**Building Product Management Application Using Real-Time Communication with SignalR**

# Introduction

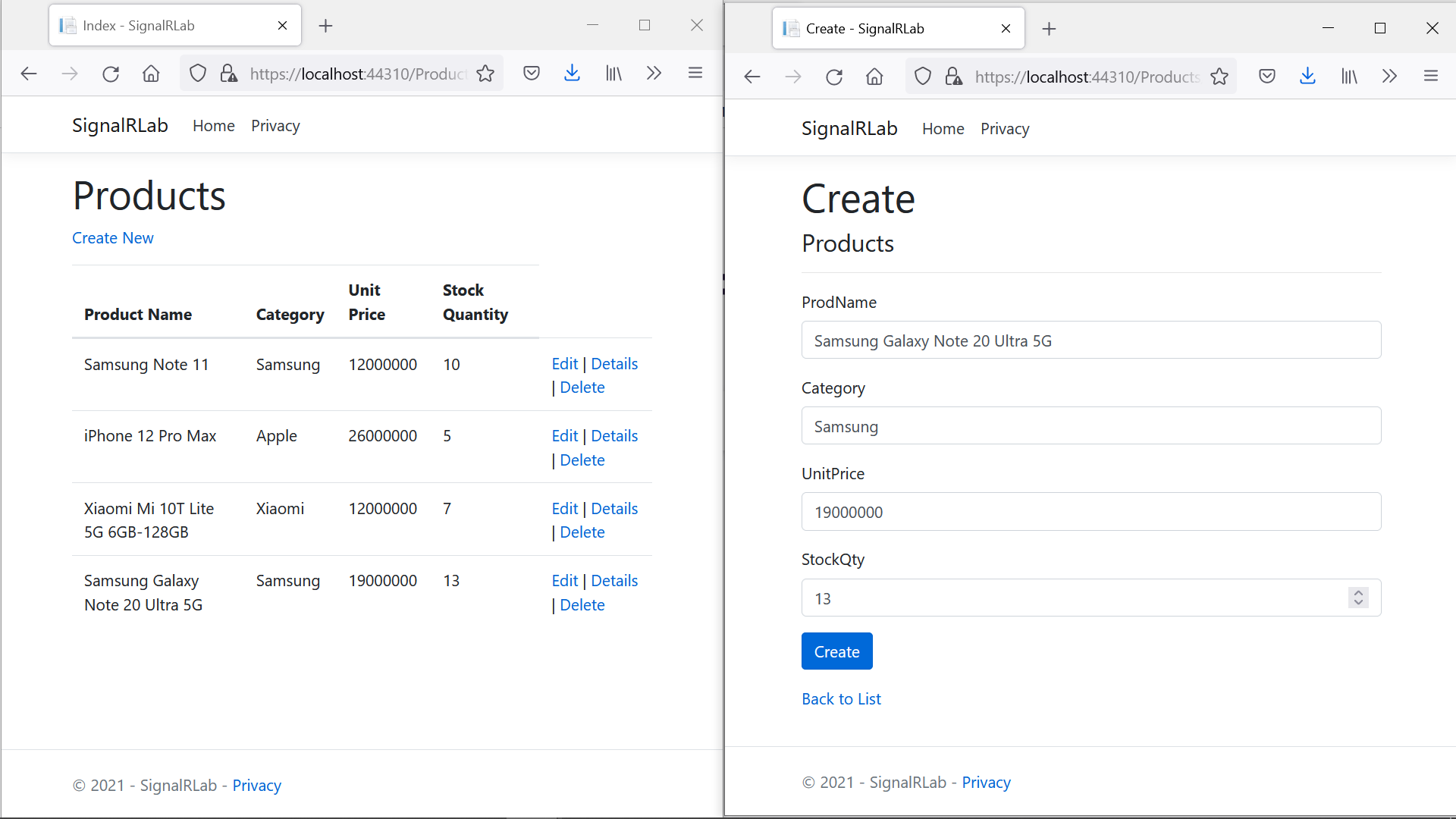
Imagine you're an employee of a product retailer named Product Store. Your manager has asked you to develop a web application for product management Product (ProductID, ProductName, Category, Quantity, Price). The application has to support adding, viewing, modifying, and removing products—a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD).

This lab explores creating an application using Real-Time Communication with SignalR, ASP.NET Core, and C#. An **SQL Server** **Database** will be created to persist the product data that will be used for reading and managing product data by **Entity Framework Core.**

# Lab Objectives

In this lab, you will:

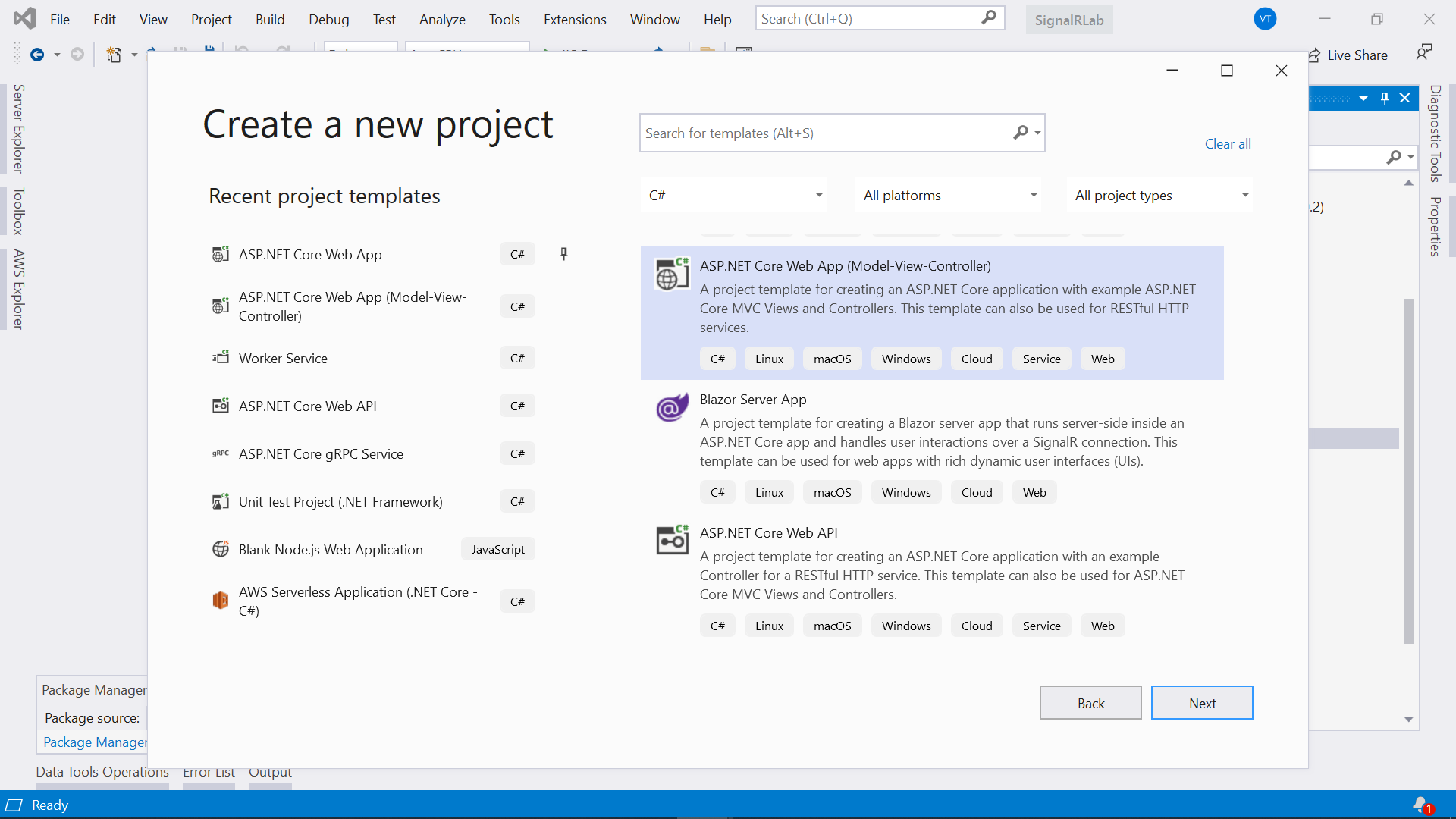
* Use the Visual Studio.NET to create ASP.NET Core Web Application Project.
* Develop application using MVC Pattern.
* Use Entity Framework to Create a SQL Server database named SignalRLab that has a Product table.
* Develop Entity Classes a and DBContext class to perform CRUD actions using Entity Framework Core.
* Apply JavaScript library for Real-Time communication application.
* Run the project and test the application actions.



