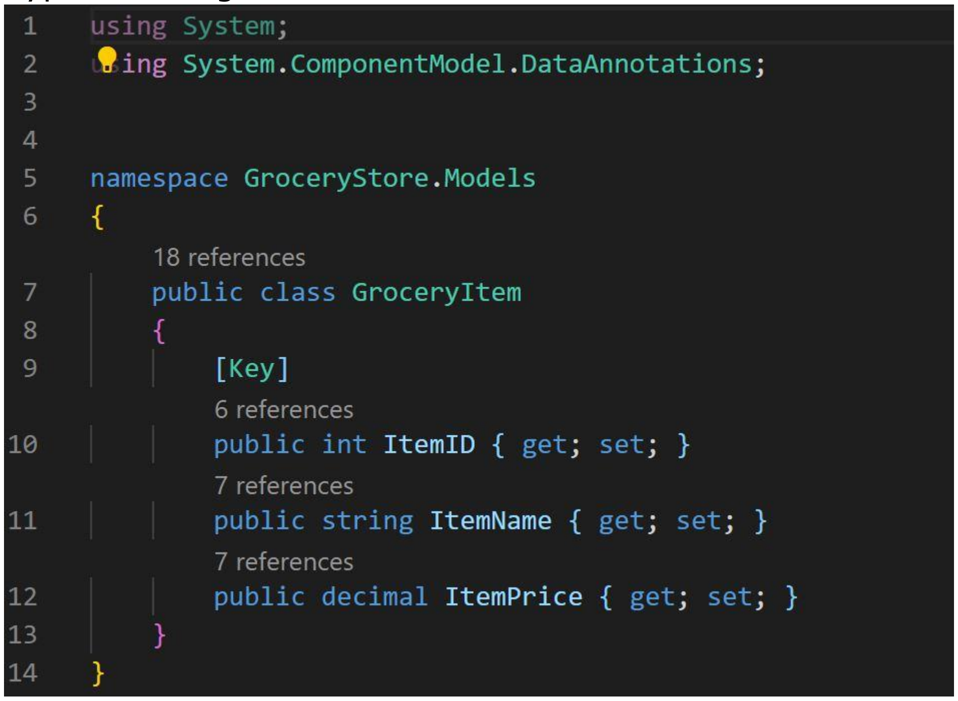
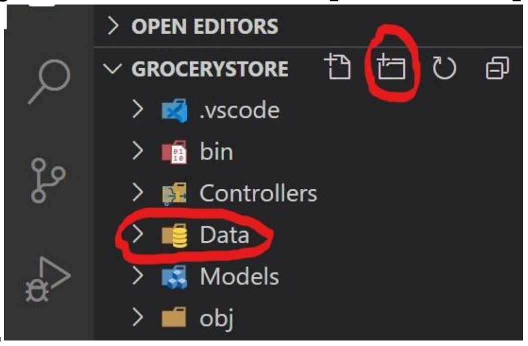
1. Opening MVC application folder
   a. Open Command prompt
   b. Go inside Grocery directory
   c. Type [code .] and press enter. Example
      C:\Desktop\GroceryStore> code .
   d. This should open VS Code with your GroceryStore project files.

2. Creating GroceryItem Model
   a. Right click on [Models Folder]
   b. Click [New File]
   c. Type [GroceryItem.cs] and press enter. This will create empty C# file.
   d. Type following code.



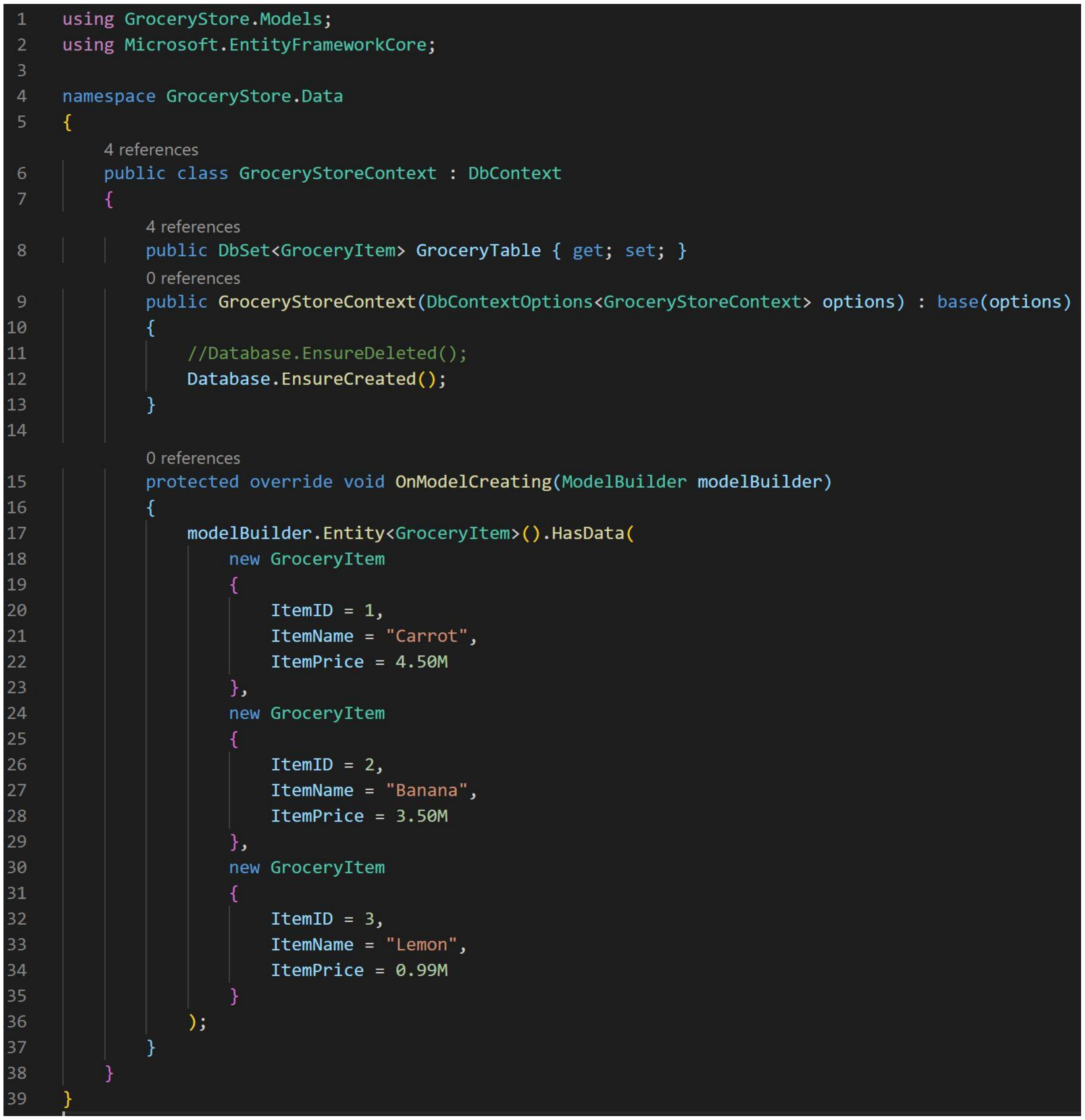
```
using System;
using System.ComponentModel.DataAnnotations;


namespace GroceryStore.Models
{
    18 references
    public class GroceryItem
    {
        [Key]
        6 references
        public int ItemID { get; set; }
        7 references
        public string ItemName { get; set; }
        7 references
        public decimal ItemPrice { get; set; }
    }
}
```

