UTM

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF ENGINEERING

SCHOOL OF COMPUTING

SEMESTER 2/20202021

SCSP3106 - APPLICATION DEVELOPMENT

SECTION: 01

ASSIGNMENT 2 - MVC CORE

LECTURER: PM DR HAZA NUZLY BIN ABDULL HAMED

NAME: MUHAMMAD SYAHMI BI SUFAKHI

MATRIC NO: A18CS0161

1. Briefly Explain
    a. MVC

       MVC is a short term for Model , Views and Controller

        - Model - Managing data for the application. It acts as entity representation of the database tables entity such as table columns
        - View - User interface, where the user can view and interact with the application. Contains Markup (Razor Syntax) and write in C#
        - Controller - handles user interaction and routing any action based interaction.

    b. .NET CORE

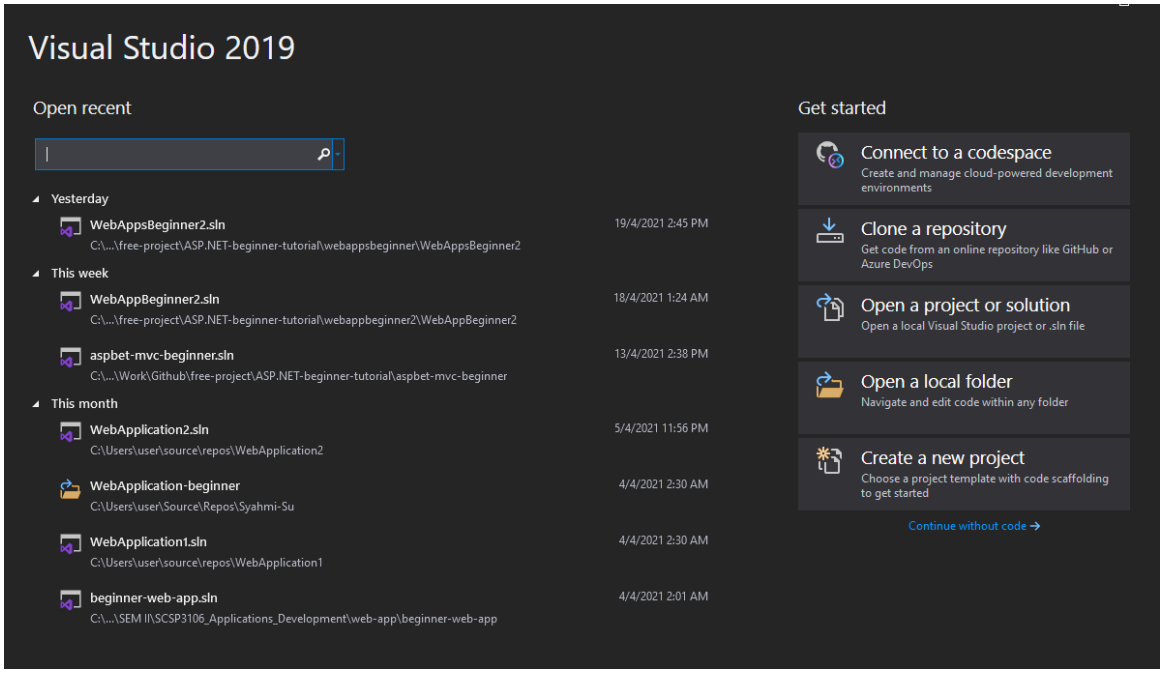
       .NET CORE is a new framework that can build Desktop,Web , Mobile and Cloud applications at the same place. It is a Cross-Platform and Open-Source framework developed by Microsoft.

       Components of .NET

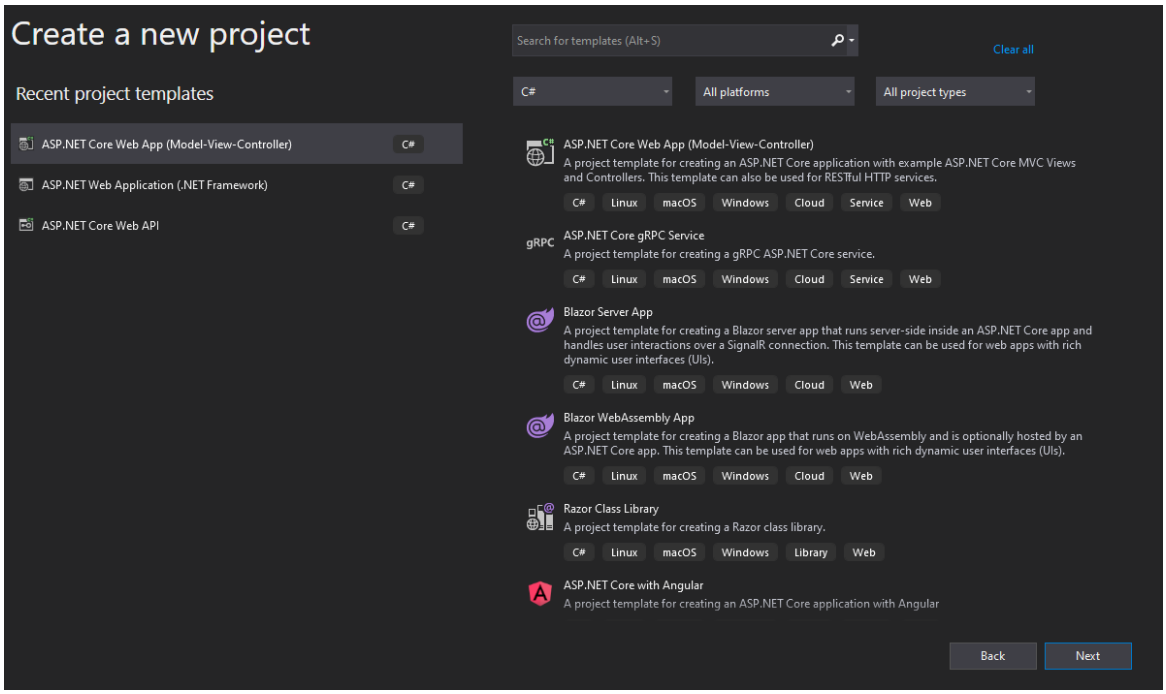
        - Entity Framework Core
        - Identity Core
        - MVC Core