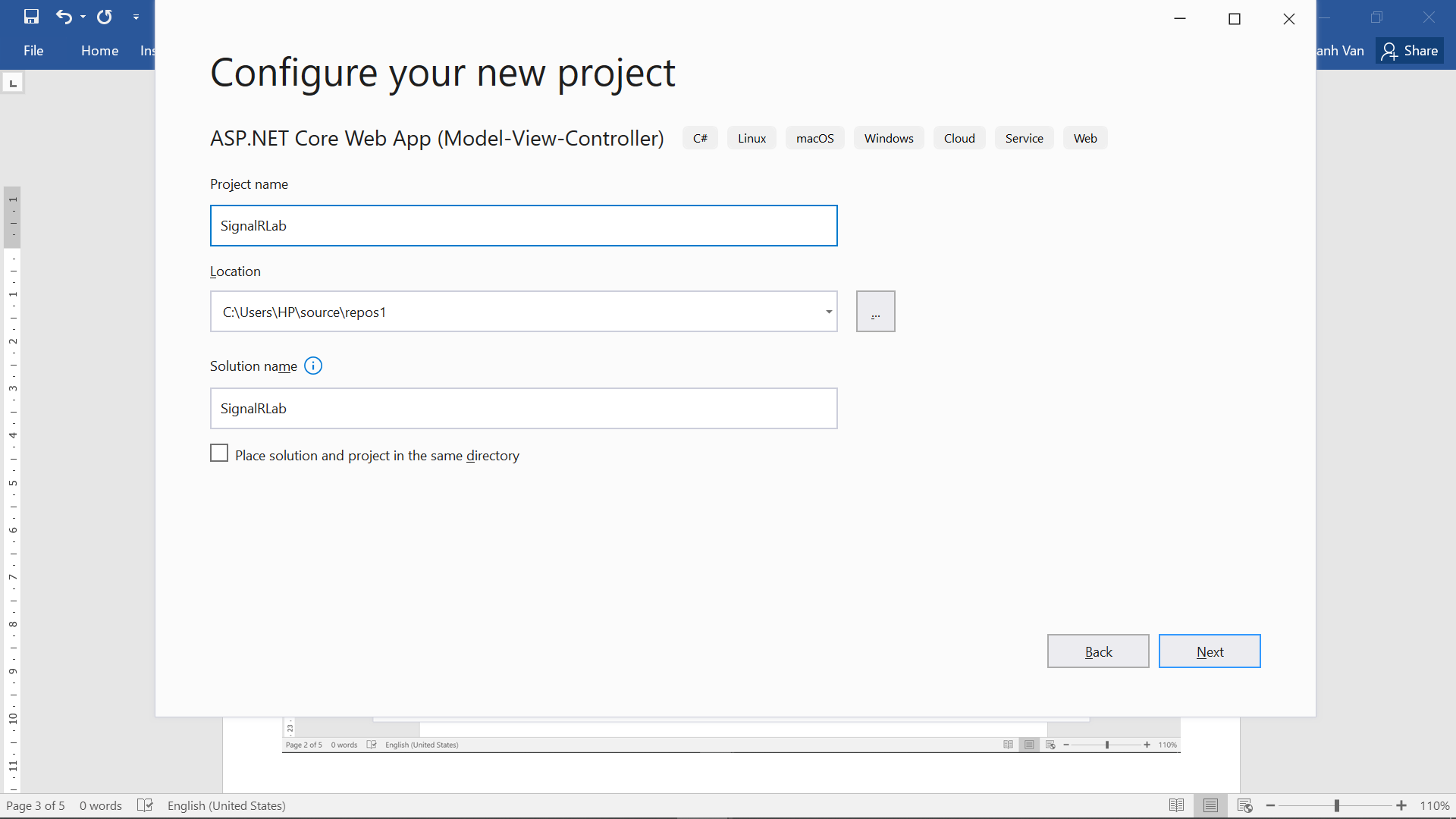
# Activity 01: Create Project

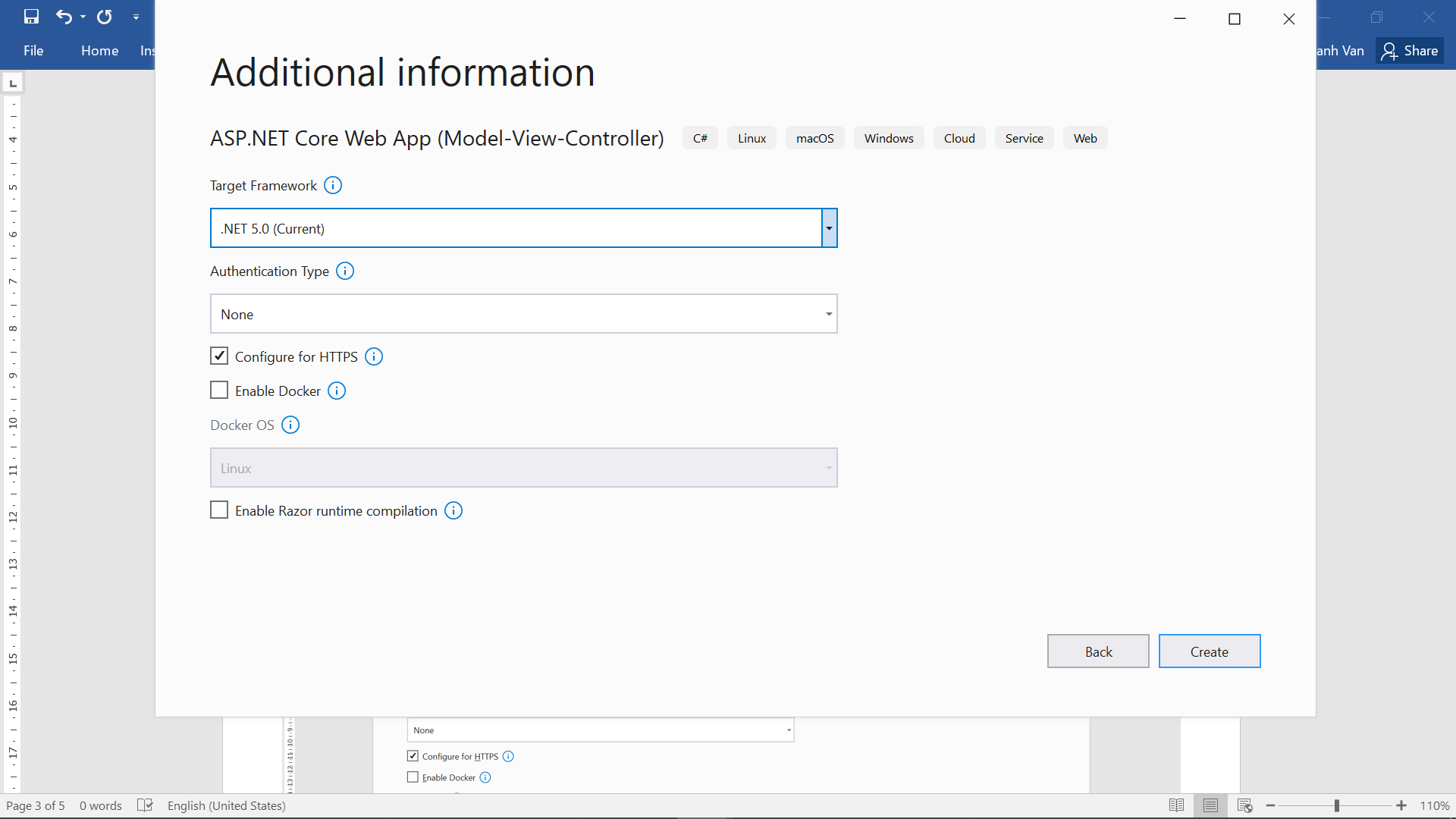
Create a Solution named SignalRLab. Create ASP.NET Core Web Application (Model-View-Controller). Open the Visual Studio .NET application and performs steps as follows:

