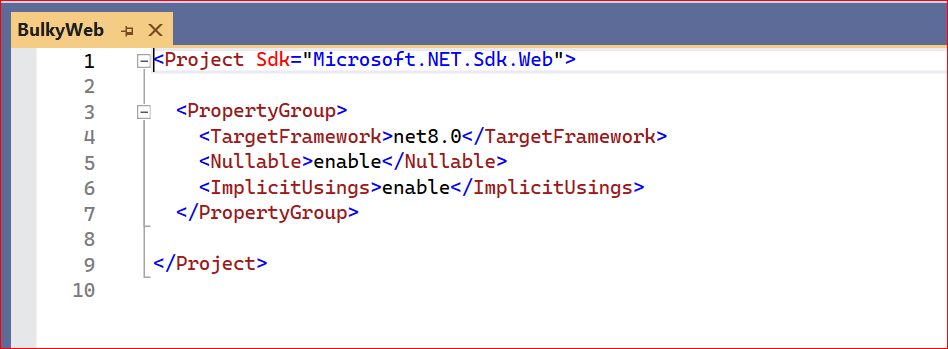
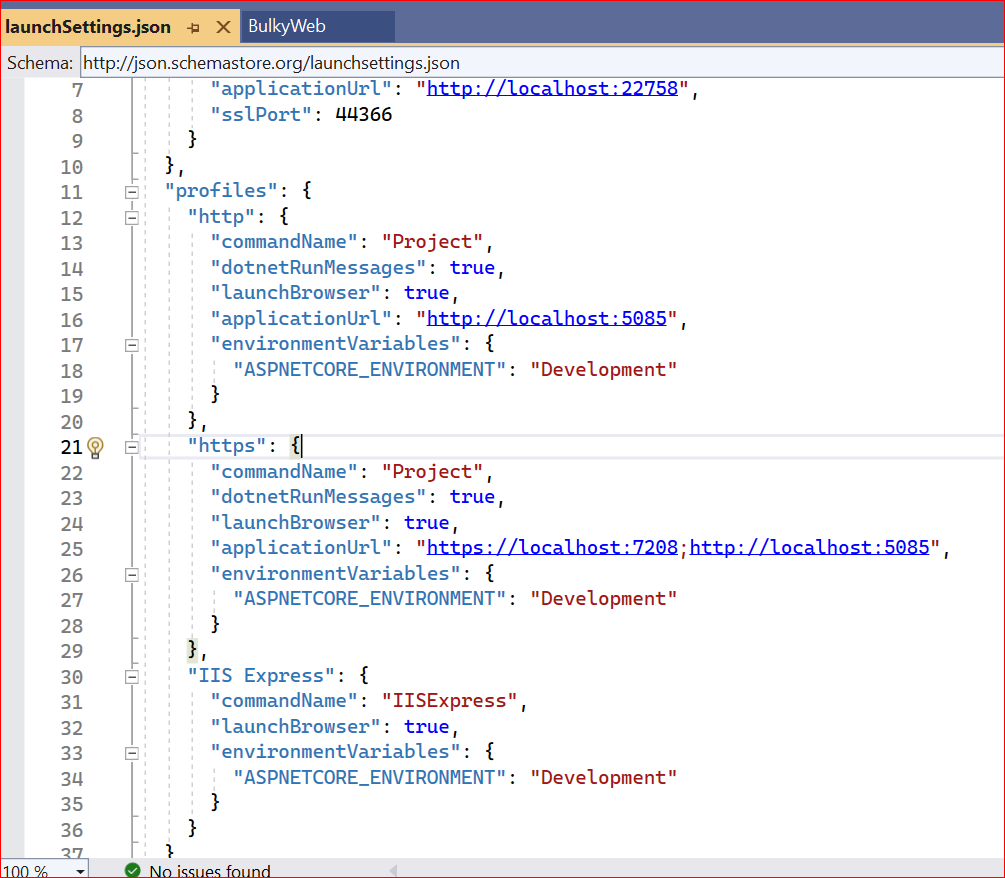


<https://bulky.azurewebsites.net/>

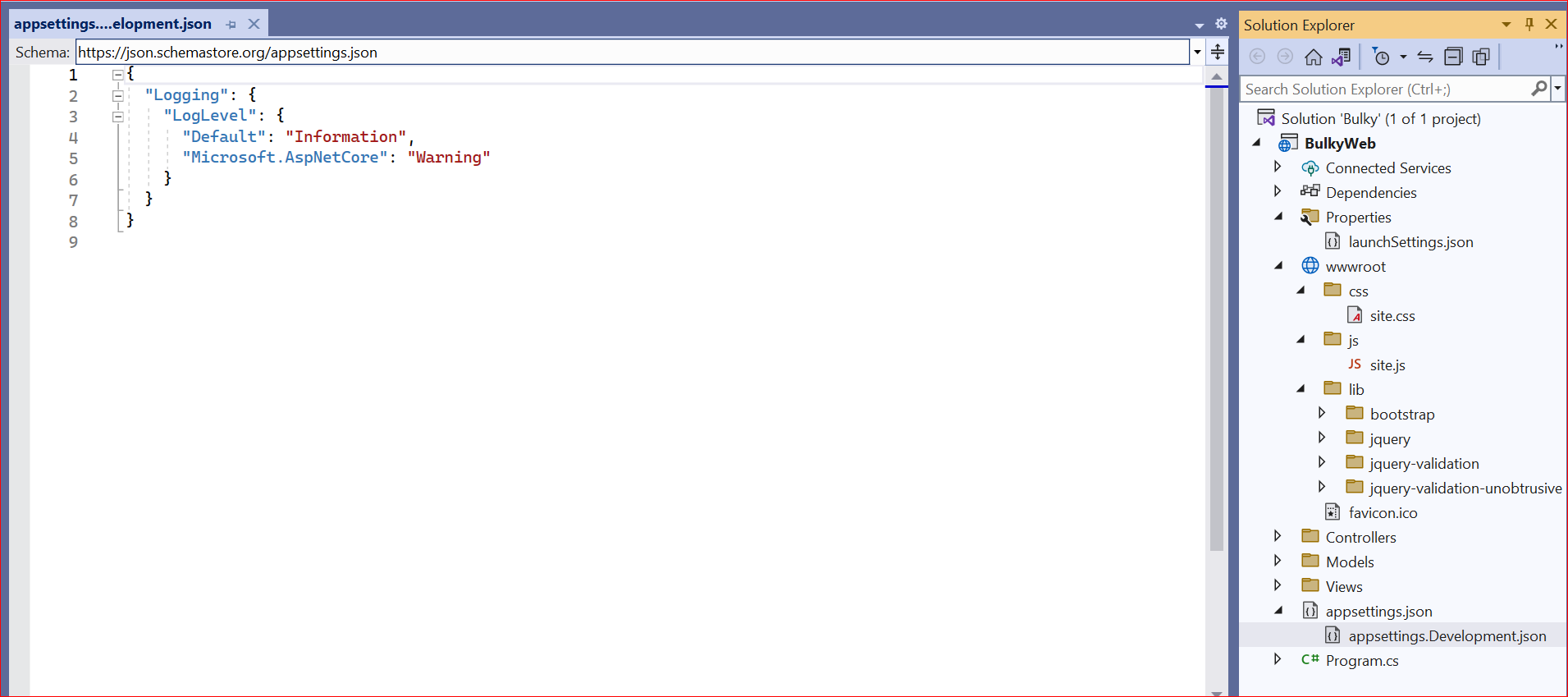
project file:



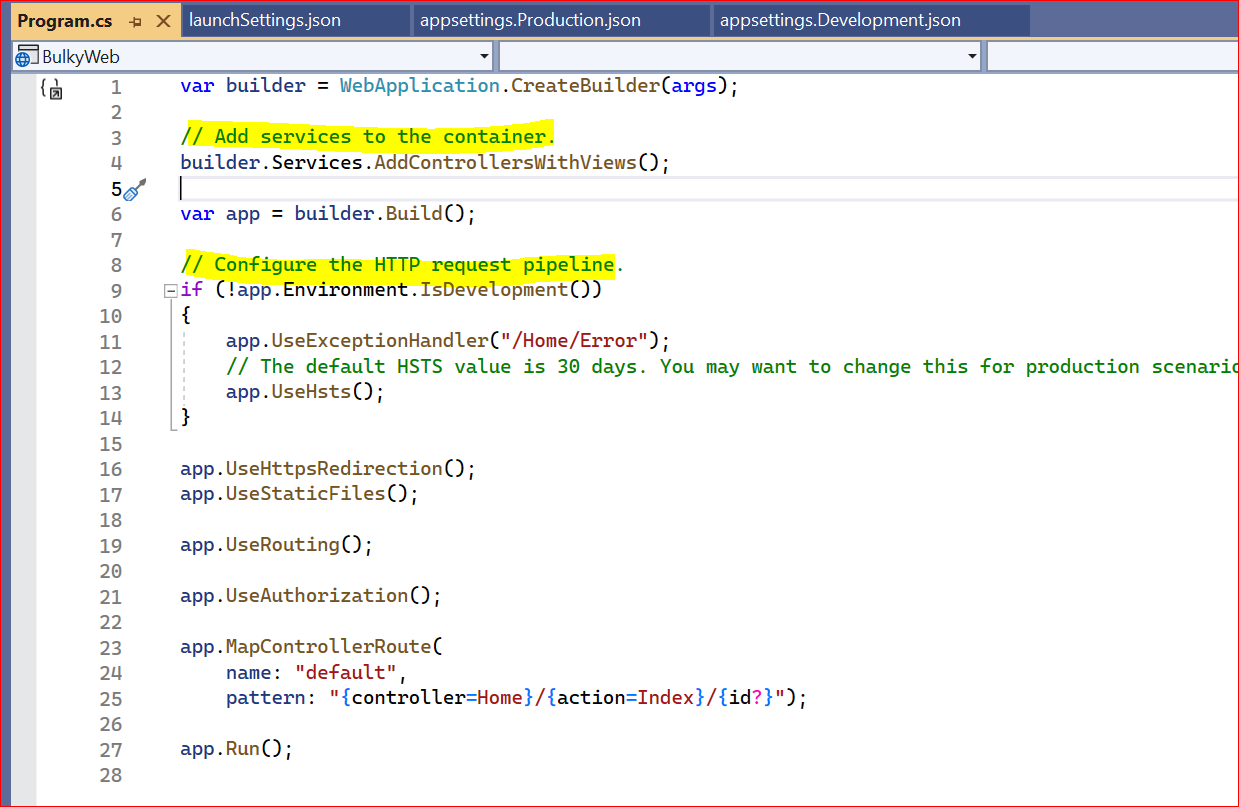
**LunchSetting:**



**Wwwroot: Static content**

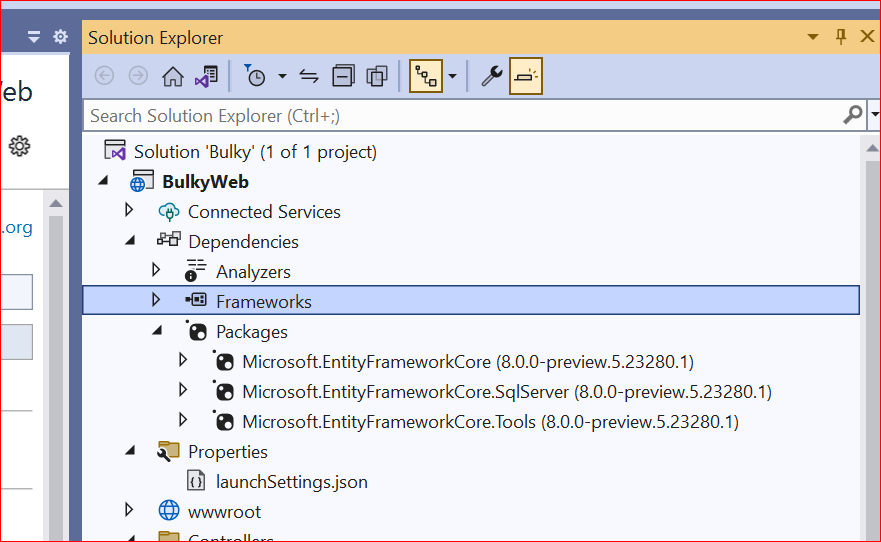


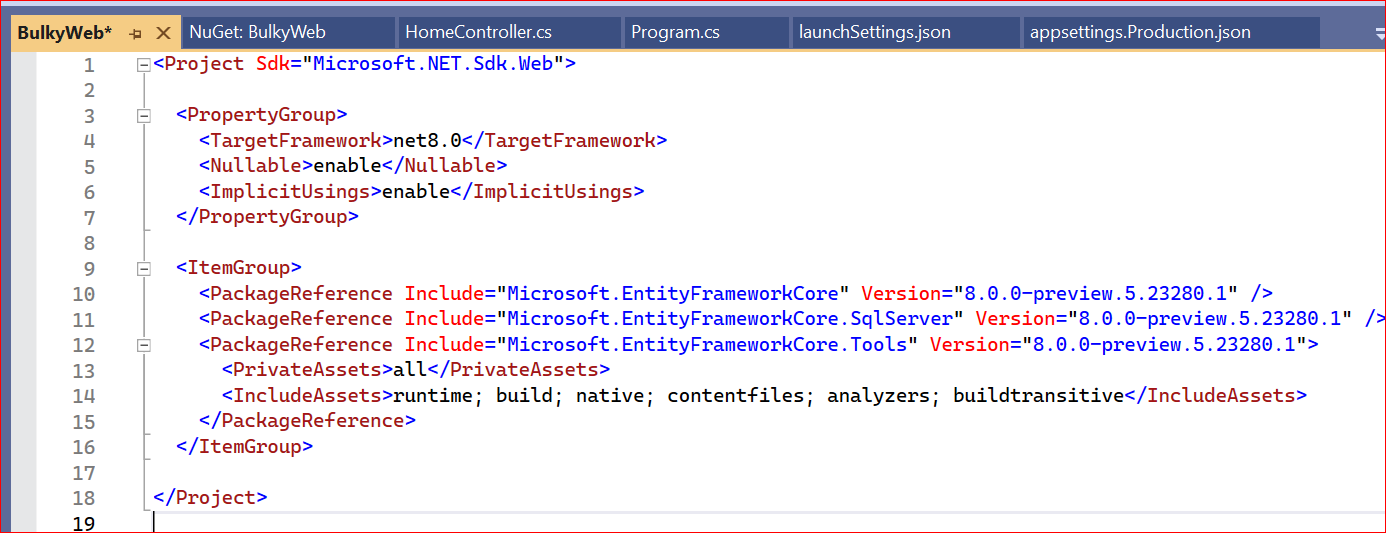
**Appsetting.json –connection string, secrete key**