2. Technical Step in VS2019:

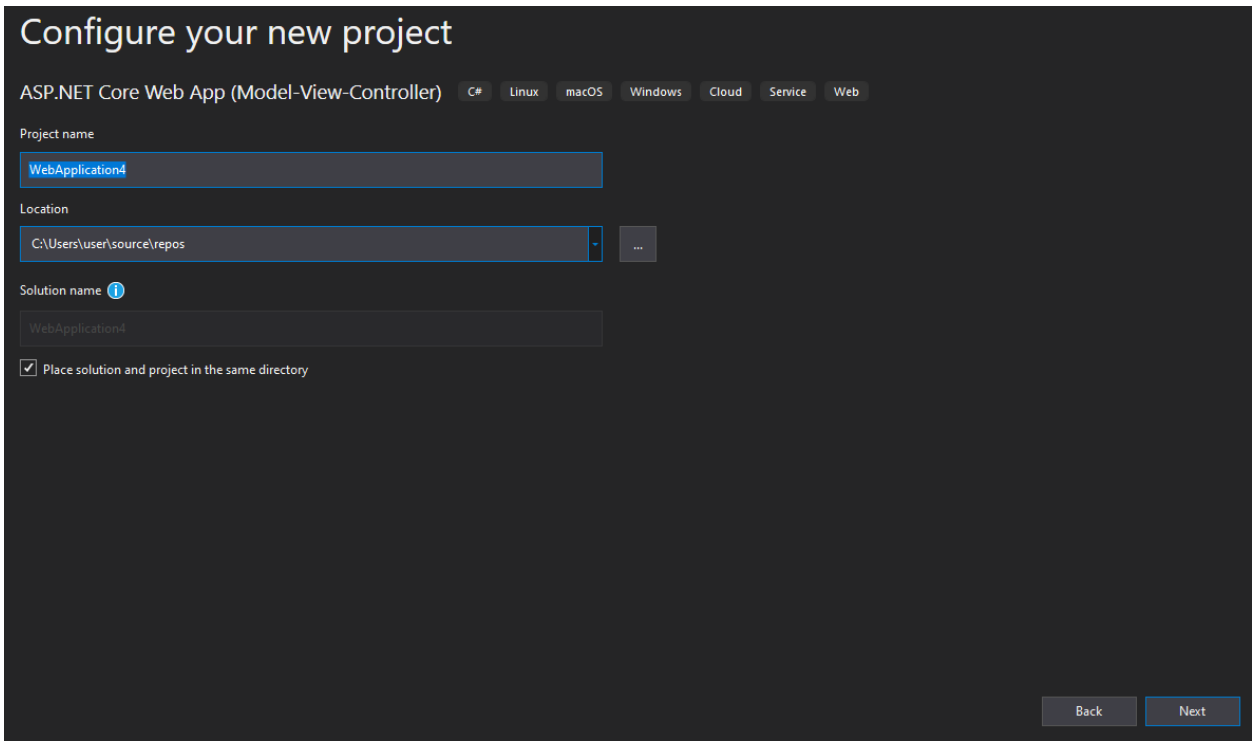
- Create MVC project
  Click Create New Project

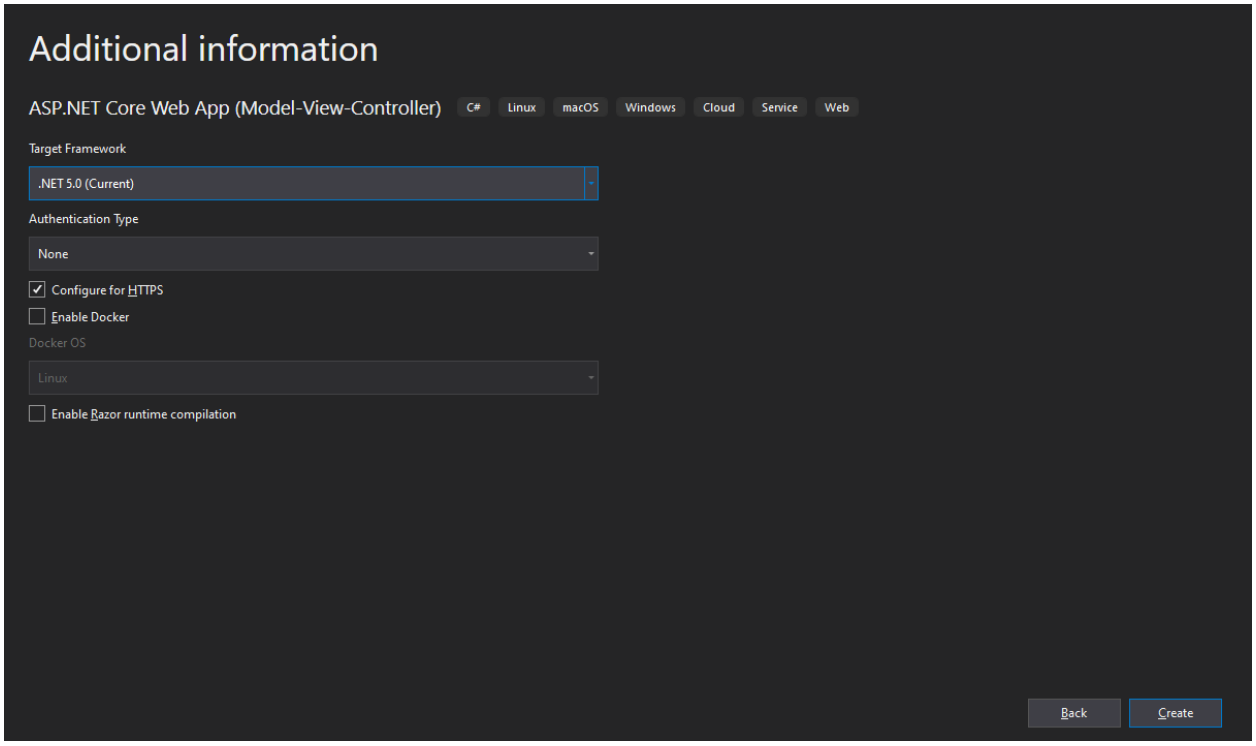


Choose ASP.NET Core Web App (Model-View-Controller) for C#

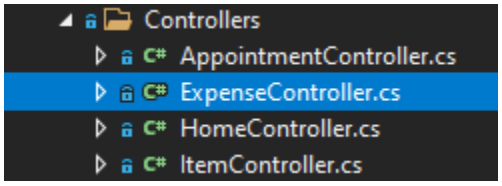


Choose .NET 5 as Target framework and Rename your project.