Create ASP.NET Core Web App (Model-View-Controller)

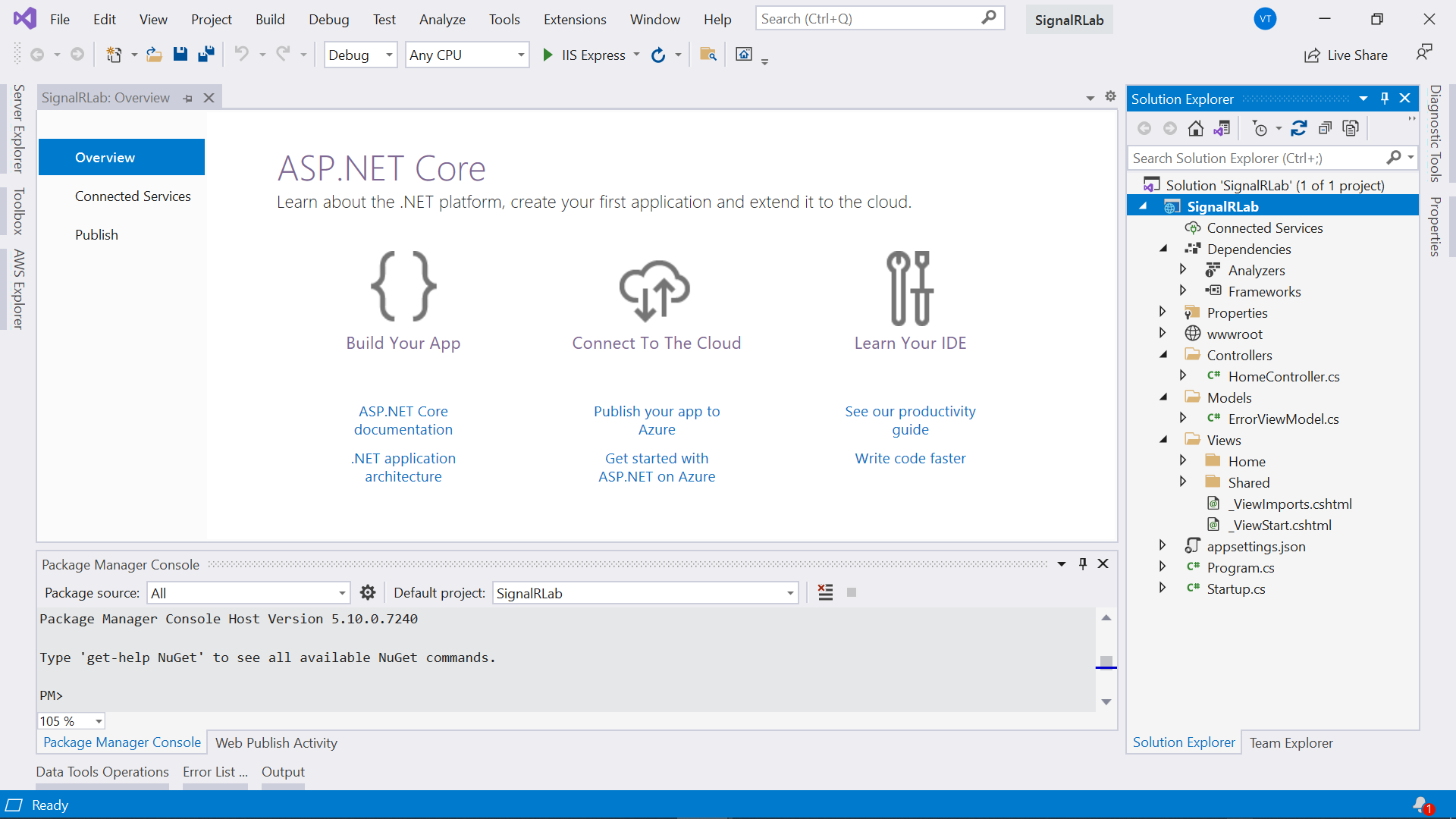


Specify Name and Location of the Project “SignalRLab”