For database connection: Packages required

Nuget packages for entity framework



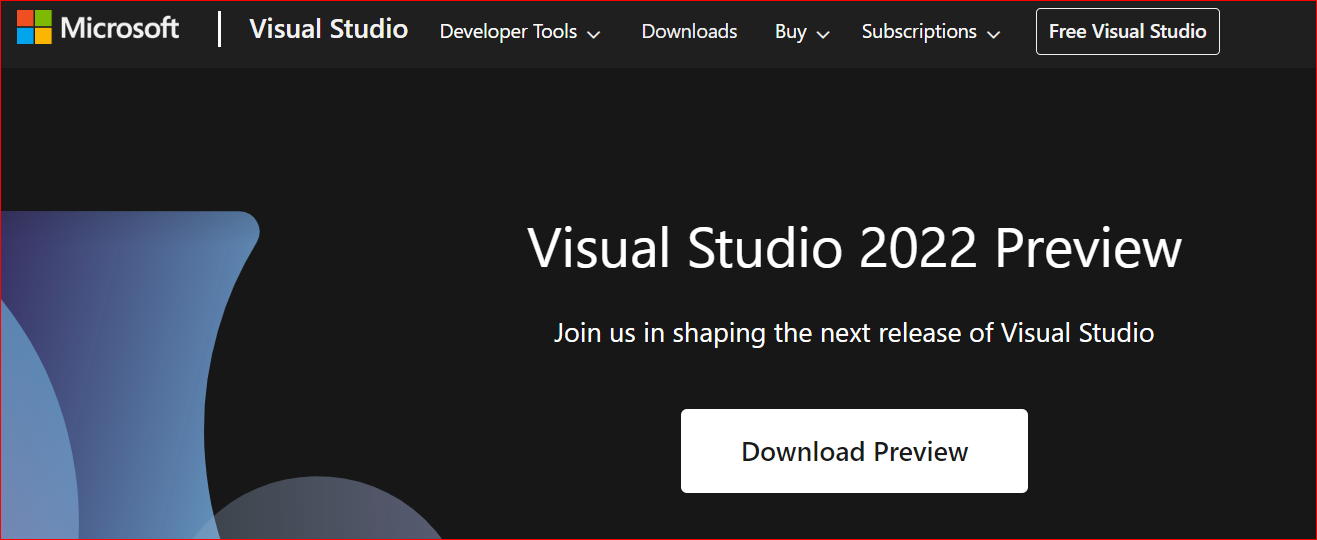


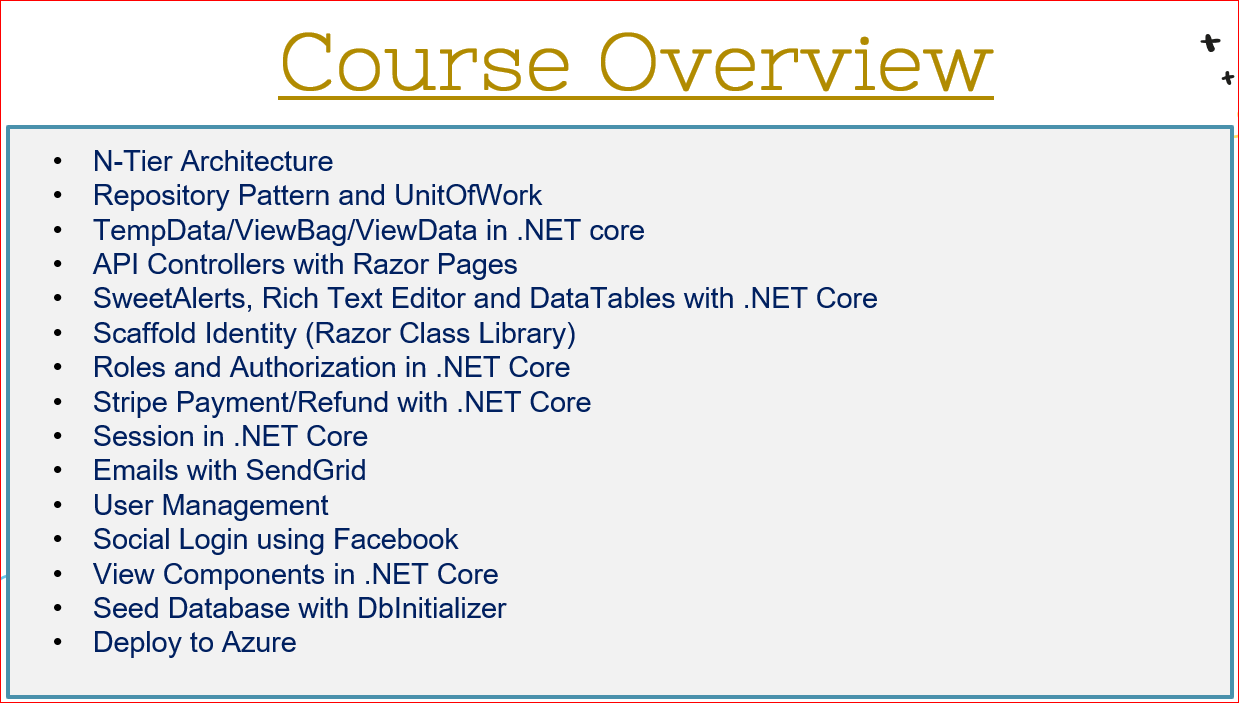
First download .Net 8.0