- MVC Structure in VS19

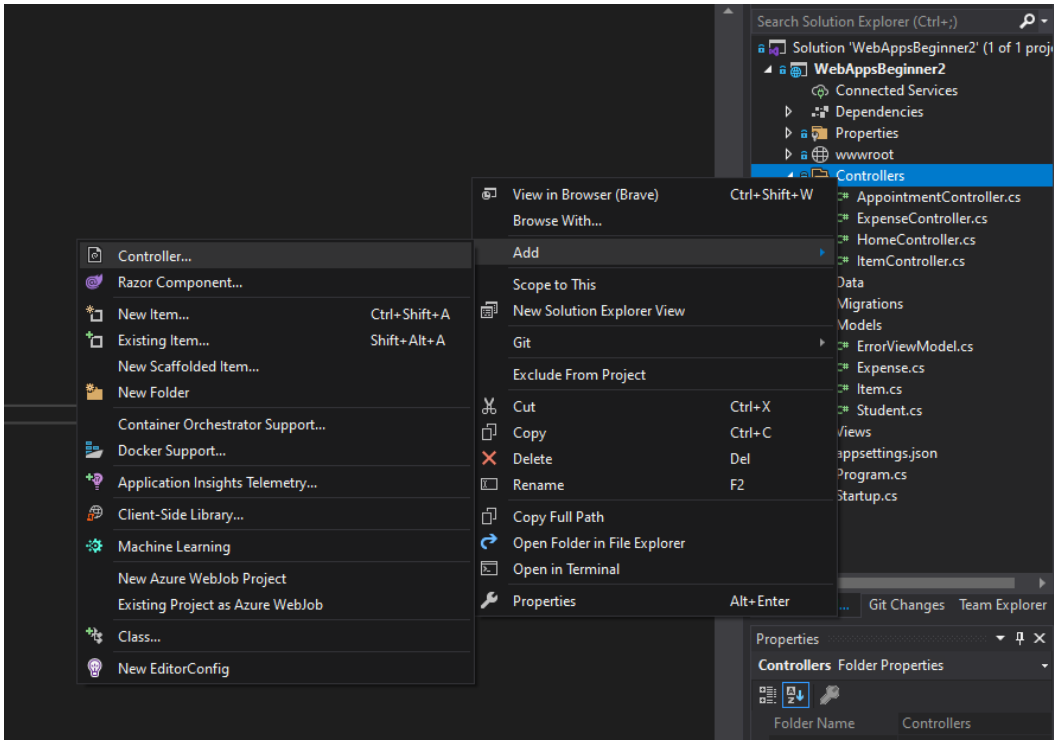


In SolutionExplorer in Project, you can see the Controller file containing all controller files according to which module that is relatable.

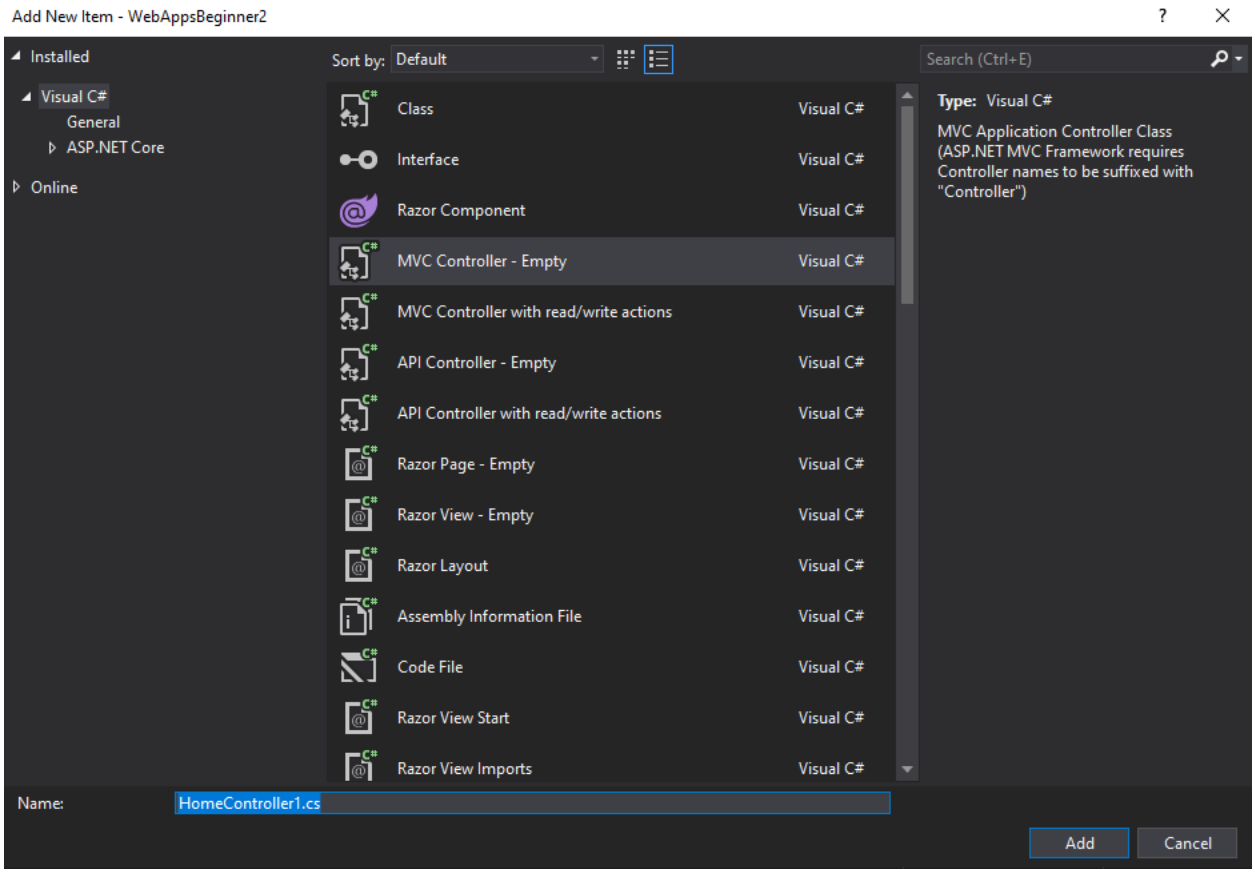
```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using WebAppsBeginner2.Data;
using WebAppsBeginner2.Models;

namespace WebAppsBeginner2.Controllers
{
    1 reference
    public class ExpenseController : Controller
    {
        private readonly ApplicationDBContext _db;

        0 references
        public ExpenseController(ApplicationDBContext db)
        {
            _db = db;
        }
        0 references
        public IActionResult Index()
        {
            IEnumerable<Expense> objList = _db.Expenses;
            return View(objList);
        }

        //GET-Create
        0 references
        public IActionResult Create()
        {
            return View();
        }
```

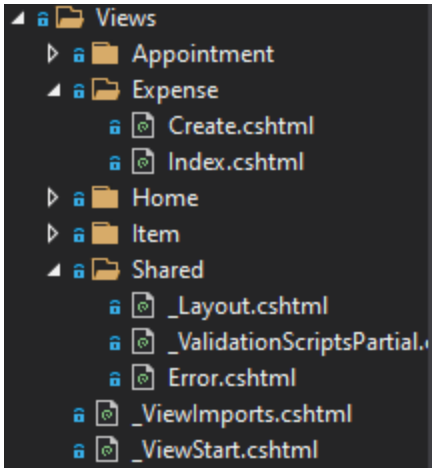
As Example in ExpenseController.cs, in the file, we create a function/class inside the public class to control interaction into the respective page we create using the public IActionResult() function.

To create a new Controller page, right click Controller folder, and choose add Controller.