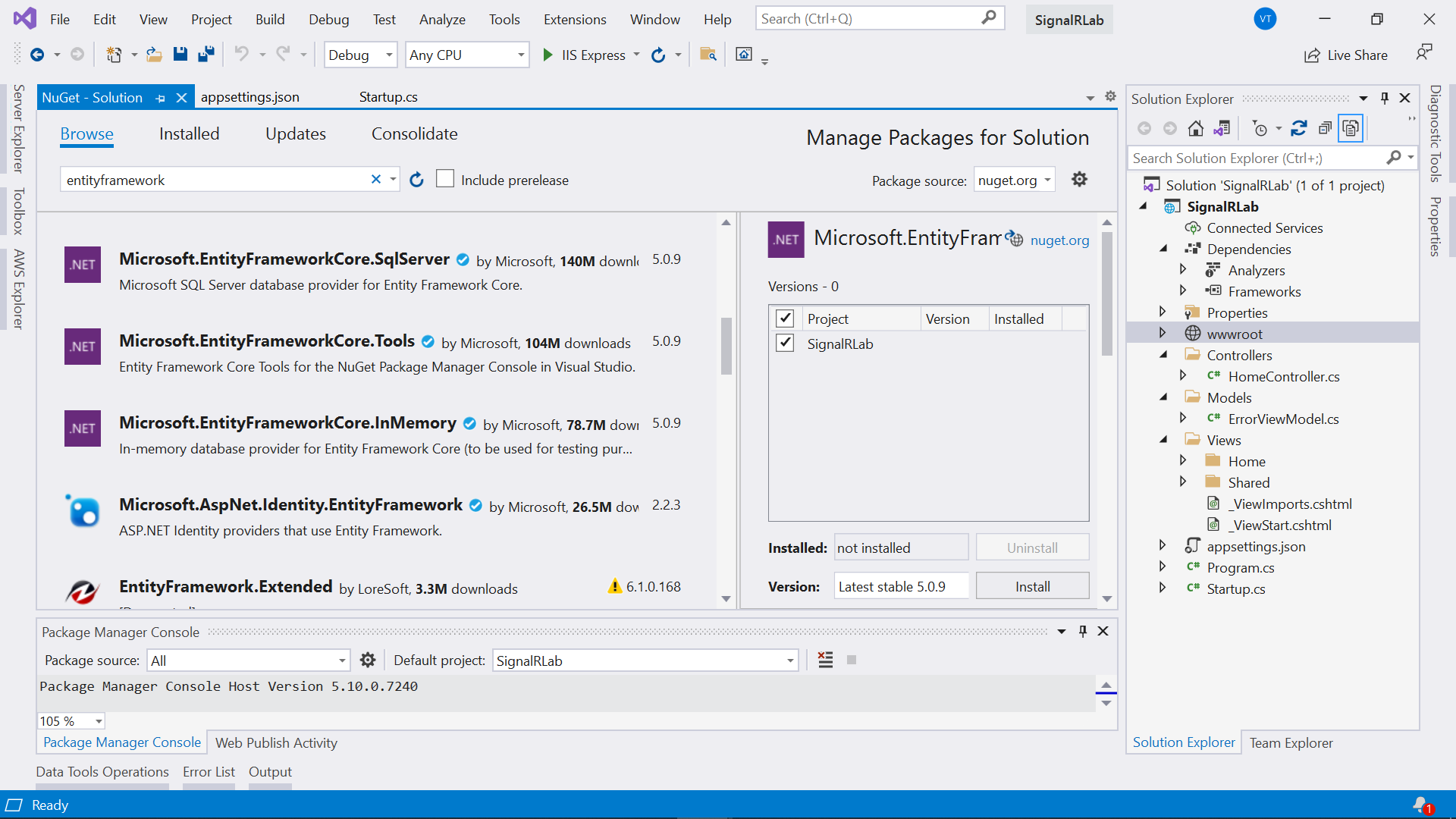


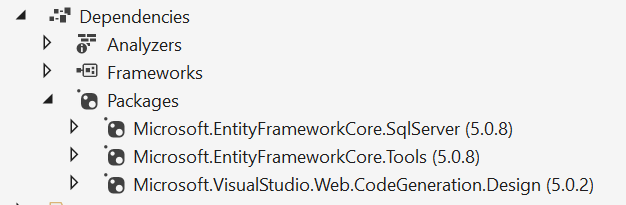
The structure of ASP.NET Core Web Application Project “SignalRLab”.



# Activity 02: Work with Entity Framework

**Step 01**. Install the following packages from NuGet:





**Step 02**. Add Connection string (appsettings.json file)

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

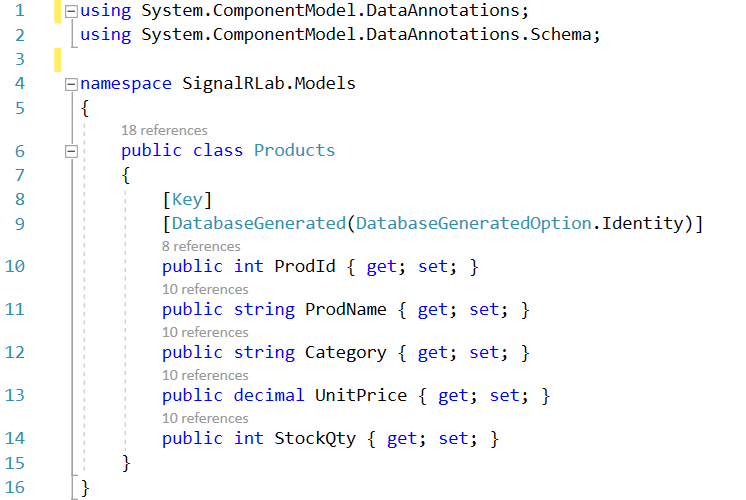
**"ConnectionStrings": {**

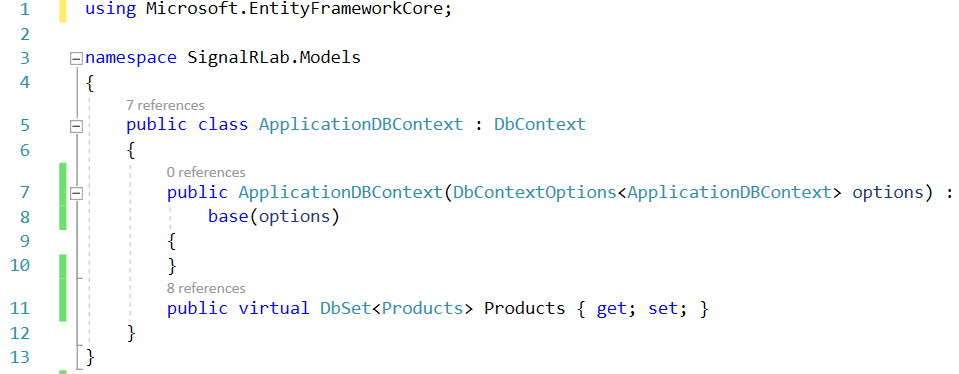
**"DefaultConnection": "Persist Security Info=False;User ID=sa;Password=1234567890;Initial Catalog=SignalRLabDB;Data Source=.;Connection Timeout=100000"**