3. Create a class that derives from DbContext
   a. Hover over GroceryStore project folder and click [new folder]



Type [Data] and press enter.

b. Right click on the [Data Folder] and click [New File].
c. Type GroceryStoreContext.cs and press enter.
d. Type following code.

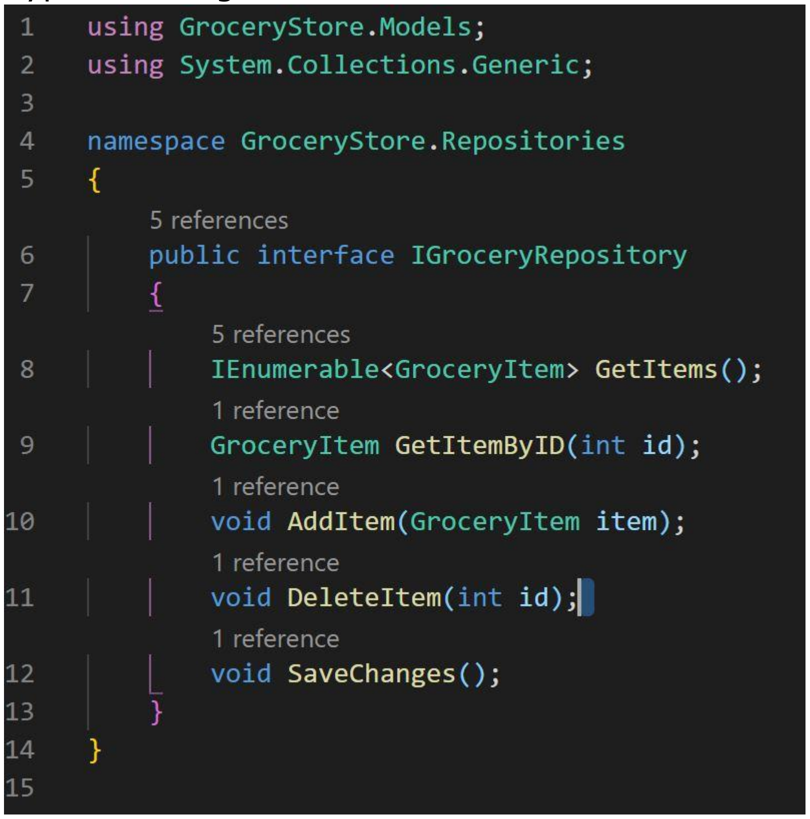


```csharp
using GroceryStore.Models;
using Microsoft.EntityFrameworkCore;

namespace GroceryStore.Data
{
    4 references
    public class GroceryStoreContext : DbContext
    {
        4 references
        public DbSet<GroceryItem> GroceryTable { get; set; }
        0 references
        public GroceryStoreContext(DbContextOptions<GroceryStoreContext> options) : base(options)
        {
            //Database.EnsureDeleted();
            Database.EnsureCreated();
        }

        0 references
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<GroceryItem>().HasData(
                new GroceryItem
                {
                    ItemID = 1,
                    ItemName = "Carrot",
                    ItemPrice = 4.50M
                },
                new GroceryItem
                {
                    ItemID = 2,
                    ItemName = "Banana",
                    ItemPrice = 3.50M
                },
                new GroceryItem
                {
                    ItemID = 3,
                    ItemName = "Lemon",
                    ItemPrice = 0.99M
                }
            );
        }
    }
}
```

4. Set up Entity Framework Core to use SQLite
   a. Lets install Microsoft.EntityFrameworkCore.Sqlite v2.1.0
   b. Click [View menu] on the top. Click Terminal.

c. Inside Terminal Type [dotnet add package Microsoft.EntityFrameworkCore.Sqlite -v 2.1.0] and press enter