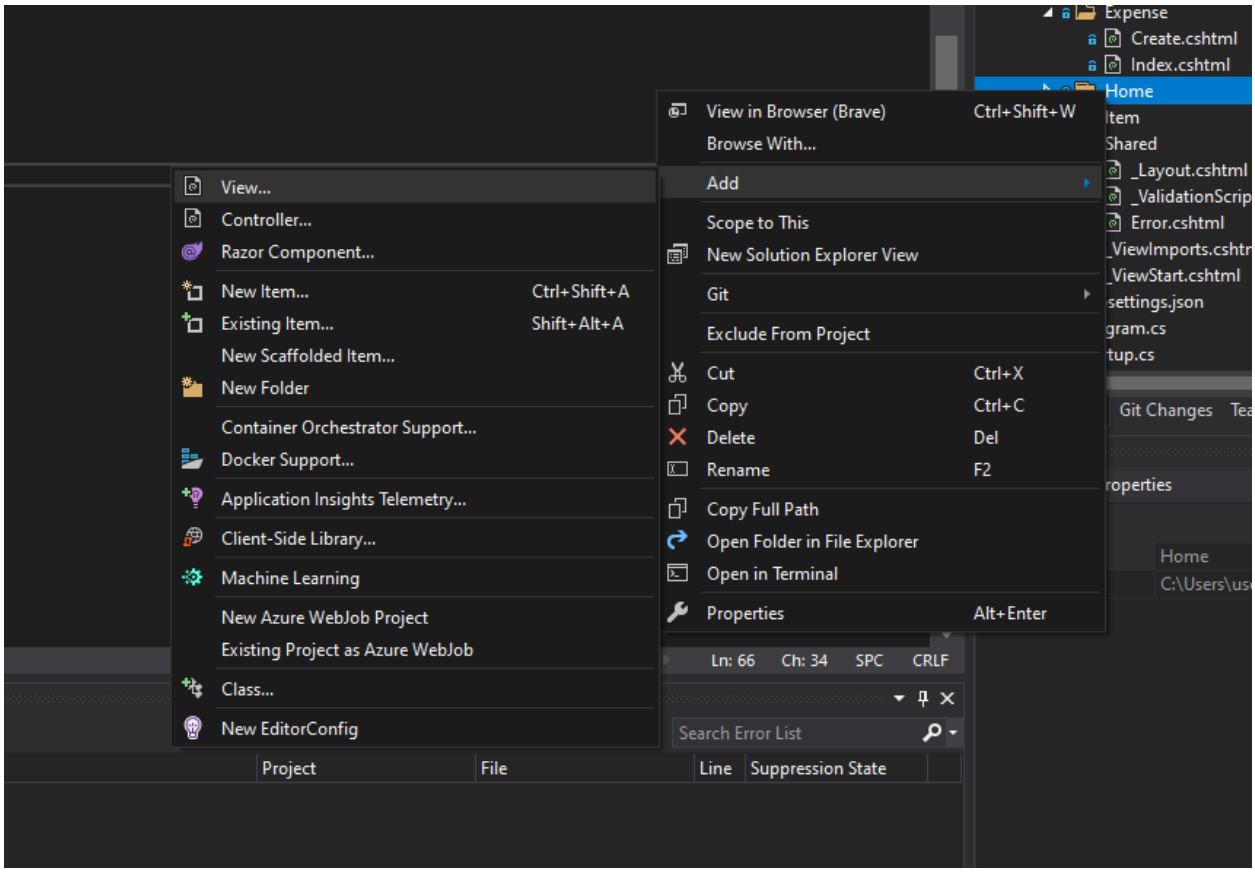
Next, Choose the MVC Controller - Empty option and rename the file appropriately.



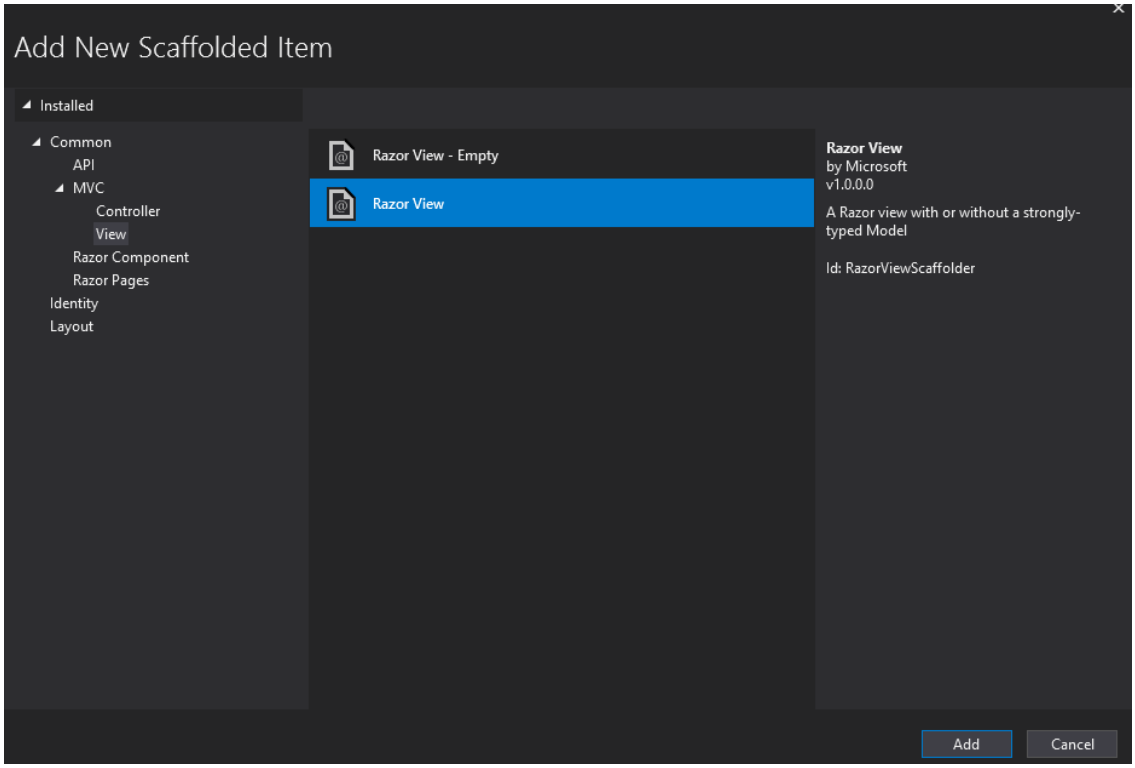


For VIew Folder, this is where the user interface page will be placed according to their route. As an example, in the Expense folder there was a Create page. To access it on the browser, we simply enter localhost:8823/Expense/Create when doing it on the local server.

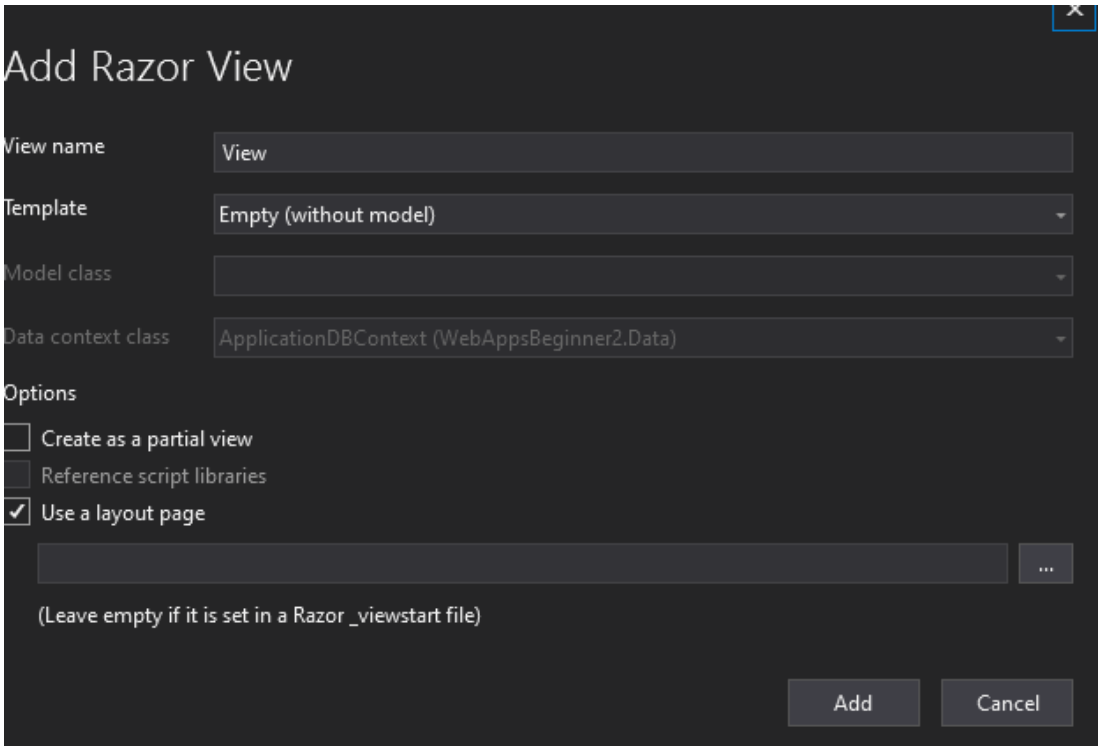
To create a new View page, simply right click on Views folder or any specific subfolder in views.