<https://dotnet.microsoft.com/en-us/download/dotnet/8.0>

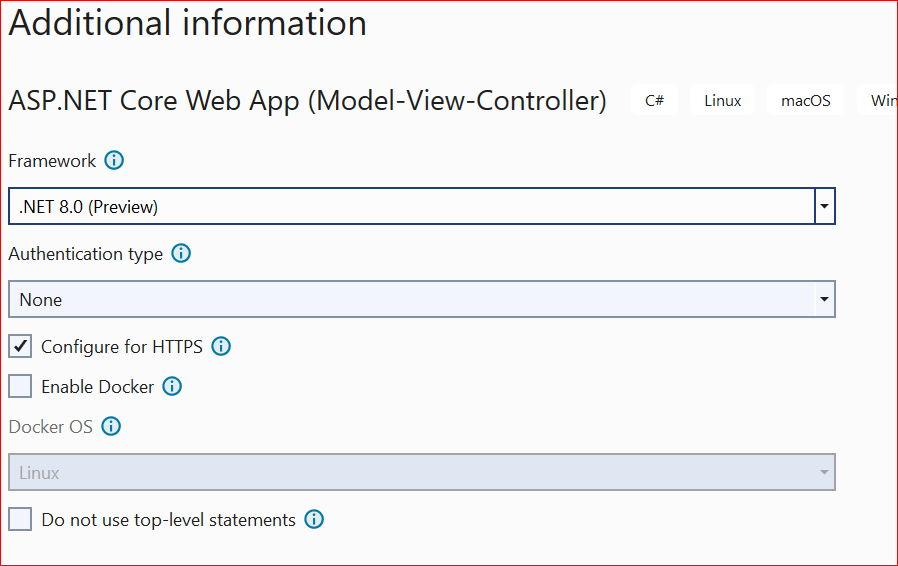
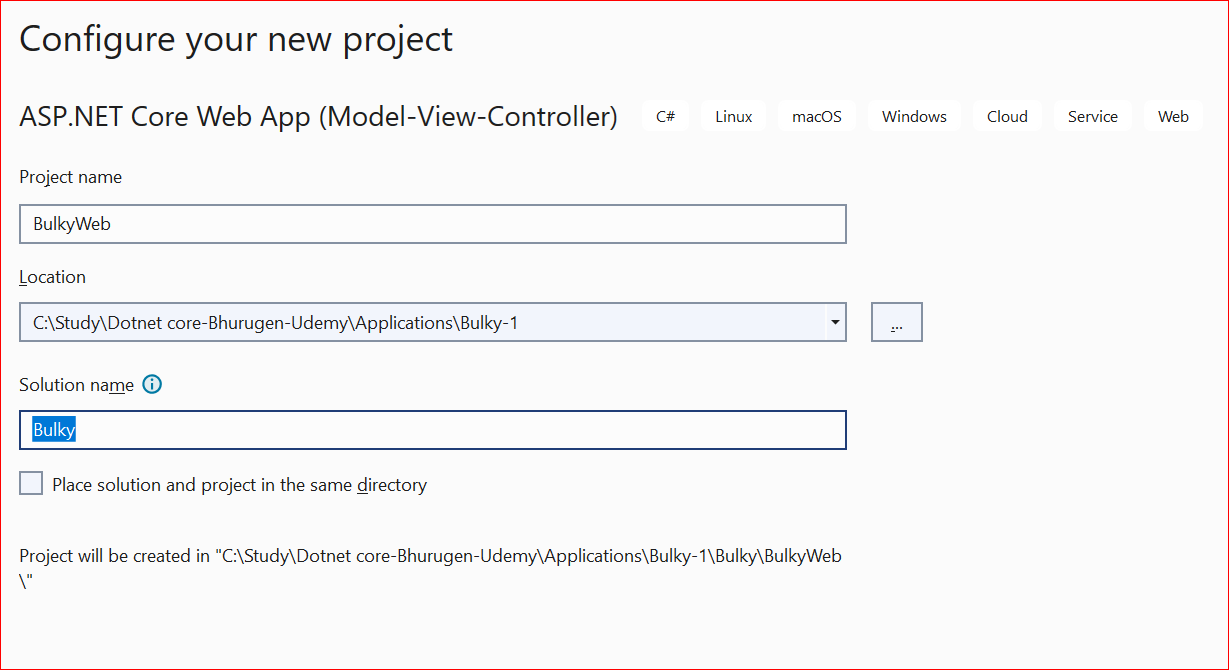
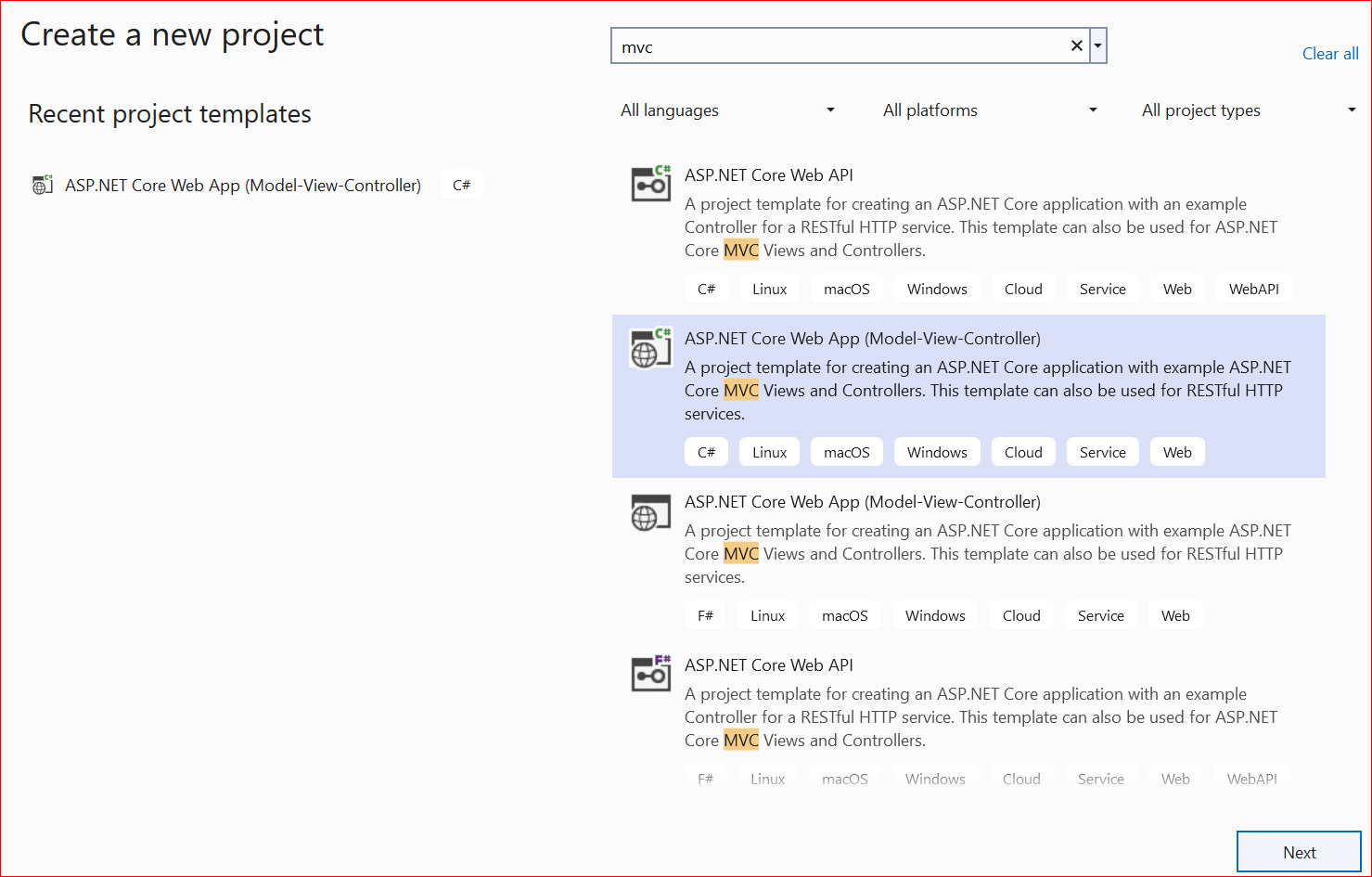