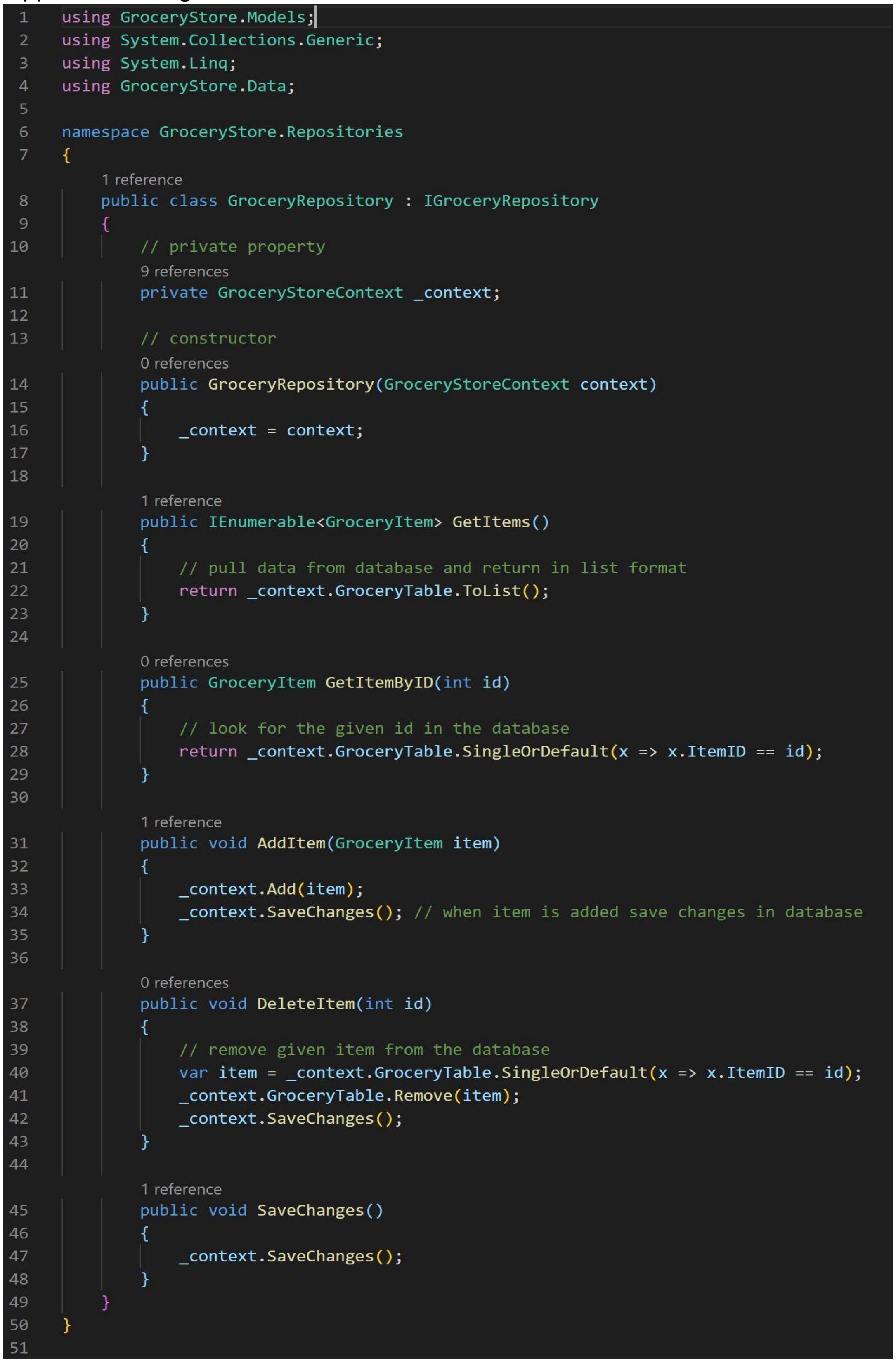
5. Use Entity Framework Core to Retrieve and Store Data
    a. Hover over GroceryStore Folder and click [New Folder]
    b. Type [Repositories]
    c. Right click on the [Repositories Folder] and click [New File]
    d. Type IGroceryRepository.cs. This will create interface file.
    e. Type following code.



```
using GroceryStore.Models;
using System.Collections.Generic;

namespace GroceryStore.Repositories
{
    5 references
    public interface IGroceryRepository
    {
        5 references
        IEnumerable<GroceryItem> GetItems();
        1 reference
        GroceryItem GetItemByID(int id);
        1 reference
        void AddItem(GroceryItem item);
        1 reference
        void DeleteItem(int id);
        1 reference
        void SaveChanges();
    }
}

```

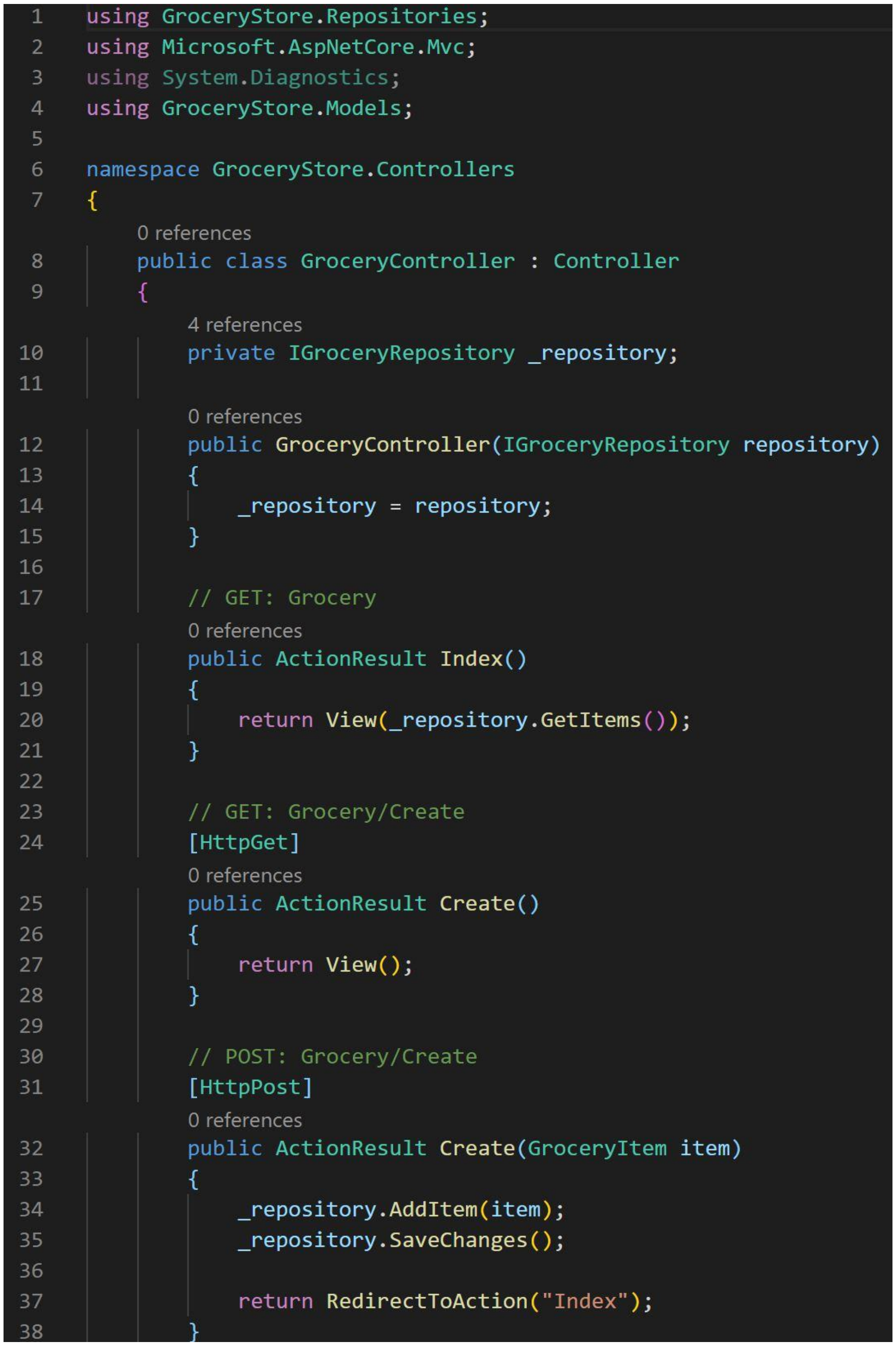
    f. Right click on [Repositories Folder]. Click [New File]
    g. Type GroceryRepository.cs