Choose Razor View folder

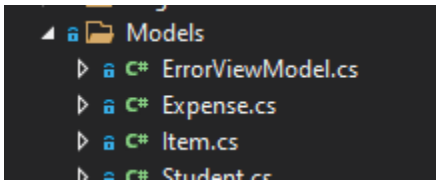


Rename the folder and choose the corresponding setting before creating.

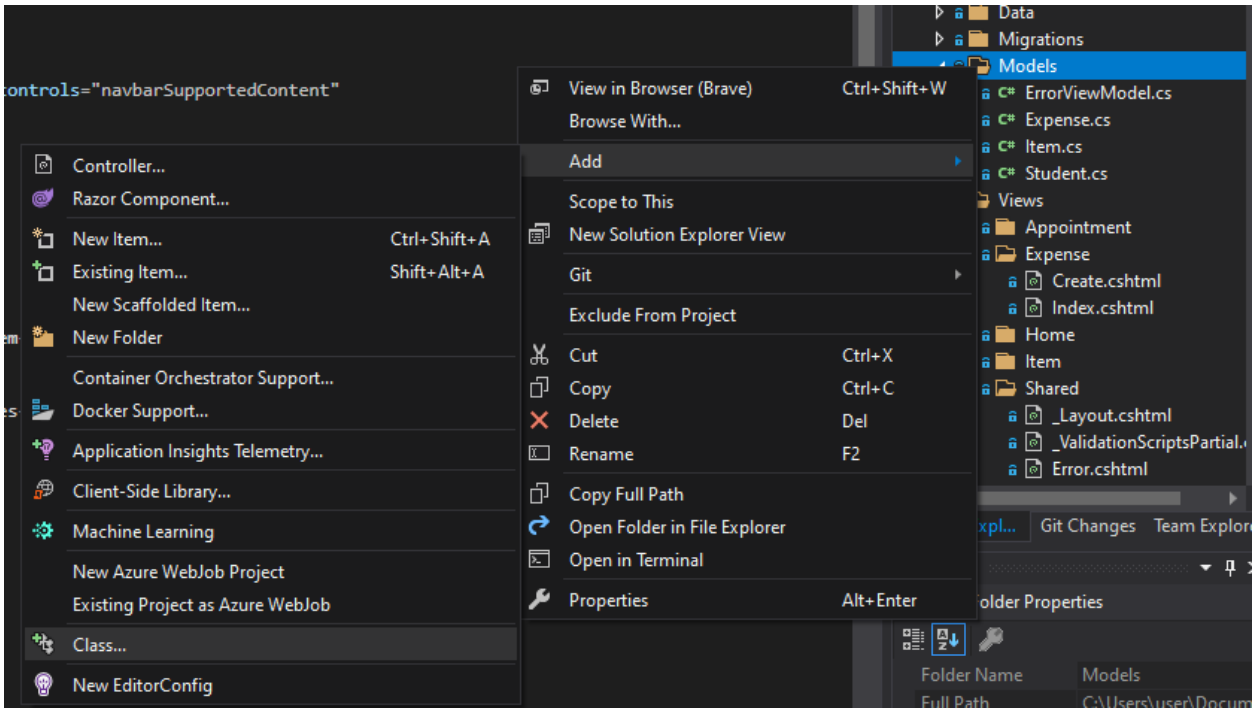


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - WebAppsBeginner2</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">WebAppsBeginner2</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-controller="Item" asp-action="Index">Borrow Item</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-controller="Expense" asp-action="Index">Expenses</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
```

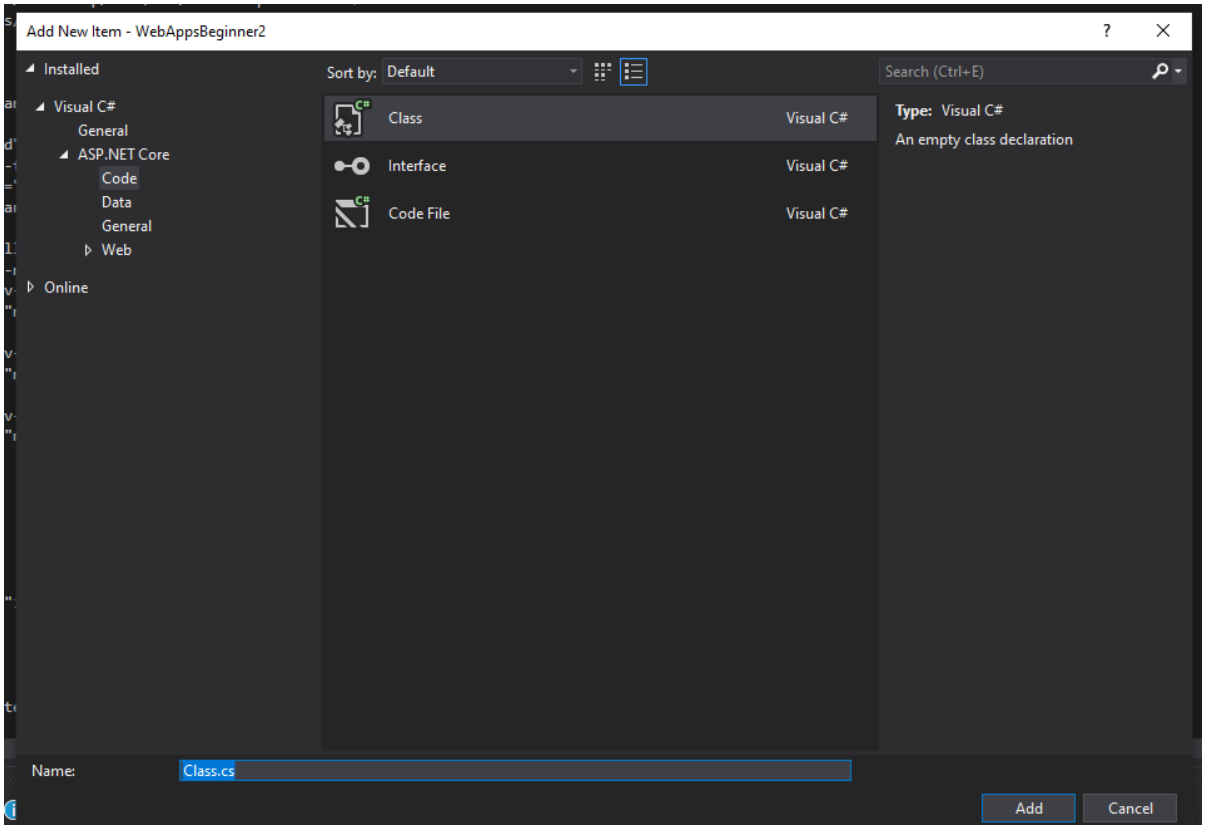
Page created.



For model folder. Model pages are created according to each table that will be used in the application. To create a new Models page, right click on the models file and choose add Class



Choose Class item and rename the item.



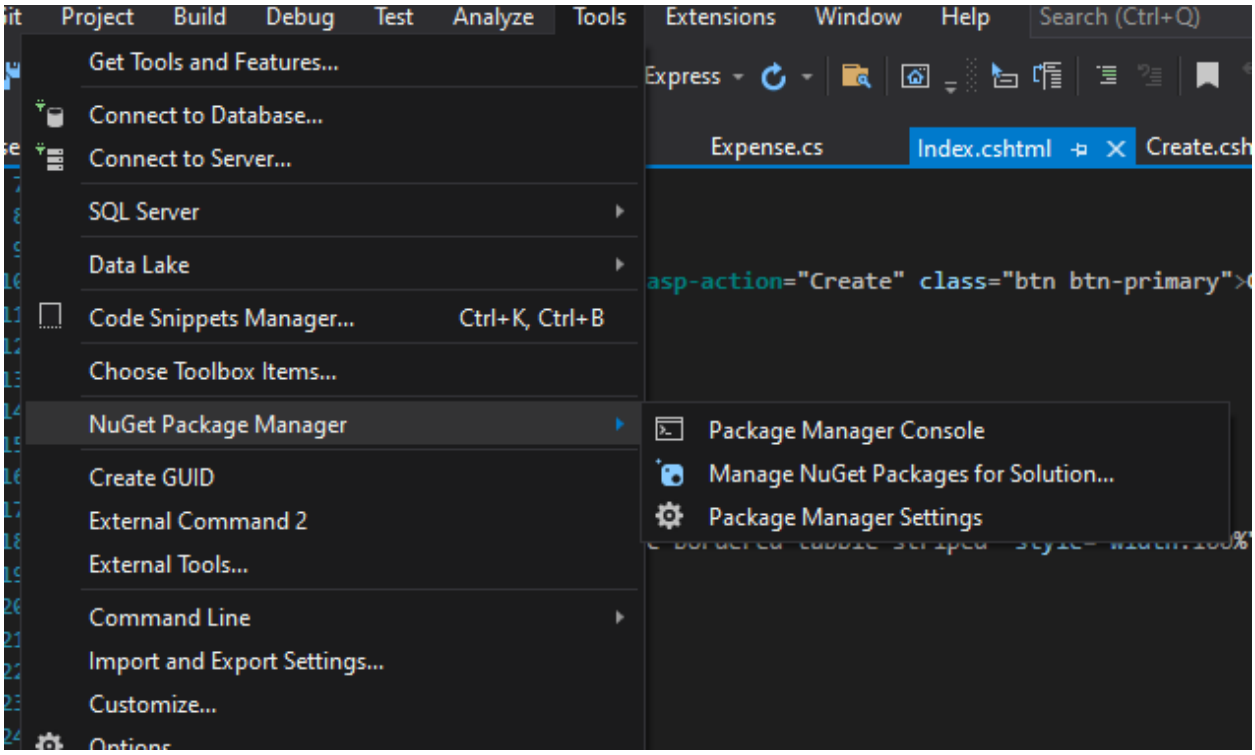
