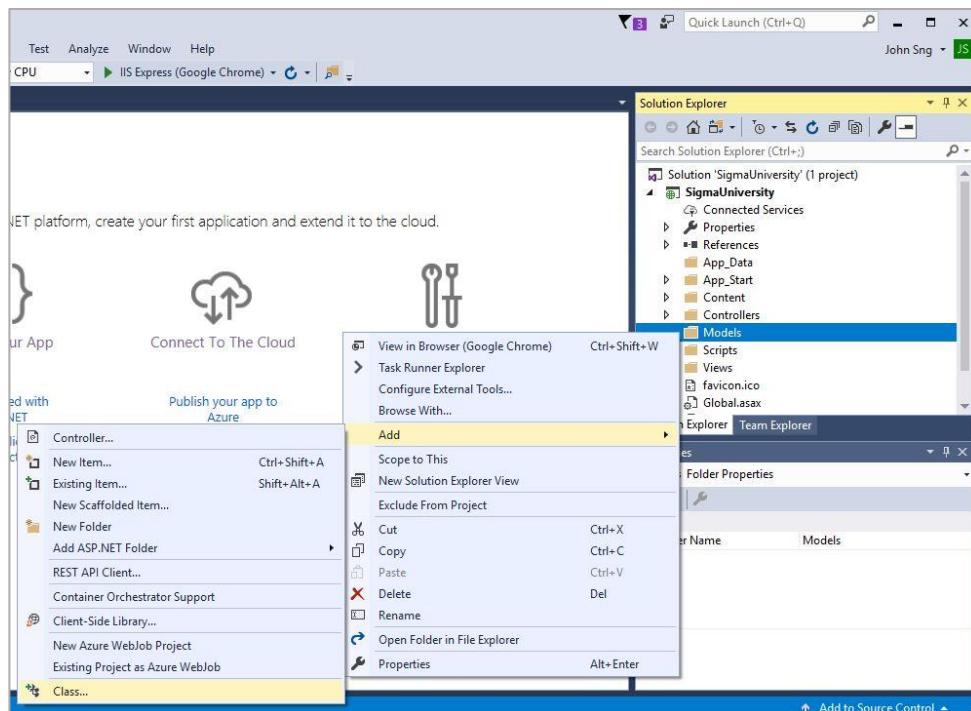
# LABSHEET 9

Module: Web Solutions Development

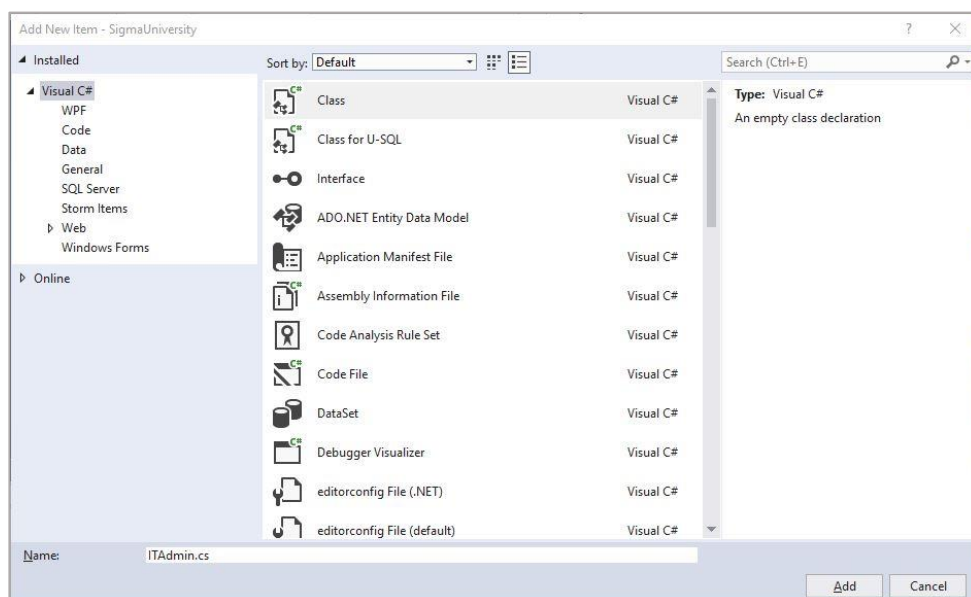
Module Code: IT5035FP

## Part (E2): Create the Model for ITADMIN Table

- 1 To create a Model class, go to the Solution Explorer panel and right-click on the **Models** folder. Select **Add → Class**.