h. Type following code inside the file.



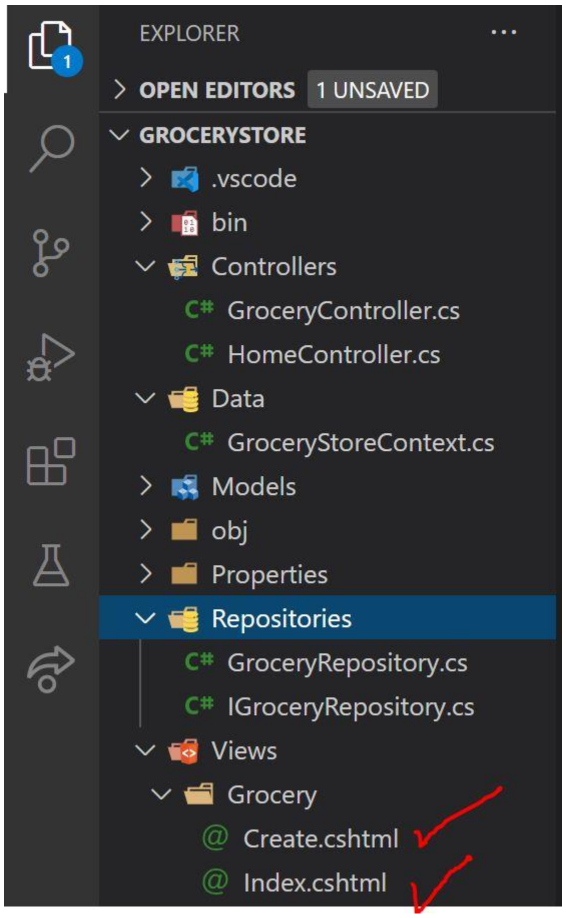
```
using GroceryStore.Models;
using System.Collections.Generic;
using System.Linq;
using GroceryStore.Data;

namespace GroceryStore.Repositories
{
    1 reference
    public class GroceryRepository : IGroceryRepository
    {
        // private property
        9 references
        private GroceryStoreContext _context;

        // constructor
        0 references
        public GroceryRepository(GroceryStoreContext context)
        {
            _context = context;
        }

        1 reference
        public IEnumerable<GroceryItem> GetItems()
        {
            // pull data from database and return in list format
            return _context.GroceryTable.ToList();
        }

        0 references
        public GroceryItem GetItemByID(int id)
        {
            // look for the given id in the database
            return _context.GroceryTable.SingleOrDefault(x => x.ItemID == id);
        }

        1 reference
        public void AddItem(GroceryItem item)
        {
            _context.Add(item);
            _context.SaveChanges(); // when item is added save changes in database
        }

        0 references
        public void DeleteItem(int id)
        {
            // remove given item from the database
            var item = _context.GroceryTable.SingleOrDefault(x => x.ItemID == id);
            _context.GroceryTable.Remove(item);
            _context.SaveChanges();
        }

        1 reference
        public void SaveChanges()
        {
            _context.SaveChanges();
        }
    }
}

```

6. Lets add new Controller to use a repository
    a. Right click on [Controllers Folder]. Click [New File]
    b. Type GroceryController.cs
    c. Type following code inside.