In the Model page. We create and code in all attributes that we want to use and to migrate into the database.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Threading.Tasks;
5   using System.ComponentModel;
6   using System.ComponentModel.DataAnnotations;
7
8   namespace WebAppsBeginner2.Models
9   {
        9 references
10      public class Expense
11      {
12          [Key]
            2 references
13          public int Id { get; set; }
14          [DisplayName("Expense")]
15          [Required]
            4 references
16          public string ExpenseName { get; set; }
17          [Required]
18          [Range(1, int.MaxValue, ErrorMessage ="Amount must be greater than 0!")]
            4 references
19          public int Amount { get; set; }
20      }
21  }
22
```

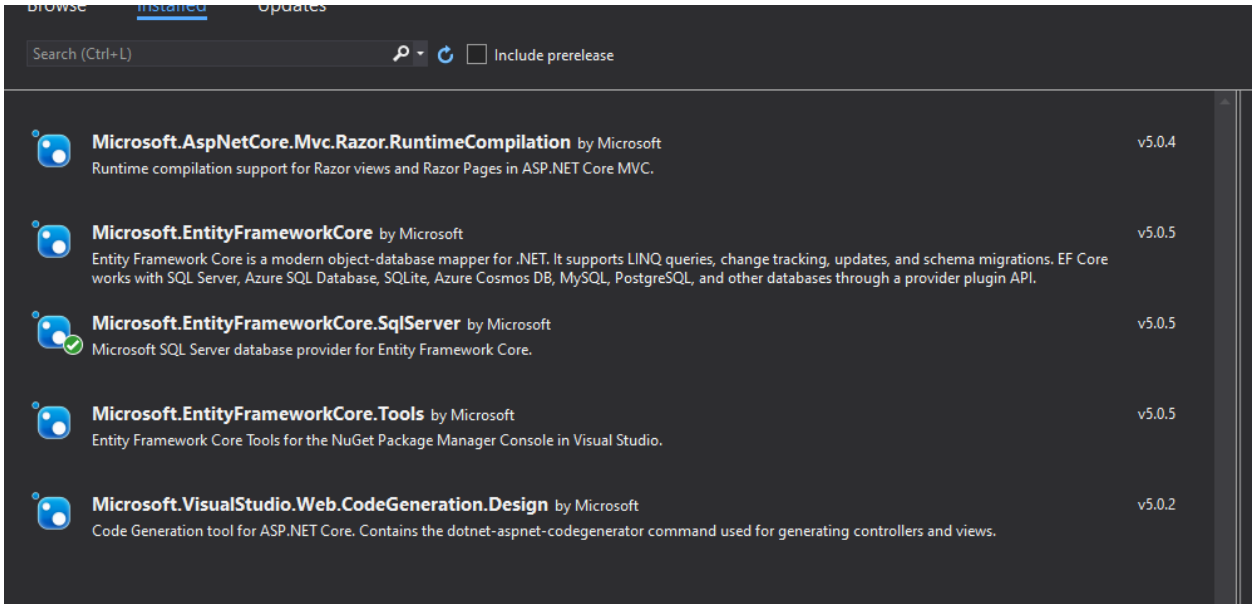
Expense.cs Model Page example. In the page, we can see that we set Id as a primary key and set ExpenseName and Amount as notNull fields.