- 2 In the pop-up **Add New Item** window, select **Visual C# -> Class** item. Set the **Name** to **ITAdmin.cs**. Click on the **Add** button.



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 3 Visual Studio will generate the codes to create the **ITAdmin** class. Enter the following codes. When a Lightbulb icon appears on the left to suggest the required class that needs to be added, click on the **Lightbulb** icon and select the class to add.

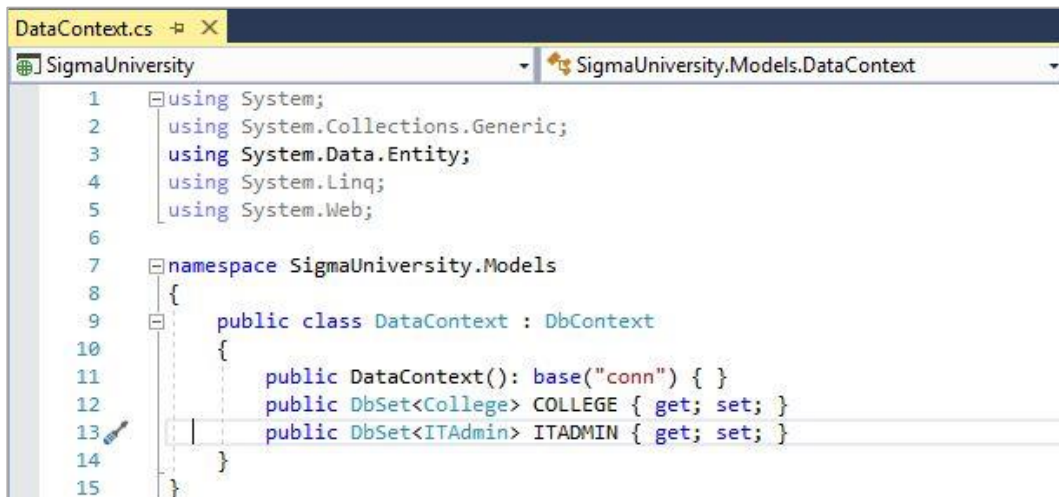


```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel.DataAnnotations;
4  using System.ComponentModel.DataAnnotations.Schema;
5  using System.Linq;
6  using System.Web;
7
8  namespace SigmaUniversity.Models
9  {
10     [Table("ITADMIN")]
11     public class ITAdmin
12     {
13         [Key]
14         public int StaffID { get; set; }
15         public string Password { get; set; }
16     }
17 }

```

- 4 Click on the **Save All** button in the main menu.
- 5 Open the **DataContext.cs** file. Add the code to define the interaction with the ITAdmin table.



```

1  using System;
2  using System.Collections.Generic;
3  using System.Data.Entity;
4  using System.Linq;
5  using System.Web;
6
7  namespace SigmaUniversity.Models
8  {
9      public class DataContext : DbContext
10     {
11         public DataContext(): base("conn") { }
12         public DbSet<College> COLLEGE { get; set; }
13         public DbSet<ITAdmin> ITADMIN { get; set; }
14     }
15 }

```

- 6 Click on the **Save All** button in the main menu.

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## Part (E3): Update the Controller to include the Login action

As the new Login page will take over as the home page (i.e. Index page), you need to rename the current Index page (e.g. from Index to List). To do this, you will need to rename the `Index()` method in the Controller to `List()`, and then generate a new List view. After that, you will create a new `Index()` method for the Login page.

- 1 Open the **HomeController.cs** file. Rename the **Index()** method to **List()** method.

```
// GET: List
public ActionResult List()
{
    var data = db.COLLEGE.SqlQuery("SELECT * FROM COLLEGE").ToList();
    return View(data);
}
```

- 2 To create a new View to list all records from COLLEGE table, right-click on the code editor window of **HomeController.cs** within the **List() {...}** function. Select the **Add View** option. In the pop-up **Add View** window, set the following fields:
  - **Template:** List
  - **Model class:** College (SigmaUniversity.Models)
  - **Data context class:** DataContext (SigmaUniversity.Models)
 Click on the **Add** button.