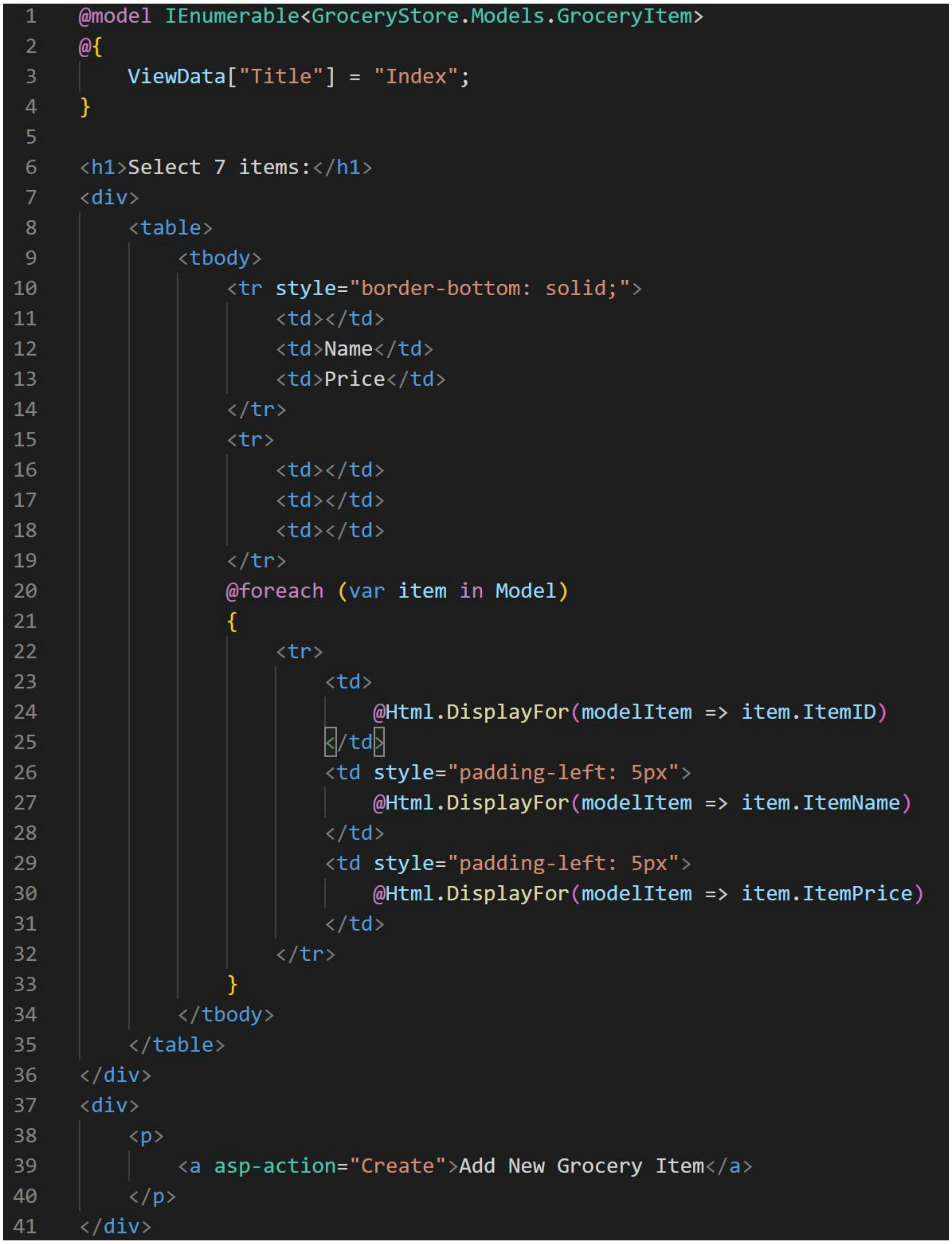


```
using GroceryStore.Repositories;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;
using GroceryStore.Models;

namespace GroceryStore.Controllers
{
    0 references
    public class GroceryController : Controller
    {
        4 references
        private IGroceryRepository _repository;

        0 references
        public GroceryController(IGroceryRepository repository)
        {
            _repository = repository;
        }

        // GET: Grocery
        0 references
        public ActionResult Index()
        {
            return View(_repository.GetItems());
        }

        // GET: Grocery/Create
        [HttpGet]
        0 references
        public ActionResult Create()
        {
            return View();
        }

        // POST: Grocery/Create
        [HttpPost]
        0 references
        public ActionResult Create(GroceryItem item)
        {
            _repository.AddItem(item);
            _repository.SaveChanges();

            return RedirectToAction("Index");
        }
```

7. Now lets work on the HTML portion
    a. Right Click on [Views Folder] inside GroceryStore Folder. Click [New Folder].
    b. Type [Grocery] and press enter.
    c. Right Click on [Grocery Folder] and click [New File].
    d. Type [Index.cshtml] and press enter.
    e. Right Click on [Grocery Folder] and click [New File].
    f. Type [Create.cshtml] and press enter.
    g. Your Folder should look like this.