**}**

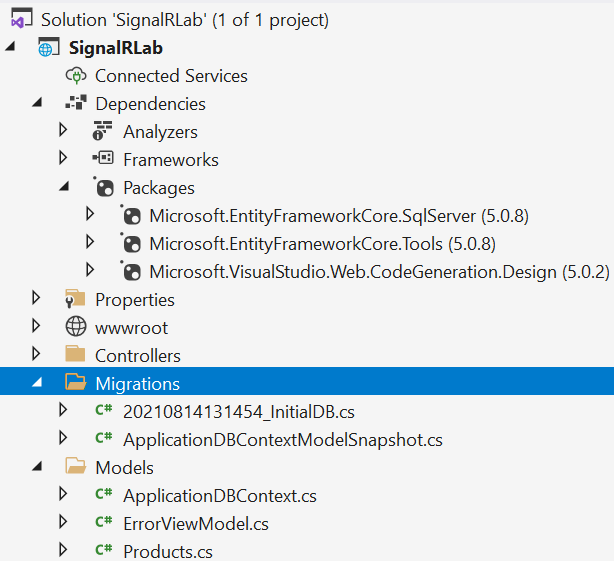
}

**Step 03**. Add “Products.cs” entity and “ApplicationDBContext.cs” classes

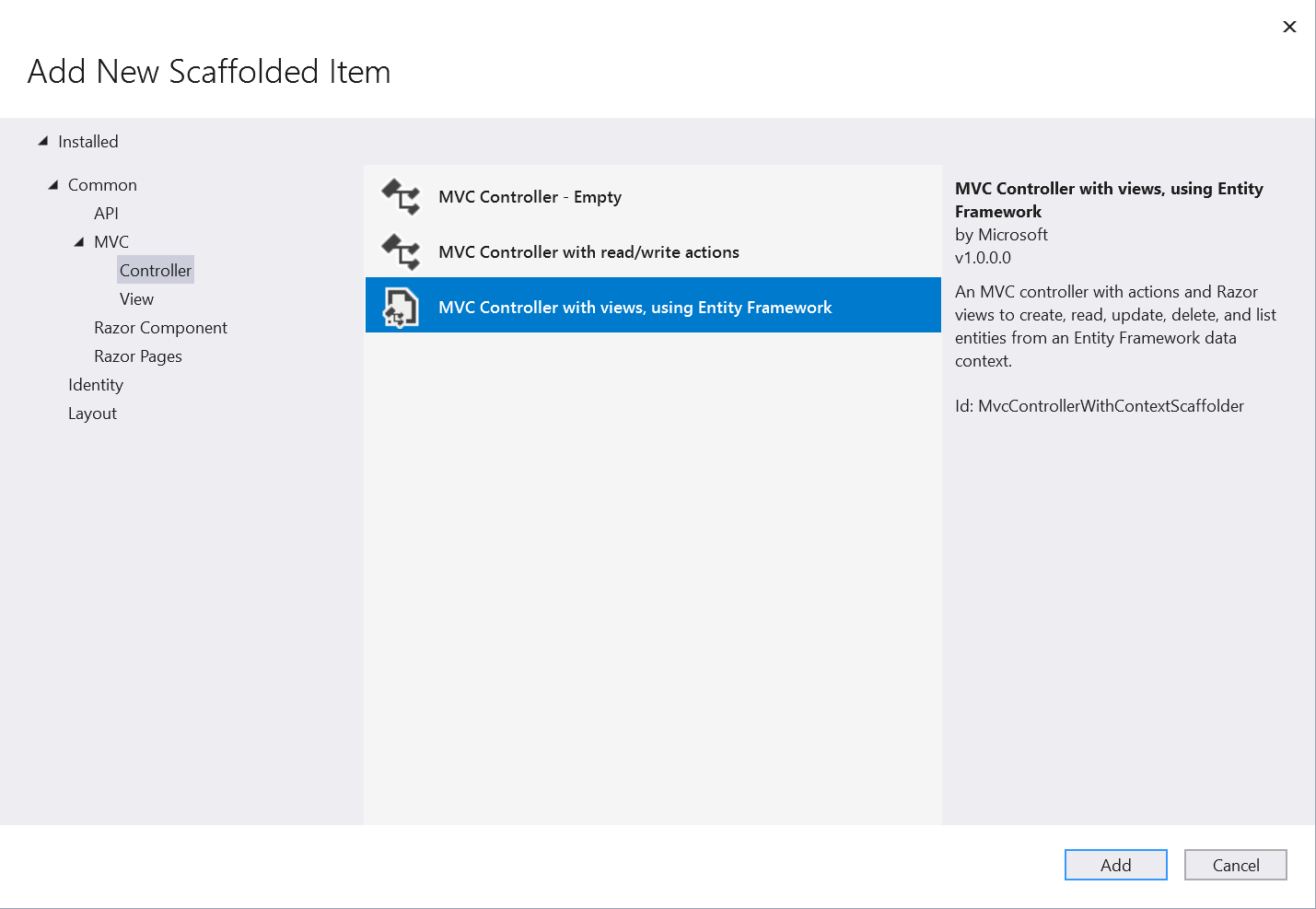


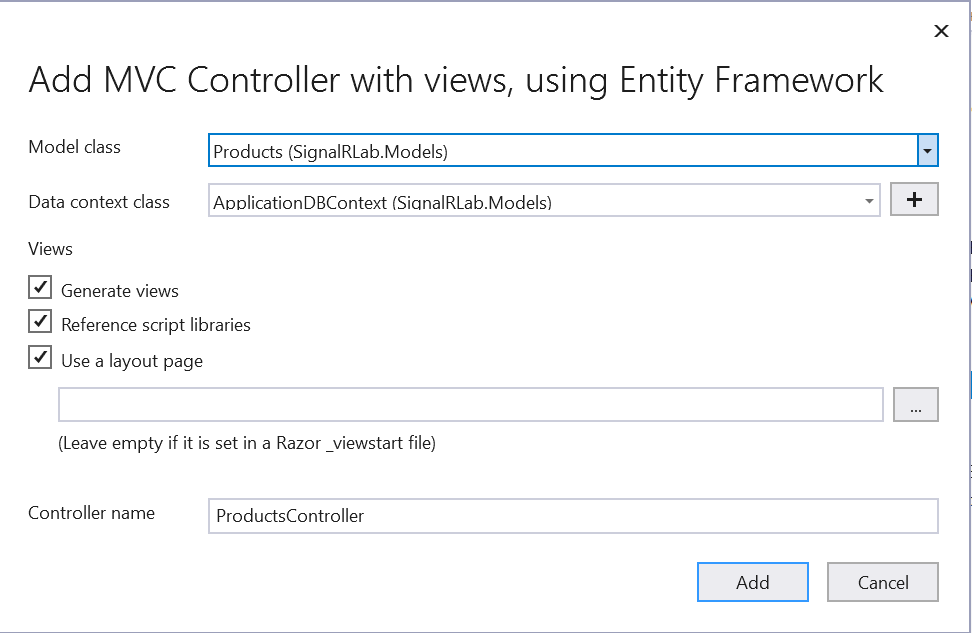


**Step 04**. Add-Migration and Update-Database



**Step 05**. Add ProductsController with Scraffolding





# Activity 03: Create SignalR Hub and configure SignalR

**Step 01**. Create SignalR Hubs in the SignalrServer.cs

# 

**Step 02.