The 'Add View' dialog box is shown with the following settings:

- View name: List
- Template: List
- Model class: College (SigmaUniversity.Models)
- Data context class: DataContext (SigmaUniversity.Models)
- Options:
  - ☐ Create as a partial view
  - ☐ Reference script libraries
  - ☒ Use a layout page: [Empty field]

Buttons: Add, Cancel

- 3 Add a new **Index()** method into HomeController.cs.

```
// GET: Home
public ActionResult Index()
{
    return View();
}
```

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 4 Add a new **Index(ITAdmin login)** method into HomeController.cs. This is the method that will execute the SQL query to validate the StaffID and Password entered. If the account (i.e. StaffID and Password) is valid, a Session will be created and the administrator will be redirected to the List page. If the account is not valid, an error message will be displayed on the Login page.

```
// POST: Home
[HttpPost]
public ActionResult Index(ITAdmin login)
{
    try
    {
        // Create a new list to store details of new login
        List<object> newLogin = new List<object>();
        newLogin.Add(login.StaffID);
        newLogin.Add(login.Password);

        // Copy login items into an array for easy addition into SQL statement
        object[] loginItems = newLogin.ToArray();
        var data = db.ITADMIN.SqlQuery("SELECT * FROM ITADMIN WHERE StaffID=@p0 AND Password=@p1", loginItems).SingleOrDefault();

        // If login is successful
        if (data != null)
        {
            Session["StaffID"] = login.StaffID.ToString(); // Set session
            return RedirectToAction("List"); // Redirect to List page
        }
        else
        {
            ViewBag.msg = "Staff ID and/or Password is invalid.";
            return View();
        }
    }
    catch
    {
        return View();
    }
}
```

- 5 In order to ensure that the all other pages can only be accessed after a successful login, you need to check the Session validity before displaying the page. To do this, a **Session validity check** must be included in the following methods in **HomeController.cs**.

## a. List()

```
// GET: List
public ActionResult List()
{
    // If session is valid
    if (Session["StaffID"] != null)
    {
        var data = db.COLLEGE.SqlQuery("SELECT * FROM COLLEGE").ToList();
        return View(data);
    }
    else
    {
        return RedirectToAction("Index");
    }
}
```

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

**b. Details(int id)**

```
// GET: Home/Details/5
public ActionResult Details(int id)
{
    // If session is valid
    if (Session["StaffID"] != null)
    {
        var data = db.COLLEGE.SqlQuery("SELECT * FROM COLLEGE WHERE CollegeCode=@p0", id).SingleOrDefault();
        return View(data);
    }
    else
    {
        return RedirectToAction("Index");
    }
}
```

**c. Create()**

```
// GET: Home/Create
public ActionResult Create()
{
    // If session is valid
    if (Session["StaffID"] != null)
    {
        return View();
    }
    else
    {
        return RedirectToAction("Index");
    }
}
```



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## d. Create(College collection)

```
// POST: Home/Create
[HttpPost]
public ActionResult Create(College collection)
{
    // If session is valid
    if (Session["StaffID"] != null)
    {
        try
        {
            // Create a new list to store details of new record to be inserted
            List<object> newRecord = new List<object>();
            newRecord.Add(collection.CollegeCode);
            newRecord.Add(collection.CollegeName);
            newRecord.Add(collection.DeanFirstName);
            newRecord.Add(collection.DeanLastName);
            newRecord.Add(collection.DeanEmail);

            // Copy record items into an array for easy addition to SQL statement
            object[] recordItems = newRecord.ToArray();
            int result = db.Database.ExecuteSqlCommand("INSERT INTO COLLEGE " +
                "(CollegeCode, CollegeName, DeanFirstName, DeanLastName, DeanEmail) " +
                "VALUES (@p0, @p1, @p2, @p3, @p4)", recordItems);

            if (result > 0)
            {
                ViewBag.msg = "College record is added.";
            }
            return View();
        }
        catch
        {
            return View();
        }
    }
    else
    {
        return RedirectToAction("Index");
    }
}
```