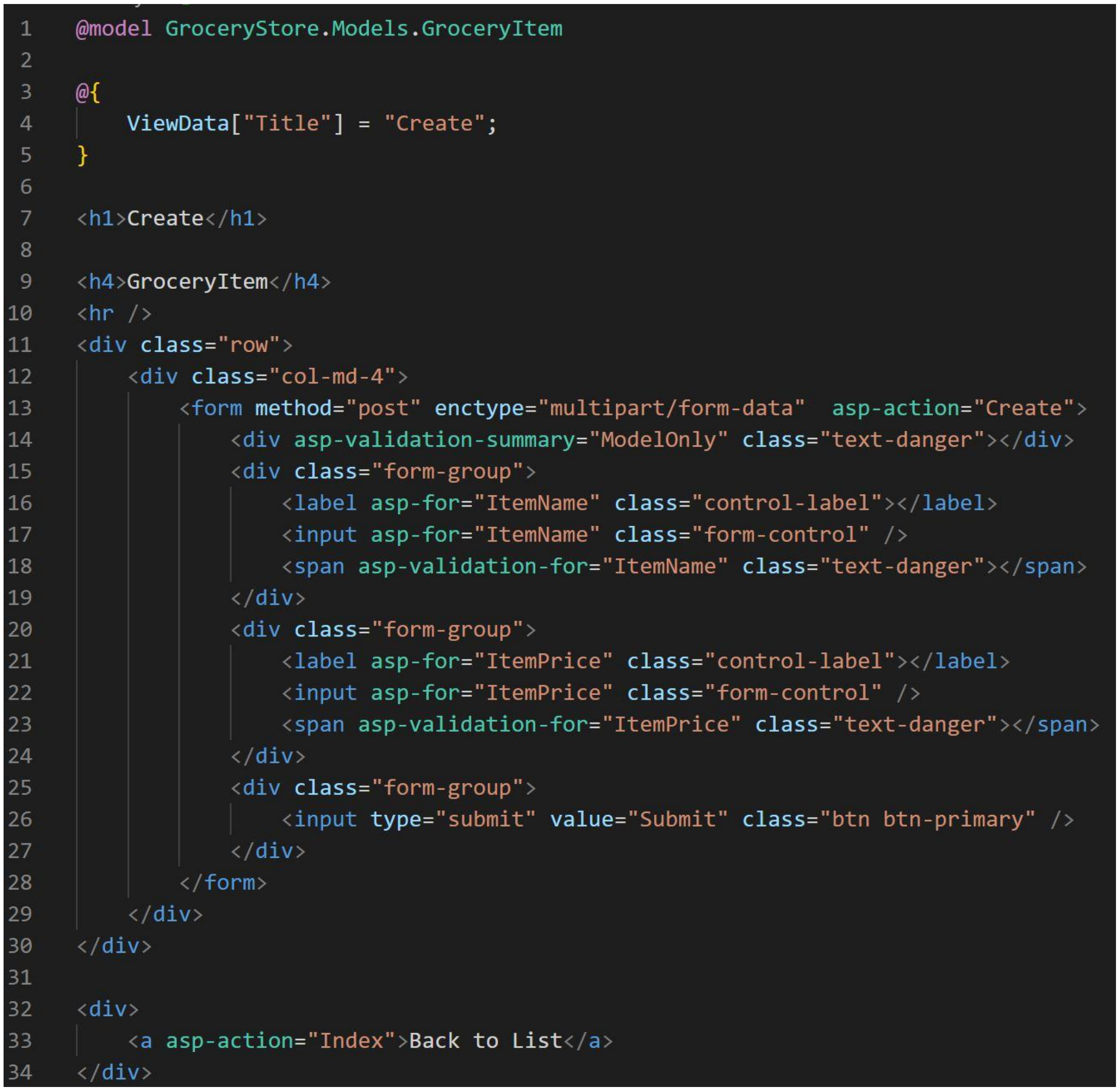


    h. Click on [Index.cshtml]
    i. Type following code inside



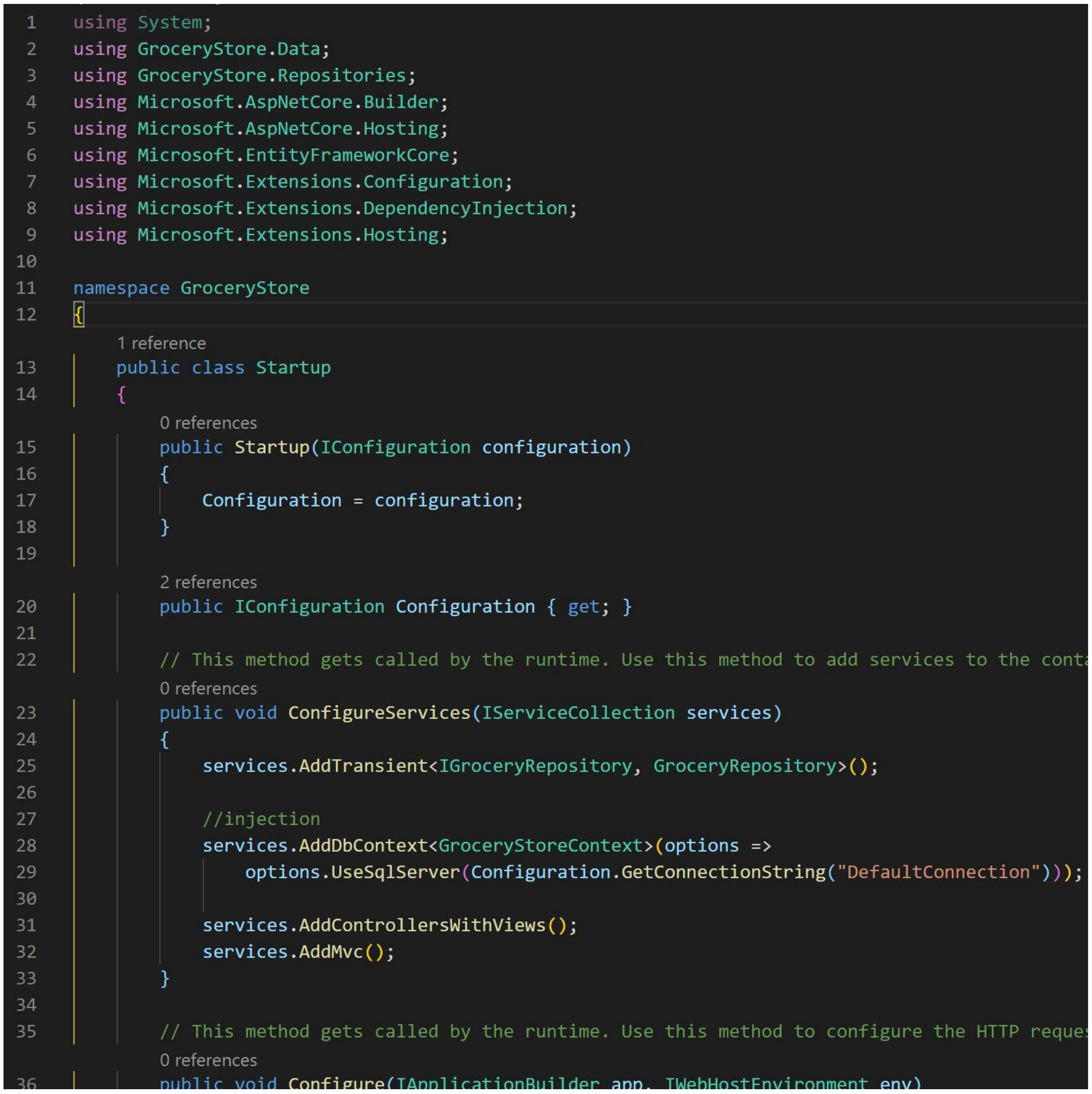
```
@model IEnumerable<GroceryStore.Models.GroceryItem>
@{
    ViewData["Title"] = "Index";
}

<h1>Select 7 items:</h1>
<div>
    <table>
        <tbody>
            <tr style="border-bottom: solid;">
                <td></td>
                <td>Name</td>
                <td>Price</td>
            </tr>
            <tr>
                <td></td>
                <td></td>
                <td></td>
            </tr>
            @foreach (var item in Model)
            {
                <tr>
                    <td>
                        @Html.DisplayFor(modelItem => item.ItemID)
                    </td>
                    <td style="padding-left: 5px">
                        @Html.DisplayFor(modelItem => item.ItemName)
                    </td>
                    <td style="padding-left: 5px">
                        @Html.DisplayFor(modelItem => item.ItemPrice)
                    </td>
                </tr>
            }
        </tbody>
    </table>
</div>
<div>
    <p>
        <a asp-action="Create">Add New Grocery Item</a>
    </p>
</div>
```

