**Objective:**

In this lab exercise, you will create an ASP.NET Core MVC project, use Bootstrap for styling the layout view, create a model class, and scaffold a CRUD controller and views for the model.

**Requirements:**

1. Install the .NET SDK if you haven't already. (<https://dotnet.microsoft.com/en-us/download/visual-studio-sdks>)
2. Install Visual Studio or Visual Studio Code. (<https://visualstudio.microsoft.com/>)
3. Completed LAB01

**Steps:**

1. Create a new ASP.NET Core MVC project:
	* Open Visual Studio or Visual Studio Code.
	* Create a new project using the "ASP.NET Core Web App (Model-View-Controller)" template.
	* Name the project **"**LabWebApp**"**and choose a location to save it.
		+ Note: you should save this in the repository folder created in LAB01 under its own subfolder
2. Use Bootstrap for styling the layout view:
	* In the "wwwroot" folder, you'll find the "lib" folder, which contains the Bootstrap library.
	* Open the "\_Layout.cshtml" file located in the "Views" > "Shared" folder.
	* Make sure that the necessary Bootstrap CSS and JS files are referenced in the "\_Layout.cshtml" file.
		+ They should be added by default in a new ASP.NET Core MVC project.
		+ If not, add the following lines within the \<head> tag for the CSS:
			- <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
		+ And add the following lines before the closing </body> tag for the JS files:
			- <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
3. Create a model:
	* In the "Models" folder, create a new class named "Product.cs".
	* Add properties to the "Product" class:
		+ public int Id { get; set; }
		+ public string Name { get; set; }
		+ public decimal Price { get; set; }
		+ public string Description { get; set; }
4. Set up Entity Framework Core and create a DbContext:
	* Install the necessary NuGet packages for Entity Framework Core:
		+ dotnet add package Microsoft.EntityFrameworkCore.SqlServer
		+ dotnet add package Microsoft.EntityFrameworkCore.Tools
	* Create a new folder named "Data" in the project.
	* In the "Data" folder, create a new class named "ApplicationDbContext.cs" and make it inherit from DbContext.
	* In the "ApplicationDbContext" class, add a DbSet<Product> property named "Products".
	* For the connection string
		+ In the "appsettings.json" file, add a '<secret>' value to the connection string for the database.
		+ Add the actual connection string to the database in the user-secrets of your application.
	* In the "Program.cs" file, register the ApplicationDbContext in the Configure Services section.
	* Add migrations, and update database accordingly
5. Scaffold a CRUD controller and views for the "Product" model:
	* In the "Controllers" folder, right-click and select "Add > Controller".
	* Choose the "MVC Controller with views, using Entity Framework" template.
	* In the "Add MVC Controller with views, using Entity Framework" dialog, select the "Product" model class and the "ApplicationDbContext" data context class.
	* Click "Add" to generate the controller and views.
6. Run the application:
	* Press F5 or click the "Run" button to launch the application.
	* Navigate to the "Products" controller (e.g., <https://localhost:5001/Products>) to see the CRUD functionality in action.

**Deliverables:**

1. Commit and Push all your files to the GitHub repository created in LAB01 under its own folder. (You'll keep all your assignments and labs separate this way, but in the same folder)
2. Submit the link to your private repository.

In this lab exercise, you have successfully created an ASP.NET Core MVC project, applied Bootstrap styling to the layout view, created a model, and scaffolded