## e. Edit(int id)

```
// GET: Home/Edit/5
public ActionResult Edit(int id)
{
    // If session is valid
    if (Session["StaffID"] != null)
    {
        var data = db.COLLEGE.SqlQuery("SELECT * FROM COLLEGE WHERE CollegeCode=@p0", id).SingleOrDefault();
        return View(data);
    }
    else
    {
        return RedirectToAction("Index");
    }
}
```

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## f. Edit (int id, College collection)

```
// POST: Home/Edit/5
[HttpPost]
public ActionResult Edit(int id, College collection)
{
    // If session is valid
    if (Session["StaffID"] != null)
    {
        try
        {
            // Create a new list to store details of the record to be updated
            List<object> record = new List<object>();
            record.Add(collection.CollegeCode);
            record.Add(collection.CollegeName);
            record.Add(collection.DeanFirstName);
            record.Add(collection.DeanLastName);
            record.Add(collection.DeanEmail);

            // Copy record items into an array for easy addition to SQL statement
            object[] recordItems = record.ToArray();
            int result = db.Database.ExecuteSqlCommand("UPDATE COLLEGE " +
                "SET CollegeCode=@p0, CollegeName=@p1, DeanFirstName=@p2, DeanLastName=@p3, DeanEmail=@p4 " +
                "WHERE CollegeCode=" + id, recordItems);

            if (result > 0)
            {
                ViewBag.msg = "College record is updated.";
            }
            return View();
        }
        catch
        {
            return View();
        }
    }
    else
    {
        return RedirectToAction("Index");
    }
}
```

## g. Delete(int id)

```
// GET: Home/Delete/5
public ActionResult Delete(int id)
{
    // If session is valid
    if (Session["StaffID"] != null)
    {
        var data = db.COLLEGE.SqlQuery("SELECT * FROM COLLEGE WHERE CollegeCode=@p0", id).SingleOrDefault();
        return View(data);
    }
    else
    {
        return RedirectToAction("Index");
    }
}
```

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

## h. Delete(int id, College collection)

```
// POST: Home/Delete/5
[HttpPost]
public ActionResult Delete(int id, College collection)
{
    // If session is valid
    if (Session["StaffID"] != null)
    {
        try
        {
            // SQL statement to delete a record
            int result = db.Database.ExecuteSqlCommand("DELETE FROM COLLEGE WHERE CollegeCode=@p0", id);

            // If record is deleted successfully, return to Index page to show updated COLLEGE table
            if (result > 0)
            {
                return RedirectToAction("List");
            }
            return View();
        }
        catch
        {
            return View();
        }
    }
    else
    {
        return RedirectToAction("Index");
    }
}
```

Note that when a record is deleted successfully, the page should be redirected to the list page: **return RedirectToAction("List");**

- 6 Click on the **Save All** button in the main menu.