<https://visualstudio.microsoft.com/vs/preview/>

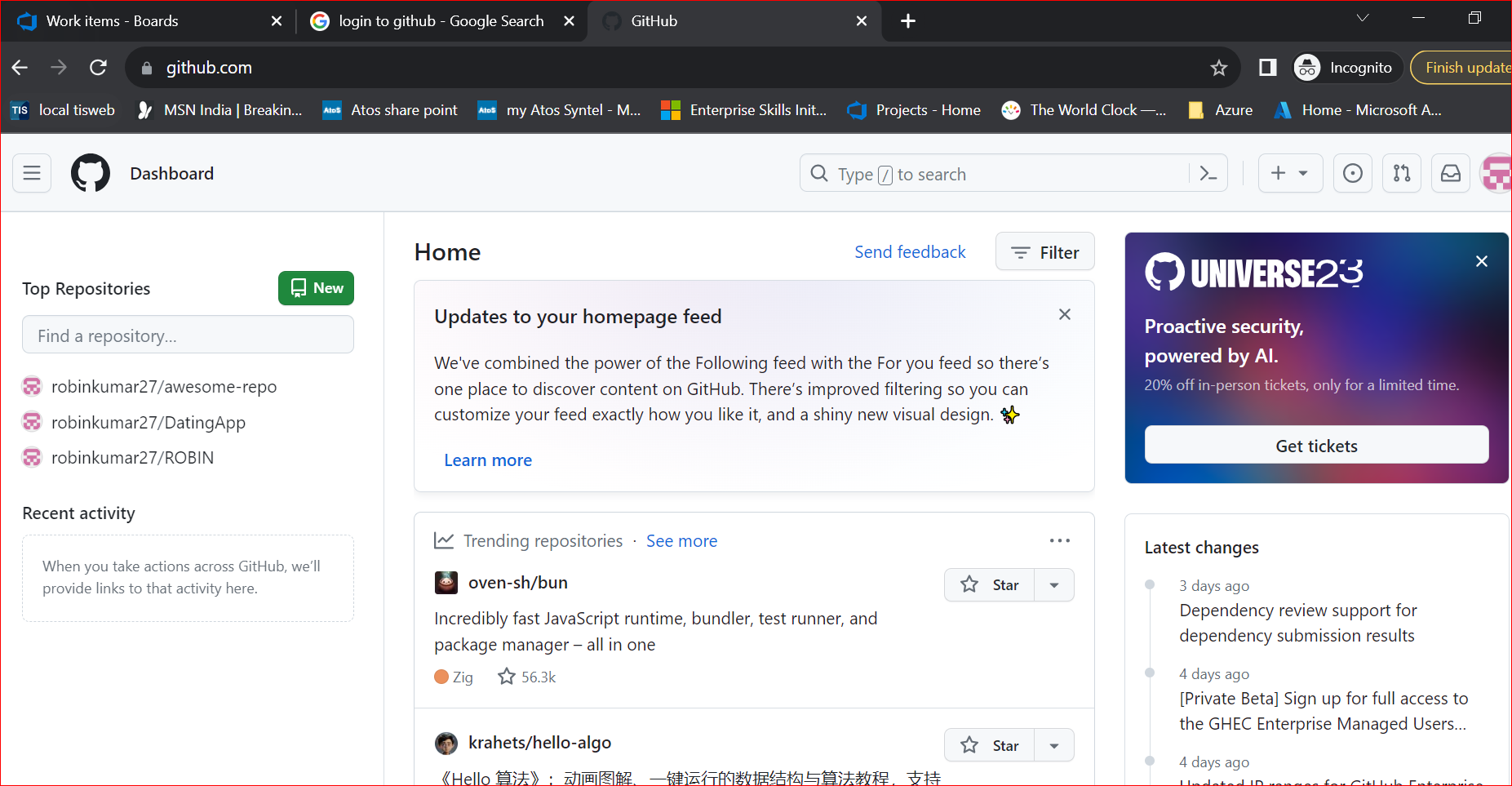




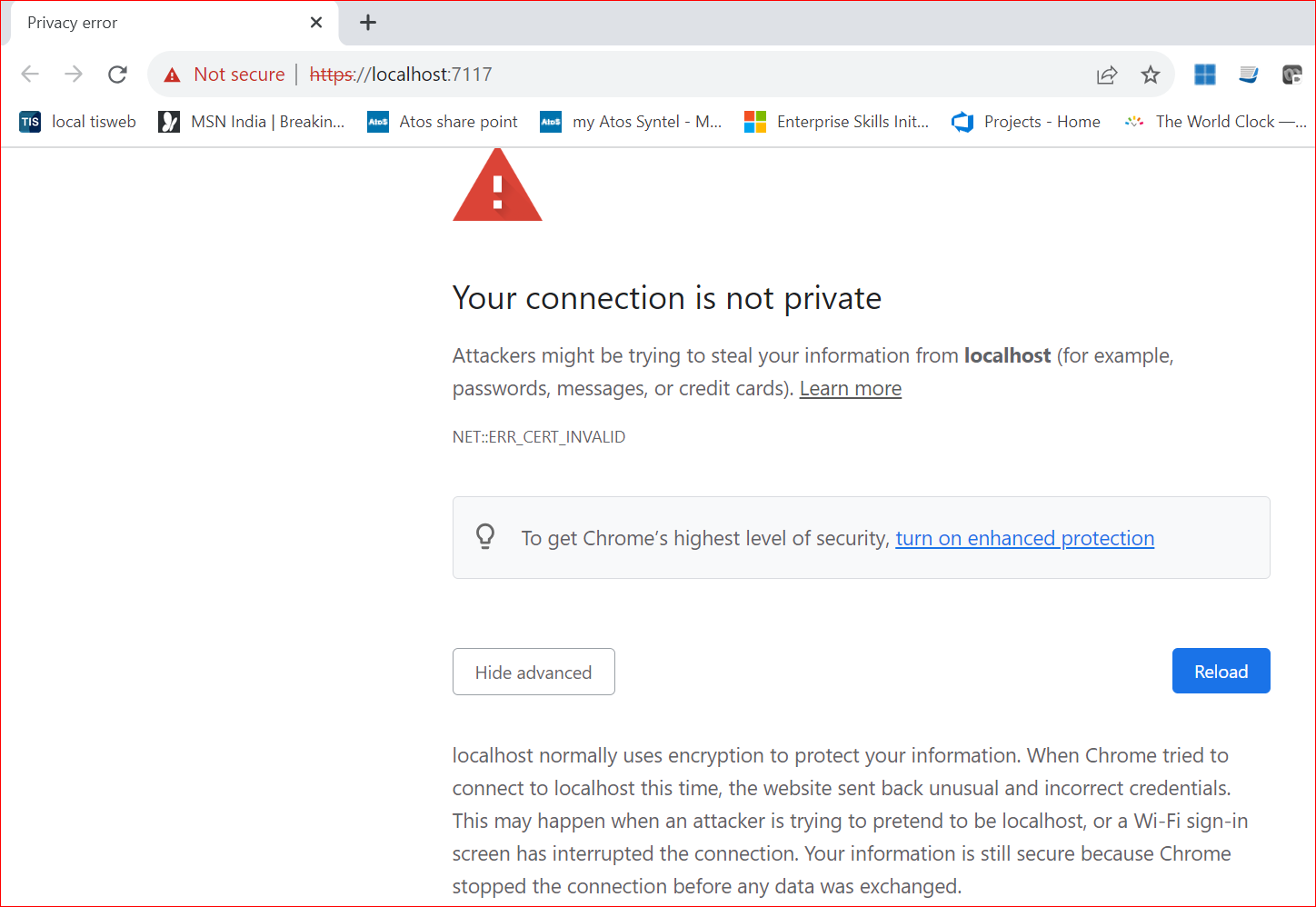
Select project template:



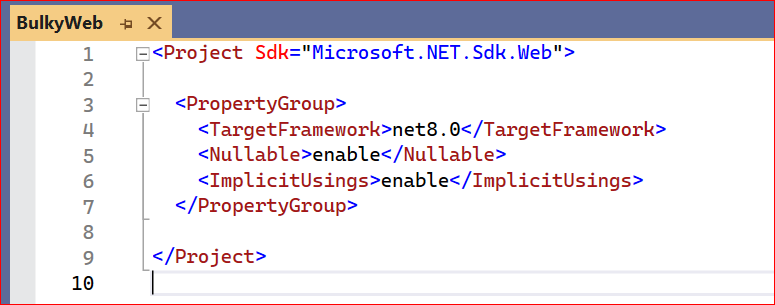
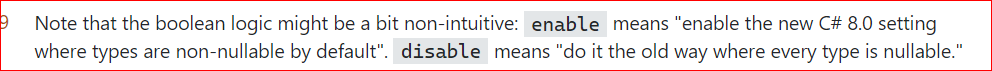
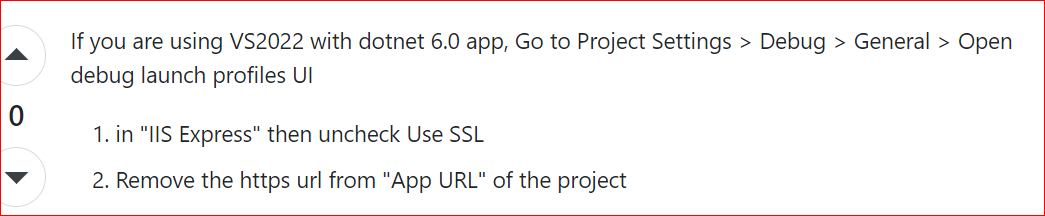
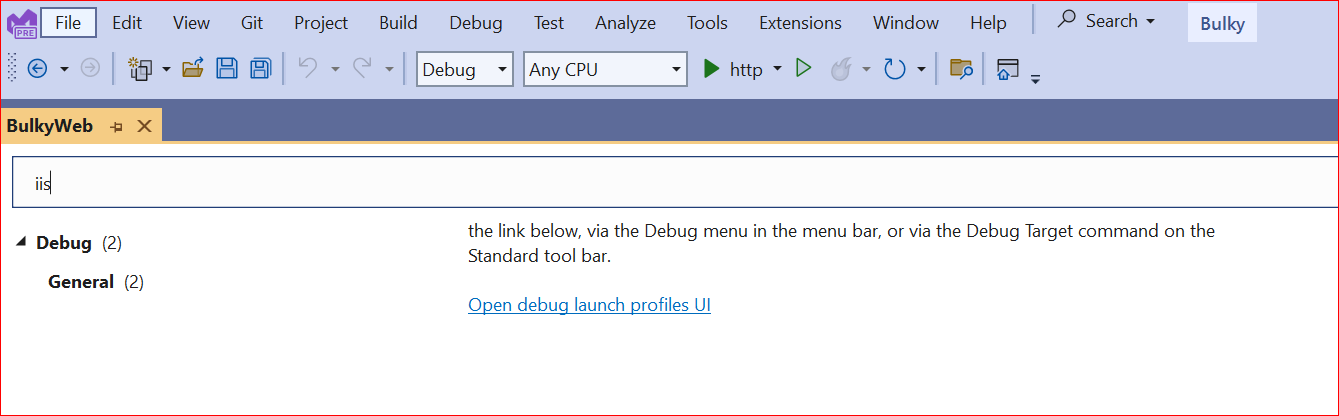
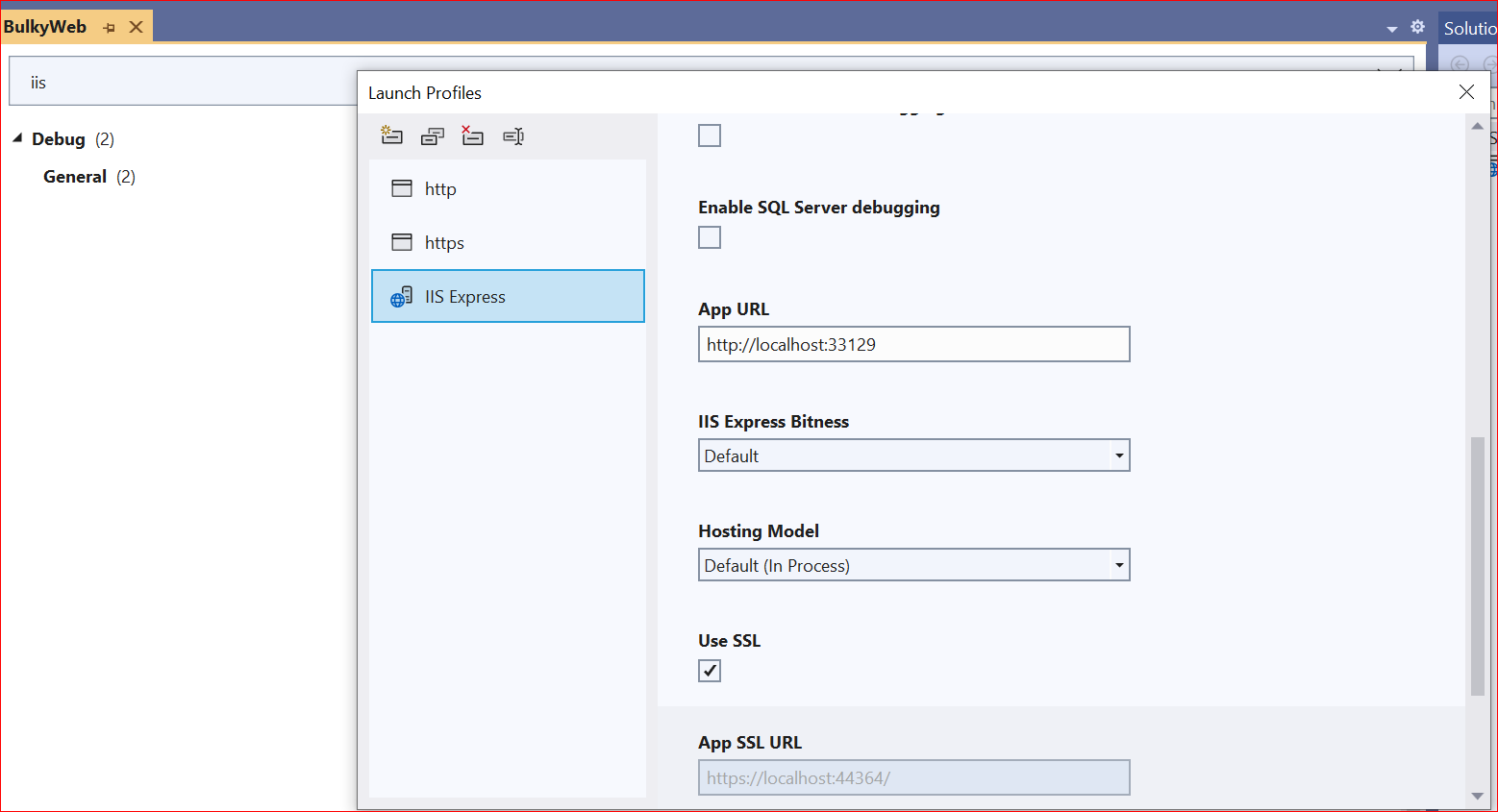
Login to github to store the code: username(robinkumar27/dib1234$)



First time it will give secure connection error:



Disable https: and change debug option to http or iis express



Implicit Global Usings

Implicit Global Usings are an opt in feature (kinda), that is new to .NET 6/C# 10. For existing projects that you are upgrading to .NET 6, you will need to add the following to your csproj file :

<ImplicitUsings>enable</ImplicitUsings>

However if you create a new project inside Visual Studio 2022 or using the latest SDK from the command line, this flag has already been enabled for you! So again, it’s somewhat opt in, it’s just that you will be opted in by default when creating a new project.