j. Click on [Create.cshtml]
k. Type following code inside

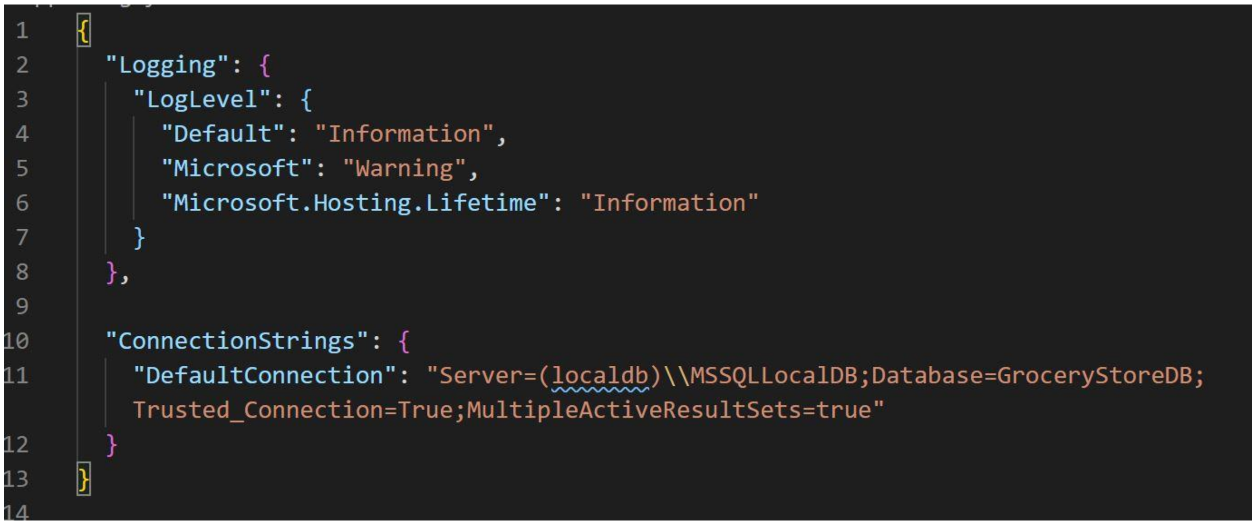


```
@model GroceryStore.Models.GroceryItem

@{
    ViewData["Title"] = "Create";
}

<h1>Create</h1>

<h4>GroceryItem</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post" enctype="multipart/form-data"  asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="ItemName" class="control-label"></label>
                <input asp-for="ItemName" class="form-control" />
                <span asp-validation-for="ItemName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ItemPrice" class="control-label"></label>
                <input asp-for="ItemPrice" class="form-control" />
                <span asp-validation-for="ItemPrice" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Submit" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>
```

8. Use Entity Framework Core to connect to Microsoft SQL Server
   a. Click [Startup.cs] inside GroceryStore Folder.
   b. Update the existing code.


```
using System;
using GroceryStore.Data;
using GroceryStore.Repositories;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace GroceryStore
{
    1 reference
    public class Startup
    {
        0 references
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        2 references
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the conta
        0 references
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddTransient<IGroceryRepository, GroceryRepository>();

            //injection
            services.AddDbContext<GroceryStoreContext>(options =>
                options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

            services.AddControllersWithViews();
            services.AddMvc();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP reques
        0 references
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
```

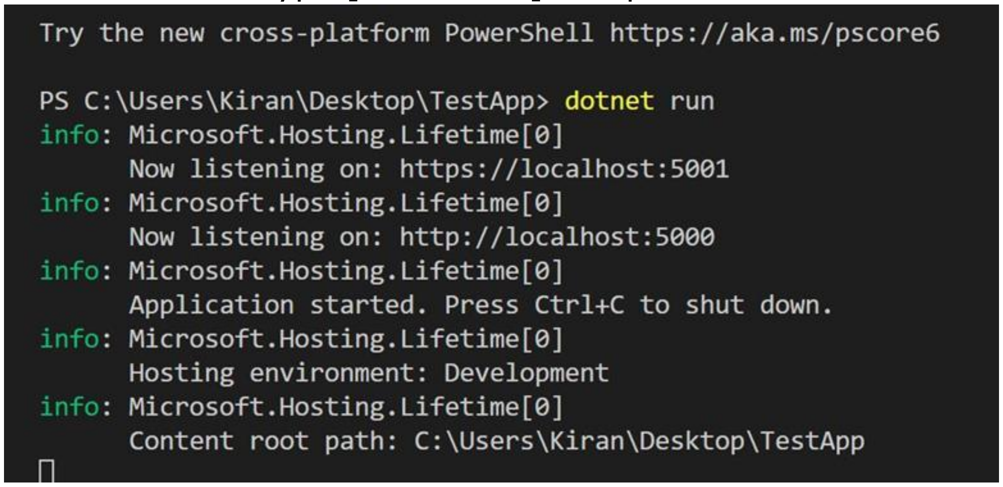
9. Specify a connection string in a configuration file
    a. Click [appsettings.json] inside GroceryStore Folder.
    b. Update existing file.



```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },

  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=GroceryStoreDB;
    Trusted_Connection=True;MultipleActiveResultSets=true"
  }
}
```

10. Finally, in order to create database in Microsoft server, we need to add MIGRATION pacakage.
    a. Click [View Menu] on the top and click Terminal
    b. Inside terminal Type [dotnet ef migrations add GroceryItemList - -context GroceryStoreContext] and press enter.
    c. If you see inside GroceryStore Folder, VS Code has just added Migrations Folder
    d. Inside terminal Type [dotnet ef database update - -context GroceryStoreContext] and press enter.
    e. This should add the table inside the database.