- CRUD Operation

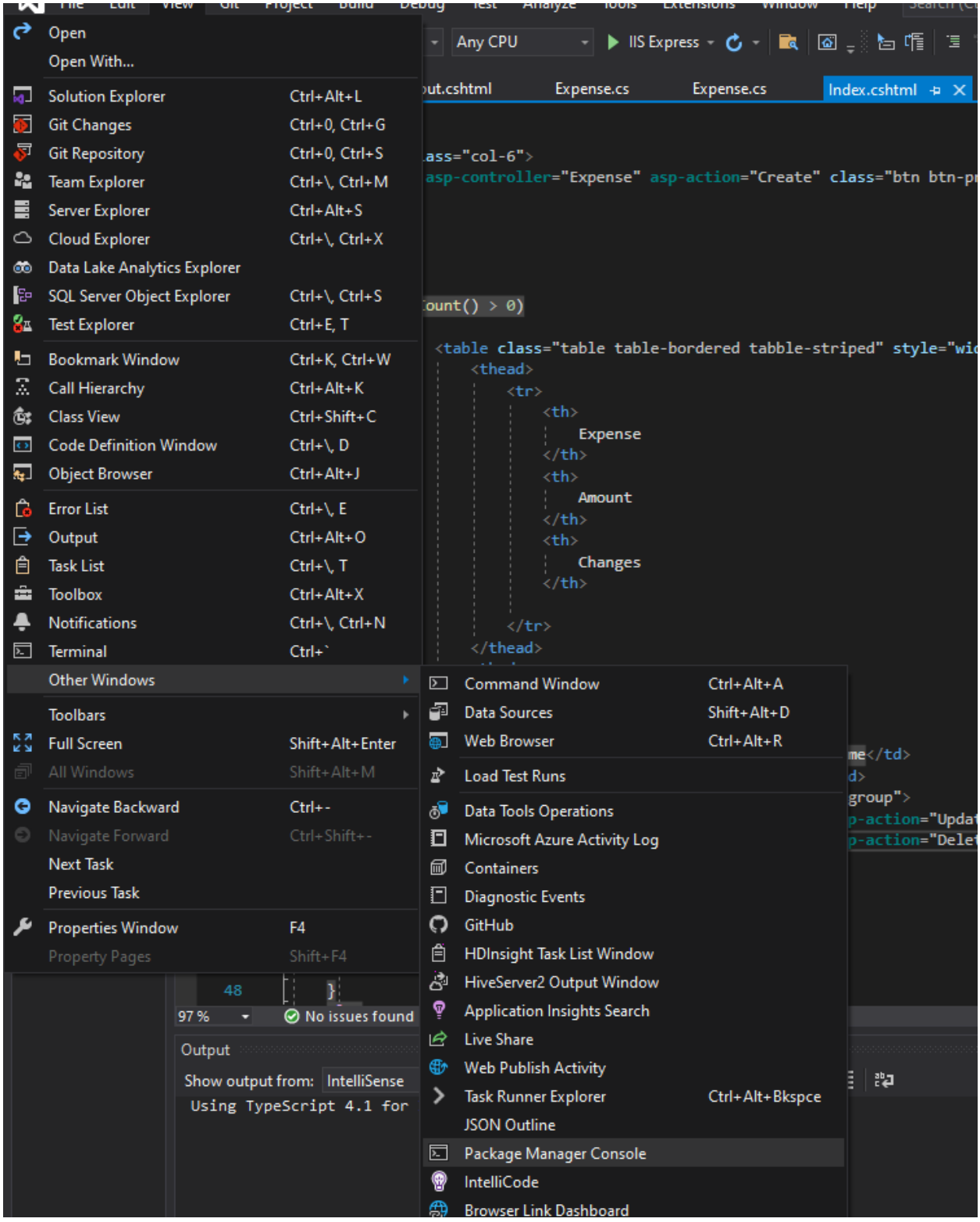
  Before starting CRUD operation, we need to set up the database in the SQL Server by migrating the database created in the project. To migrate the database scheme. Install migrating package framework at NuGet Package manager.



Install all the package;



To run the migration, click on view > Other window > Package Manager Console

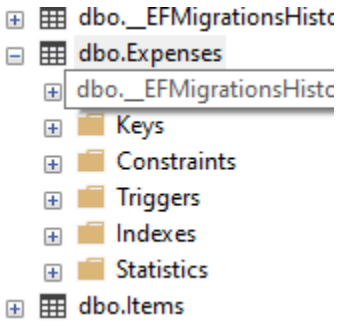


Type add-migration <name> on the console and update-database to update database into the SQL Server

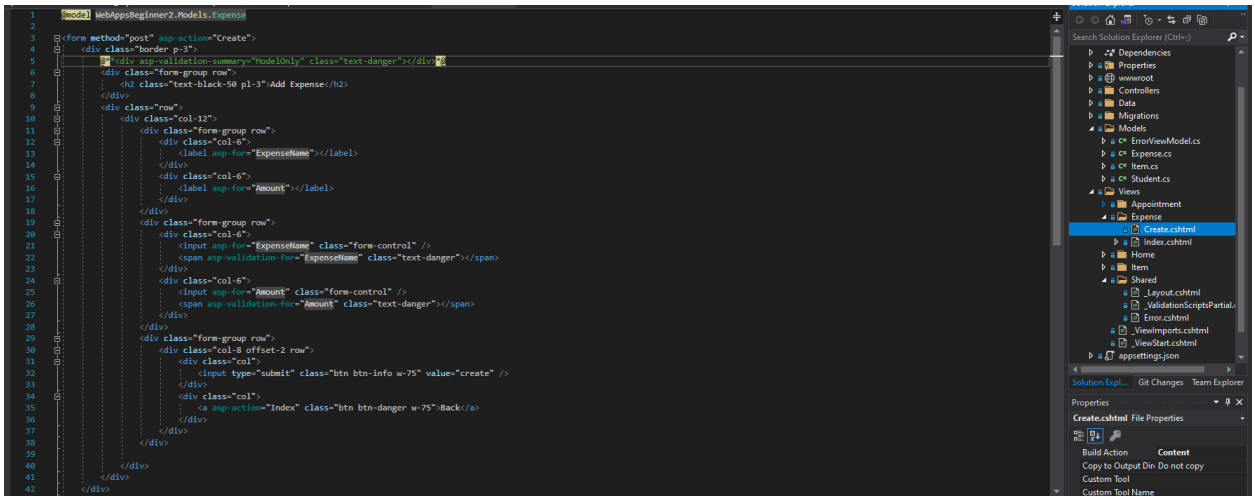
```
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages. Some packages may include dependencies which are governed by additional licenses. Follow the package source (feed) URL to determine any dependencies.

Package Manager Console Host Version 5.9.1.0

Type 'get-help NuGet' to see all available NuGet commands.

PM> add-migration DBSQL
Build started...
```

```
PM> update-database
Build started...
```

Database Created



Create operation, we need to create an interface first, for that, we create a new subfolder inside a View folder.