When enabled, implicit usings are actually a hidden auto generated file, inside your obj folder, that declares global using statements behind the scenes. In my case, if I go to my project folder then go obj/Debug/net6.0, I will find a file titled “ProjectName.GlobalUsings.g.cs”.

Opening this file, I can see it contains the following :

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

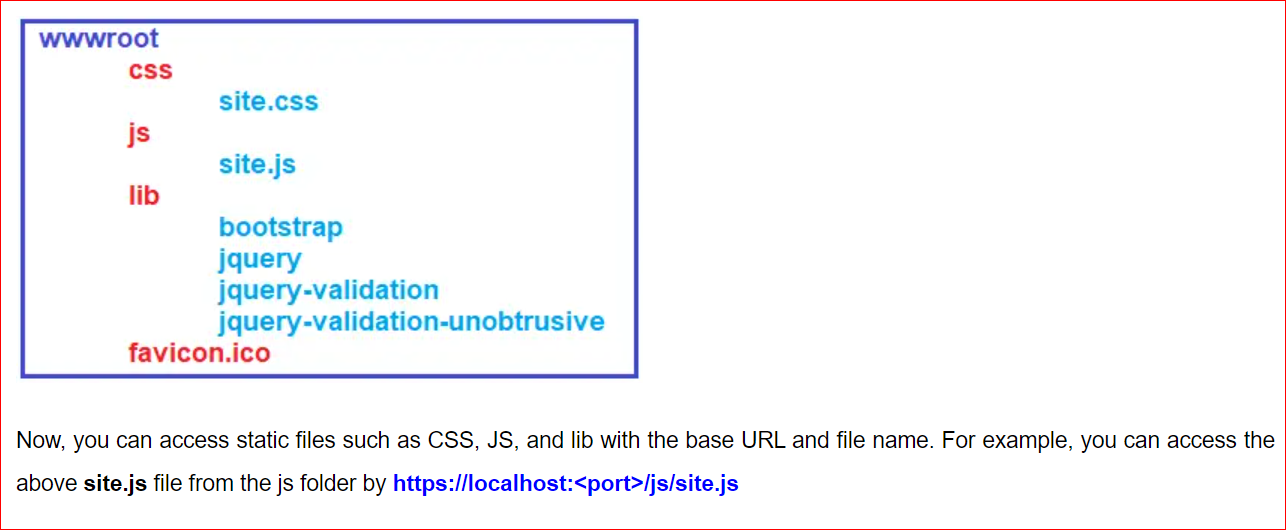
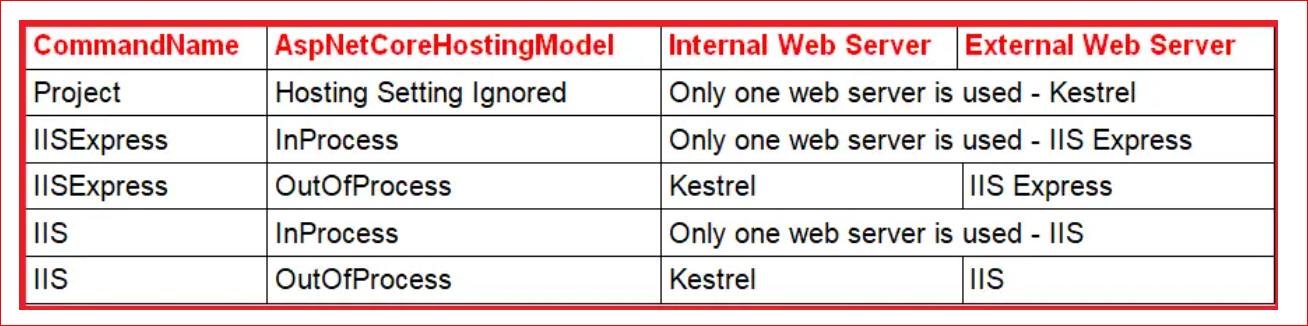
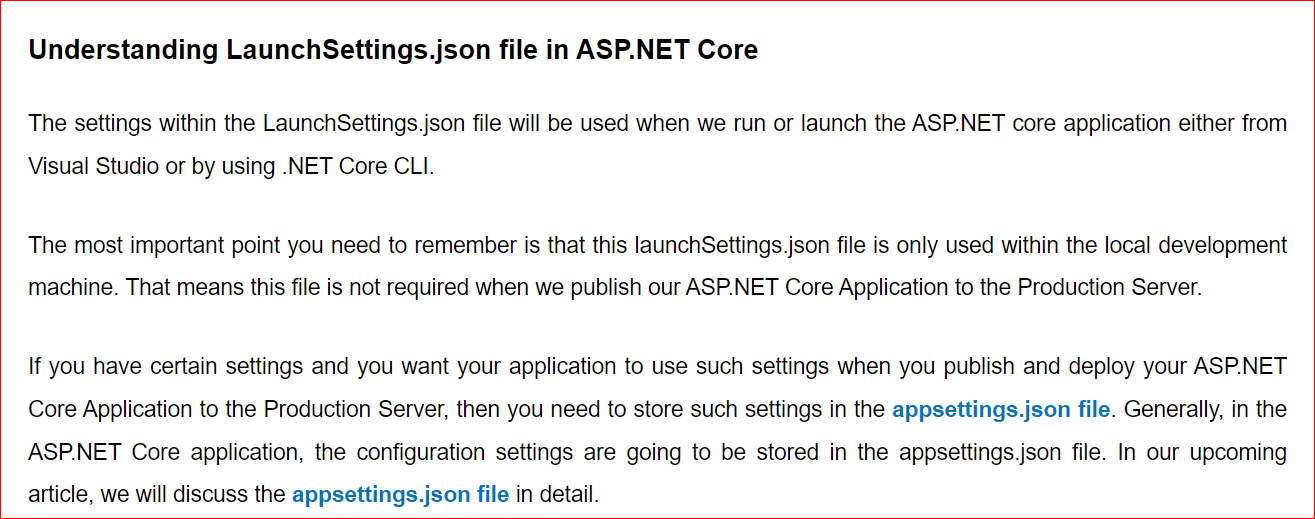
global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

Note the fact this is an auto generated file, and we can’t actually edit it here. But we can see that it declares a whole heap of global using statements for us.

The project I am demoing this from is actually a console application, but each main project SDK type has their own global imports.



**Wwwroot folder:**

This folder will basically host all of the static content of your .net core or your project. Now what is static content?

Static content basically means any CSS, any JavaScript, any NuGet packages or third party libraries,or if you have any images, files, PDFs, PowerPoint and so on.