** Add SignalR to Startup.cs

# 

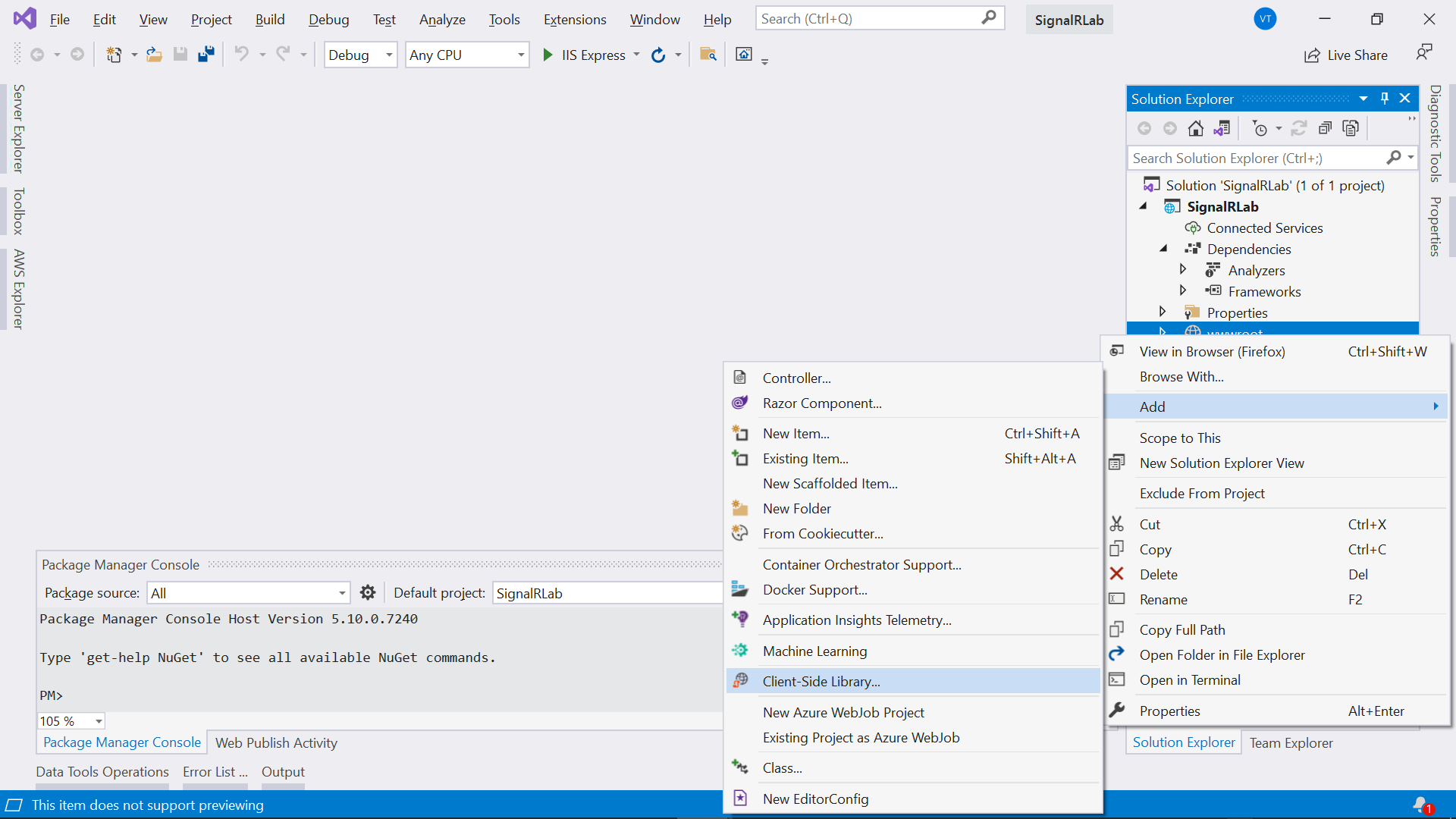
# 

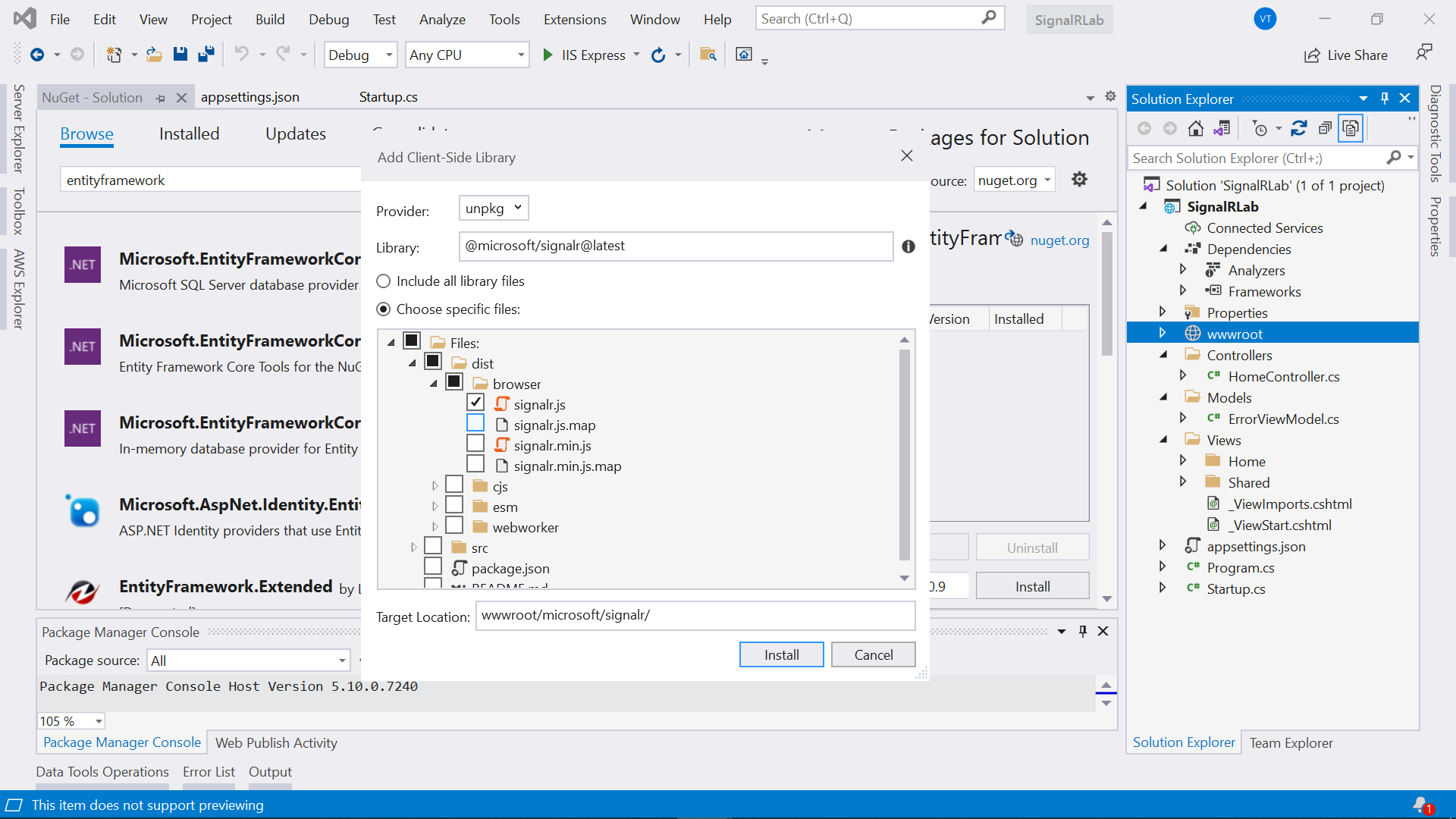
# 

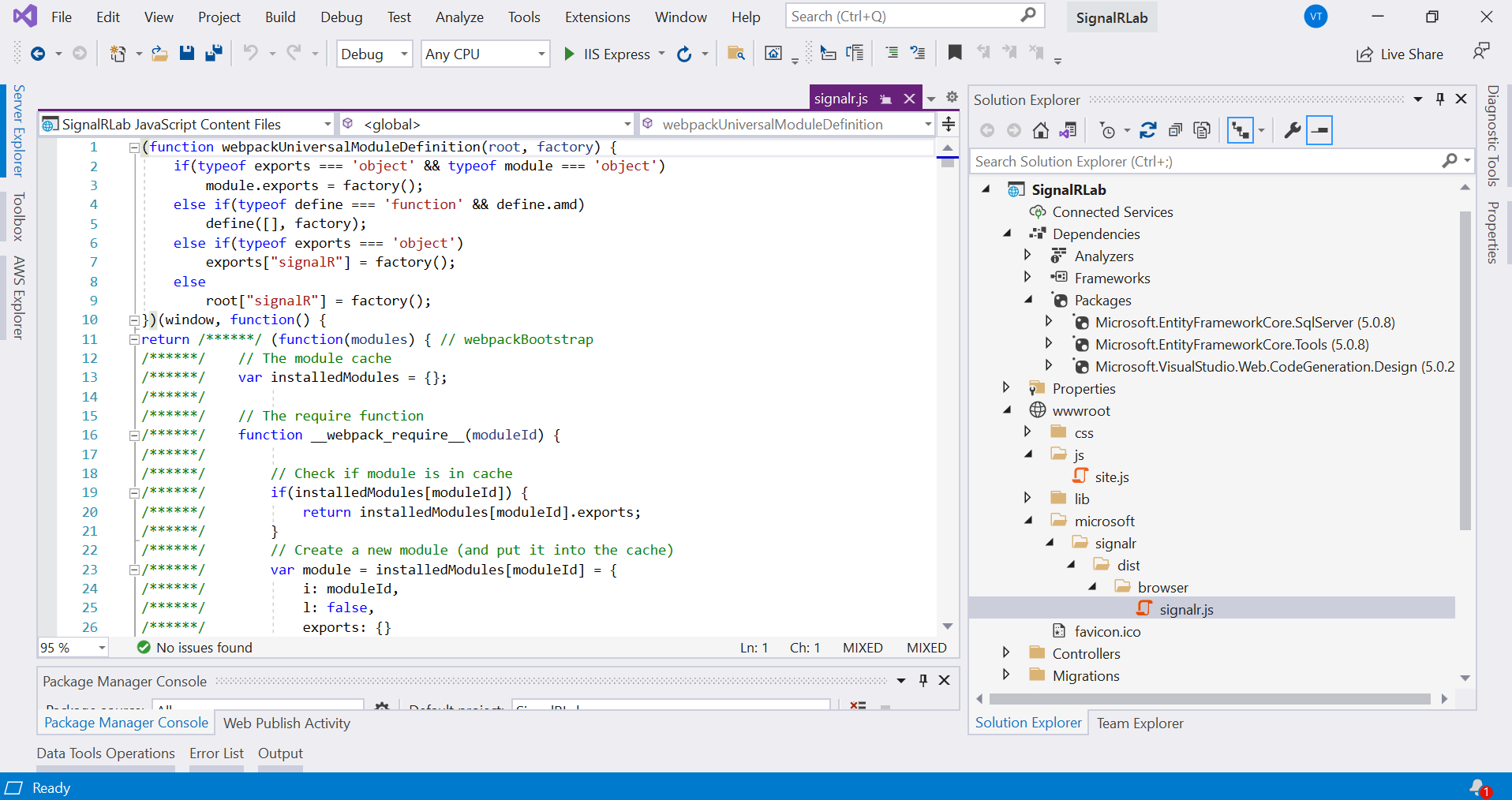
# Activity 04: Build CRUD functions with SignalR

**Step 01**. Add Client-Side Library

* Right-click the project, and select Add > **Client-Side Library**.
* In the Add Client-Side Library dialog, for Provider select **unpkg**.