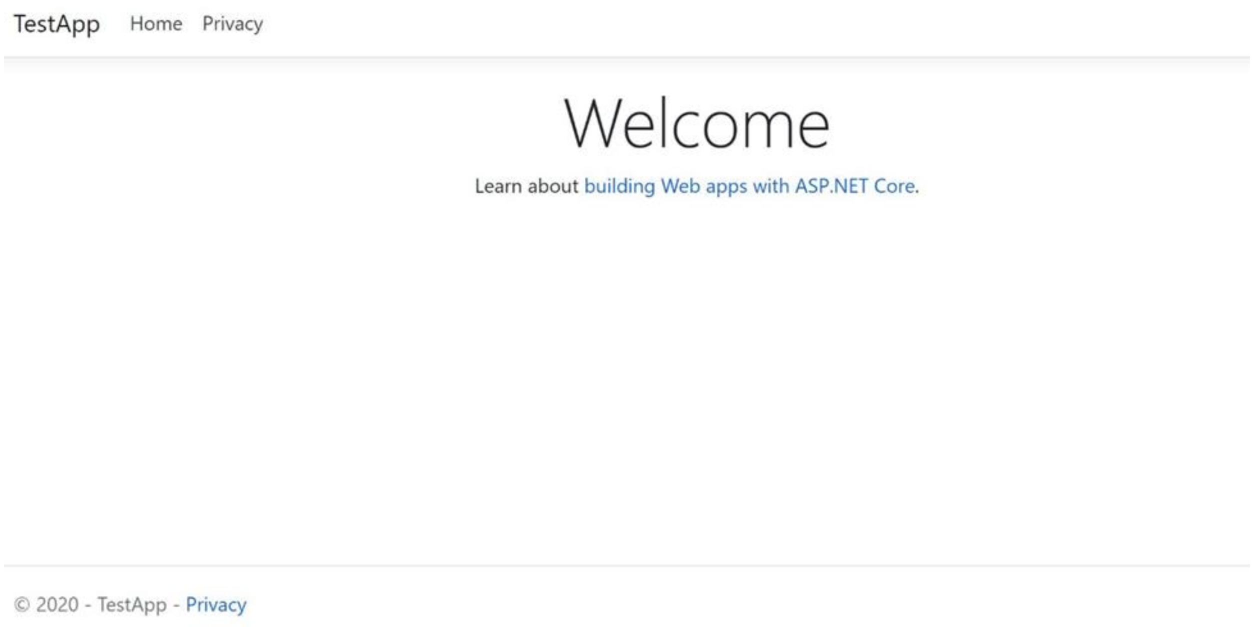
11. Time to run the application
    a. Inside Termina Type [dotnet run] and press enter.



```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Kiran\Desktop\TestApp> dotnet run
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Kiran\Desktop\TestApp
```

    b. Open a web browser and in the address bar Type [ localhost:5001 ]

c. More then likely, it is going to give you an error because 5001 is a secure line and we don’t have certificate created yet.
d. Click on the [ Advance button ] and click [ continue ]
e. You should get a web page that looks like this. (see image below)

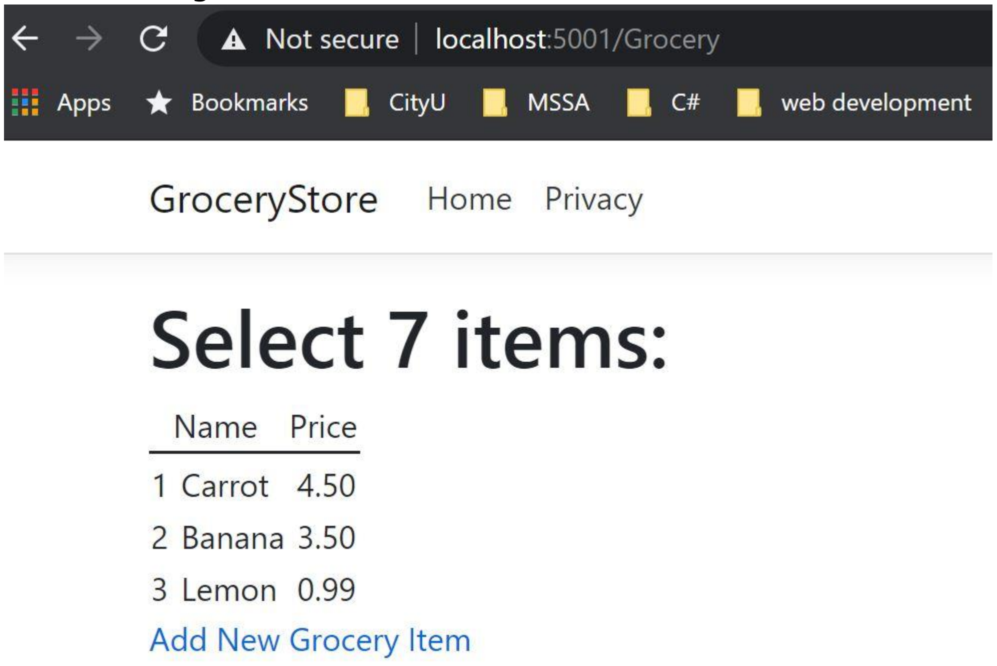


Welcome

Learn about building Web apps with ASP.NET Core.

© 2020 - TestApp - Privacy

f. Go to the address bar and add [localhost:5001/Grocery] and press enter
g. You should get this.



Select 7 items:

|  | Name | Price |
|---|---|---|
| 1 | Carrot | 4.50 |
| 2 | Banana | 3.50 |
| 3 | Lemon | 0.99 |

Add New Grocery Item