For Delete Operation, In index.cshtml page, we create a new button in the list to delete the list in the database

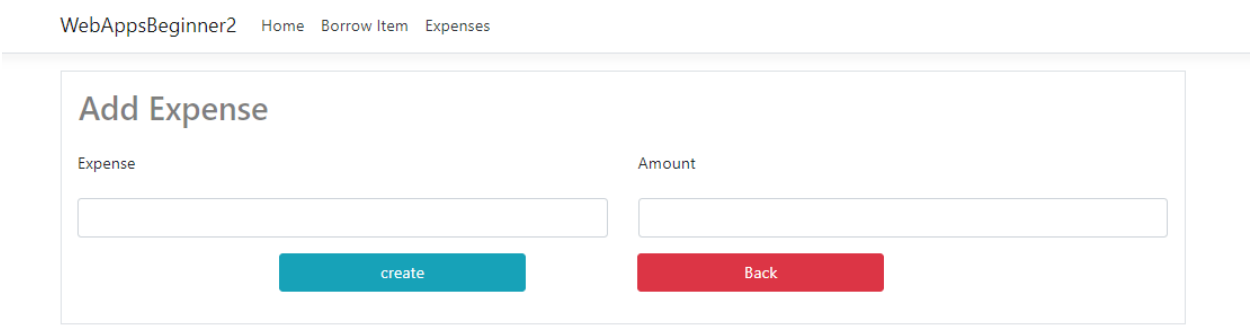
```
<div class="w-100 btn-group" role="group">
    <a asp-controller="Expense" asp-action="Update" asp-route-Id="@expense.Id" class="btn btn-primary" mx-1>Update</a>
    <a asp-controller="Expense" asp-action="Delete" asp-route-Id="@expense.Id" class="btn btn-danger" mx-1>Delete</a>
</div>
```

We also add a Delete class in Expense Controller.

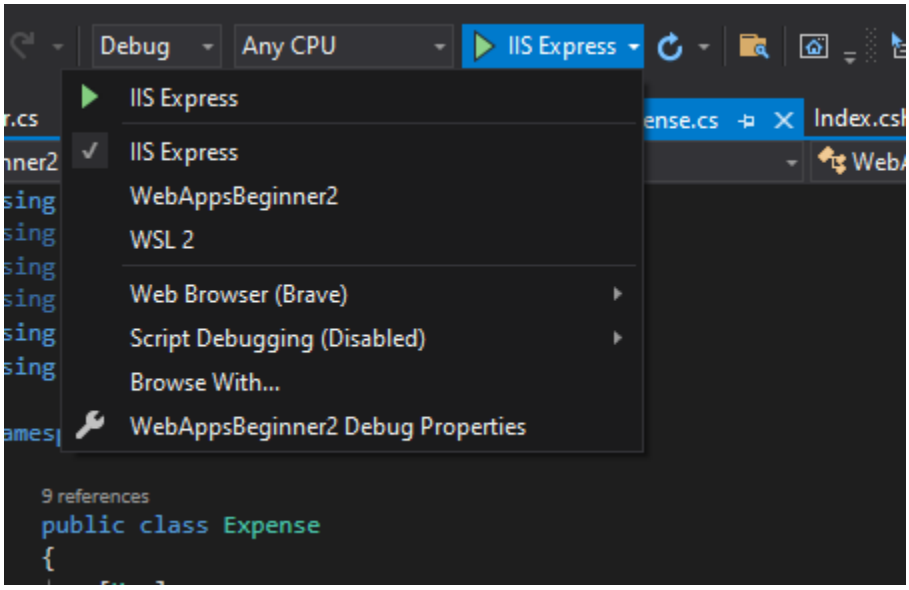
```
//GET-Delete
0 references
public IActionResult Delete(int? id)
{
    if(id == null || id == 0)
    {
        return NotFound();
    }
    var obj = _db.Expenses.Find(id);
    if(obj == null)
    {
        return NotFound();
    }
    return View(obj);


}
```

```
//POST-Delete
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public IActionResult DeletePost(int? id)
{
    var obj = _db.Expenses.Find(id);
    if(obj == null)
    {
        return NotFound();
    }

        _db.Expenses.Remove(obj);
        _db.SaveChanges();
        return RedirectToAction("Index");


}
```

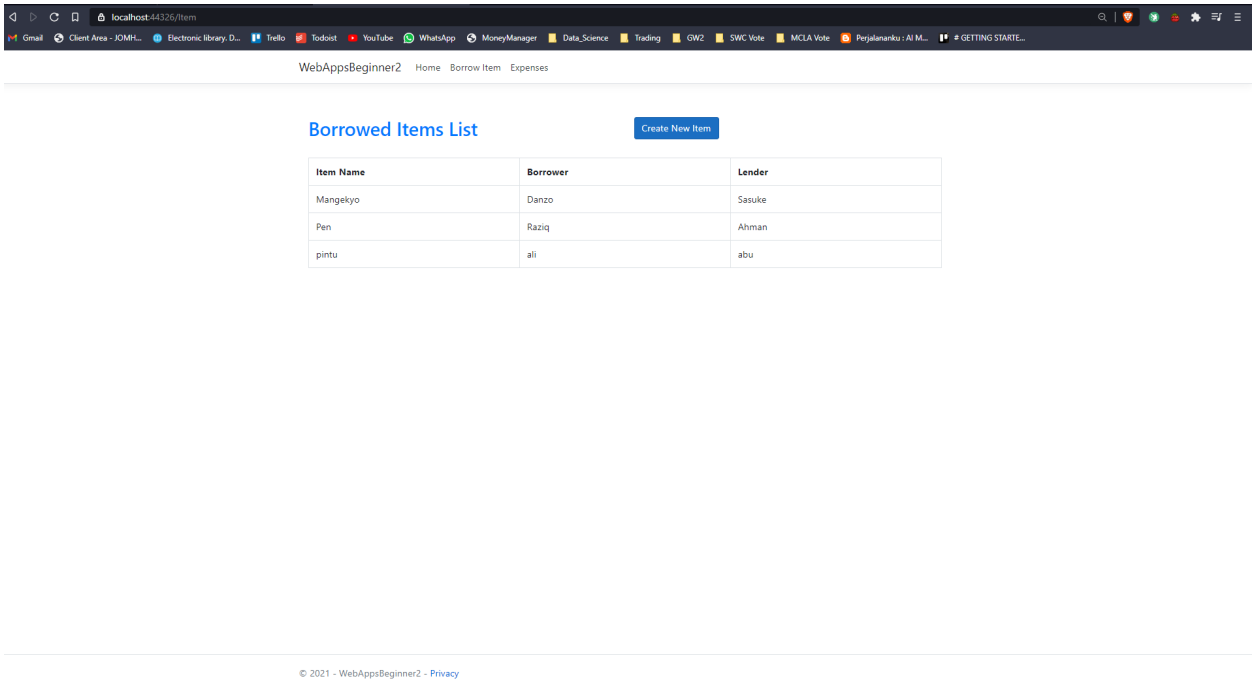
In this page, we create a button to bring us to the Create page to insert new expenses into the database.



- How To Run the project



Click on the IIS Express to run the project.



Your project is running successfully.