* For Library, enter **@microsoft/signalr@latest**.
* Select Choose specific files, expand the *dist/browser* folder, and select ***signalr.js* and *signalr.min.js***.
* Set Target Location to ***wwwroot/microsoft/signalr/***, and select Install.

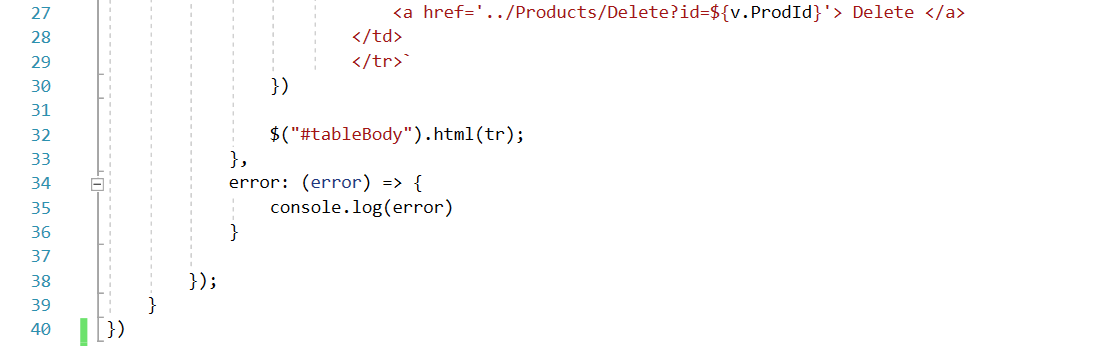






**Step 02**. Create a callback function in the script (site.js)





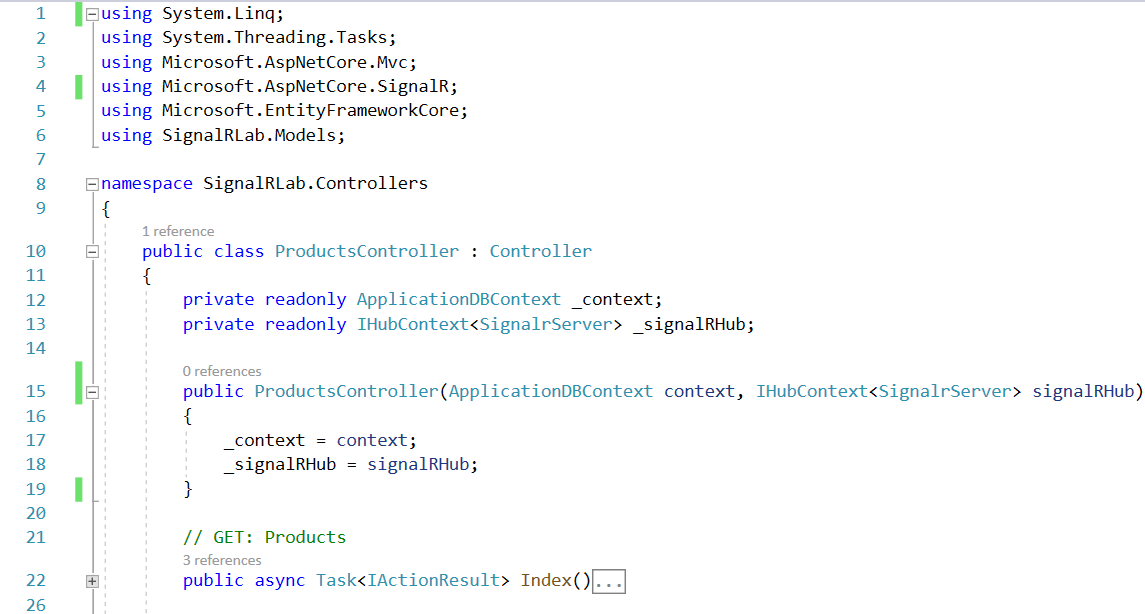
**Step 03**. Add the notification to CRUD actions