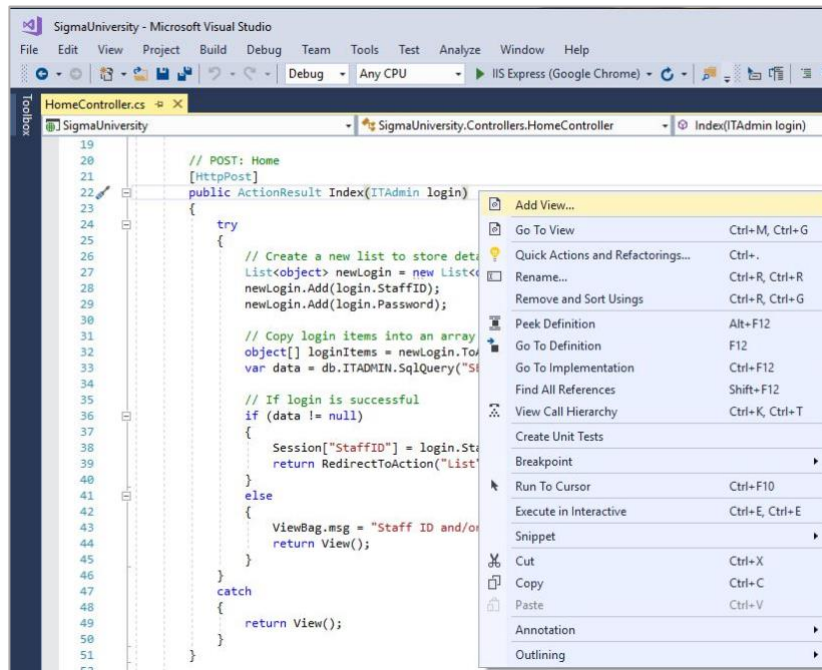
# LABSHEET 9

Module: Web Solutions Development

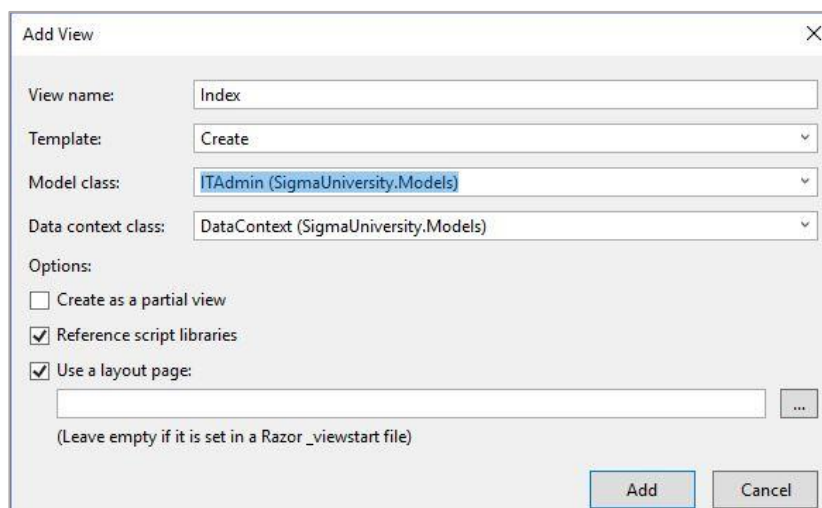
Module Code: IT5035FP

## Part (E4): Create the View for Login Page

- 1 To create a View for the Login page, right-click on the code editor window of **HomeController.cs** within the **Index(ITAdmin login) {...}** function. Select the **Add View** option.



- 2 In the pop-up **Add View** window, set the following fields:
  - **Template:** Create
  - **Model class:** ITAdmin (SigmaUniversity.Models)
  - **Data context class:** DataContext (SigmaUniversity.Models)
 Click on the **Add** button



# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

- 3 When you are prompted that an Index file already exists, click on the **Yes** button to replace it.



- 4 A new **Index.cshtml** file will be generated. When you run the Index.cshtml file, you will observe that there is only 1 field to enter the Password. The StaffID field is not available. To add this field, add the following code into **Index.cshtml** after the line **@Html.ValidationSummary**, and before the **Password** code block.

```
@Html.ValidationSummary(true, "", new { @class = "text-danger" })
<div class="form-group">
    @Html.LabelFor(model => model.StaffID, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.StaffID, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.StaffID, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Password, htmlAttributes: new { @class = "control-label col-md-2" })
```

- 5 To display the login error message, add the following code after the **Password** code block and before the **Login button** code block.

```
@Html.ValidationMessageFor(model => model.Password, "", new { @class = "text-danger" })
</div>
</div>

<div class="form-group">
    <div class="col-md-10">
        @if (ViewBag.msg != null)
        {
            <h5>@ViewBag.msg</h5>
        }
    </div>
</div>

<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" value="Login" class="btn btn-default" />
```

# LABSHEET 9

Module: Web Solutions Development

Module Code: IT5035FP

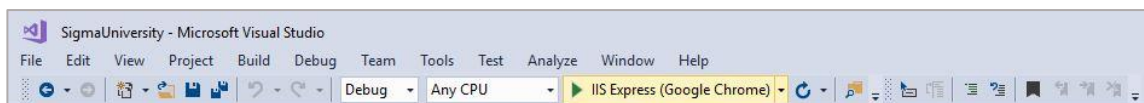
- 6 As the original home page (i.e. Index page) has been renamed to List page, you need to update the “Back to List” link in the following pages to point to List instead of Index.

- Create.cshtml
- Delete.cshtml
- Details.cshtml
- Edit.cshtml

The following code shows the updated link that points to List.

```
<div>  
    @Html.ActionLink("Back to List", "List")  
</div>
```

- 7 Click on the **Save All** button in the main menu.
- 8 To build and test the project, click on the **IIS Express (Google Chrome)** button.



- END -