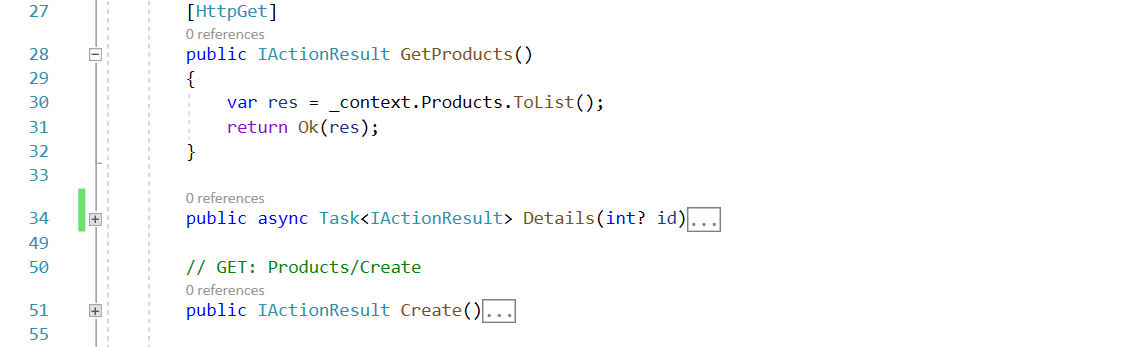
* The SignalR hub is the core abstraction for sending messages to clients connected to the SignalR server.
* Use a SignalR IHubContext to send notifications to clients from outside a hub.

private readonly IHubContext<SignalrServer> \_signalRHub;

* Then use an instance of IHubContext, call client methods as if you were in the hub itself

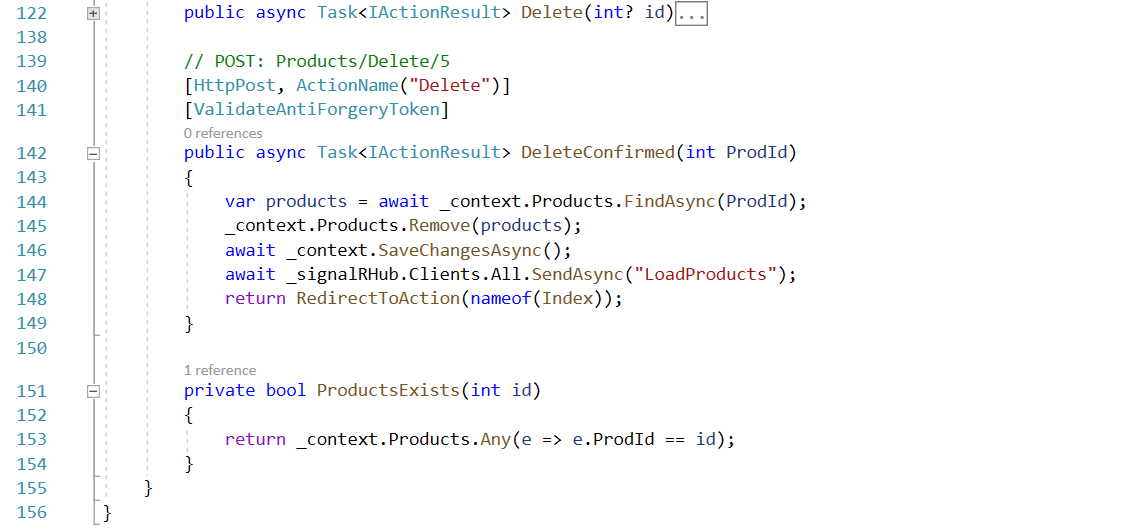
await \_signalRHub.Clients.All.SendAsync("LoadProducts");











**Step 04**. Add SignalR JavaScript client to View

Views/Shared/\_Layout.cshtml

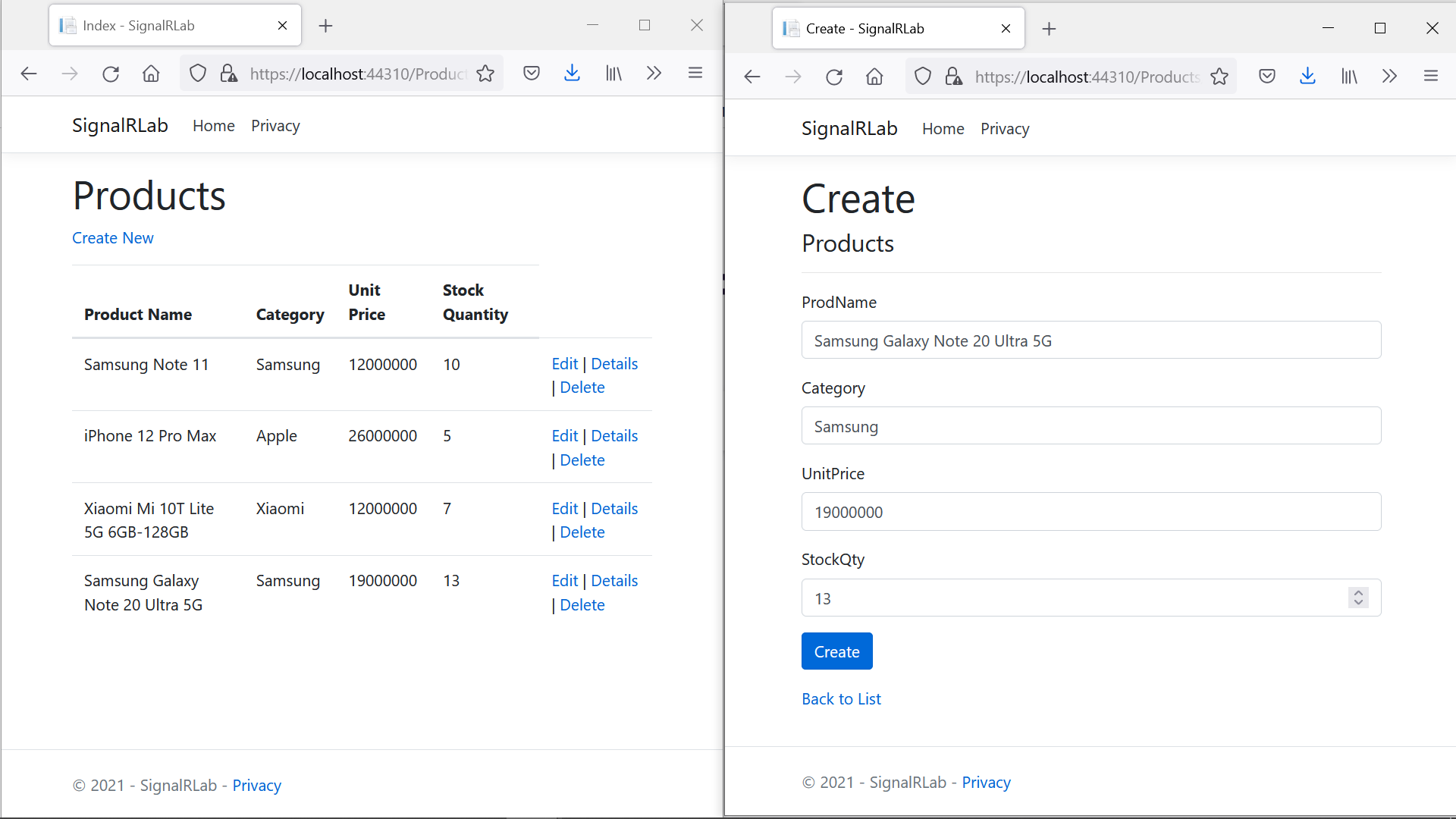


Views/Products/Index.cshtml



# Activity 05: Build and run Project. Test all CRUD actions

# 



Client 1 Client 2