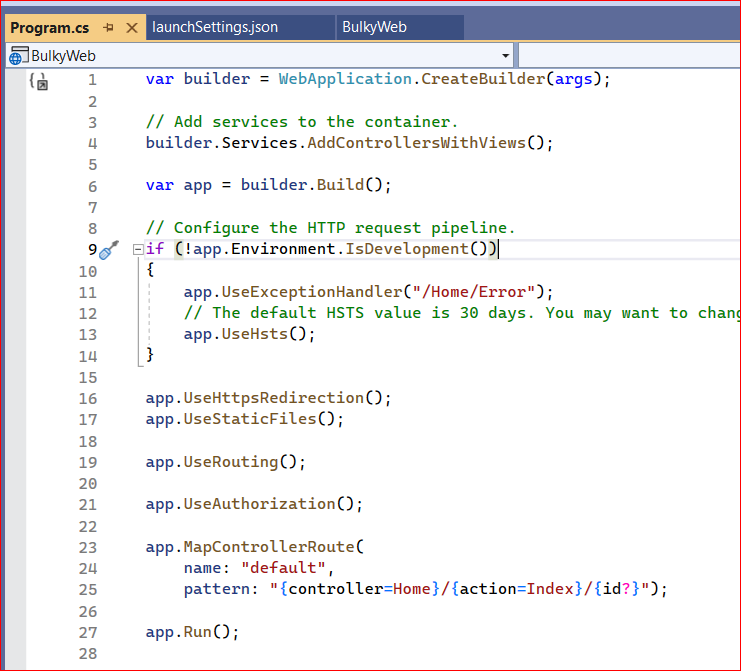
Anything that is static, which does not have an HTML code goes into the Www root folder.

**App settings** will be the place where you will host all of your connection string. Now when I mean connection string, it is not just connection string, but basically any secret key that you have for your application. An example of that will be for your email. You might be using SendGrid or you might have a secret key for email. You will add that in app settings.If you are using Azure Blob Storage or Azure Storage account, you will be storing all of those connection in one place.

**Program.cs**

First, we have to add some services to our container, and next we have to configure the request pipeline.

Pipeline basically means that when a request comes to an application, how do you want to process that?



It basically has a default pattern that if nothing is defined in the route, you should go to something called as a home controller. Inside there you should go to index action and ID here can be defined or not.

A question mark in .net basically means that ID can be defined or it can be null. So we are defining the default route that it should follow. And finally we have app .run that will basically run the project.

**In the MVC architecture**, Model represents the shape of the data.

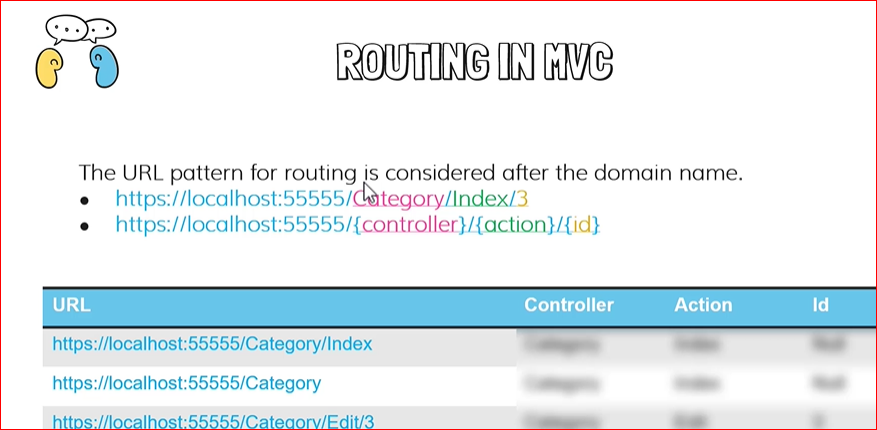
View, which stands for the user interface.View will control the HTML element of your web project.

But when you have the data, you will have to process or even fetch that data because data might be inside a database. So you need something to fetch that data and display that in a table on your HTML view that you have. That missing piece or heart of the MVC application is the controller, controller will basically handle the user request and it will act as an interface between model and view.

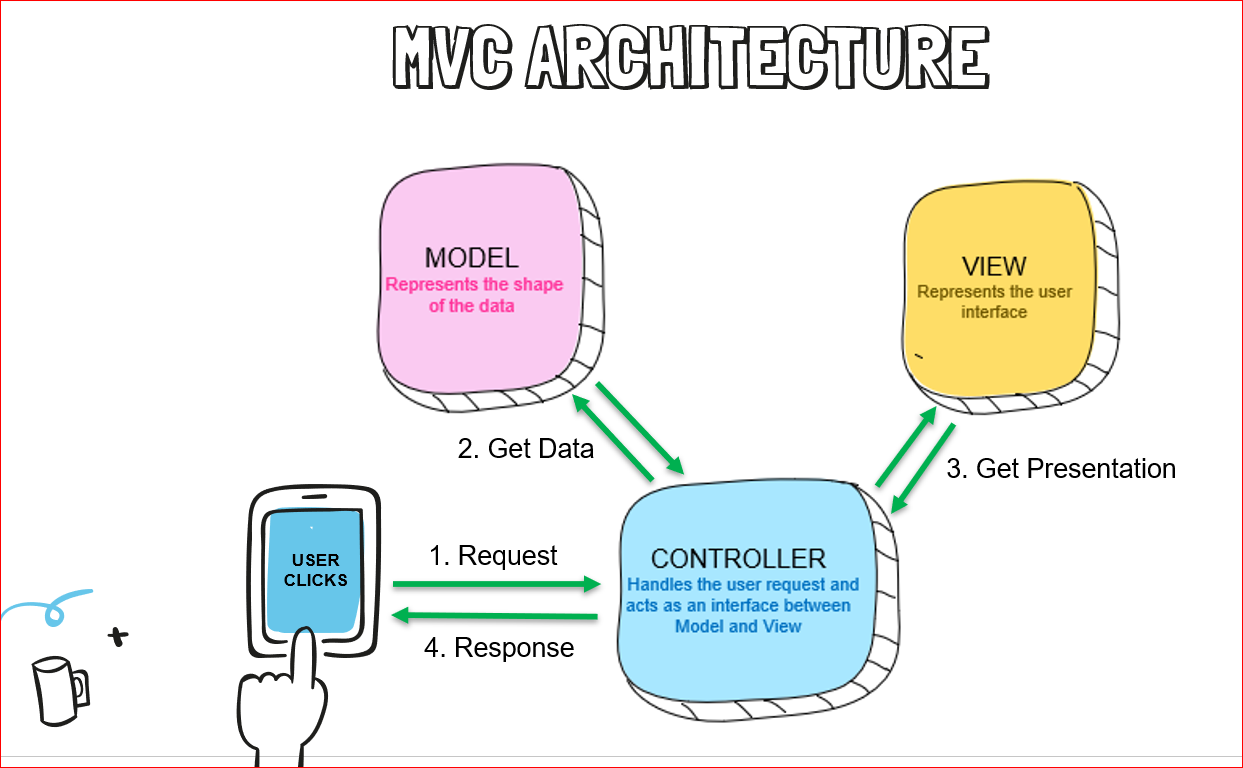
What happens is when a user clicks on one of the button or opens the website, the request will first go to the controller. Controller will then determine what model it has to fetch. It will retrieve all the data that is needed using the models and then it will pass all the data that is required to be displayed to the view component.

What view will do is it will add all the data in its HTML formatting and pass that data back to the controller and then controller will send that response back and the data or website will be displayed on your screen.

**Routing** basically defines that in the URL when you type something where it should send that request to. In an MVC application The typical route that we have is a controller and then action.



So if you have a home controller, you should have the name followed by the controller keyword and that must be placed inside the controllers folder.If you place it somewhere else, it will not work. These are the rules that are **defined for the MVC architecture**.



We have controllers and all the views that corresponds to that controller must be placed inside the views folder with a subfolder of the exact name as the controller name.

**But the main question**, how does it know where it has to load this particular view that we see? We are telling go to the home controller and get the index action. So at that point it will go to the home controller and it will execute this index action in index action.

We do not have any code, We are only saying return view. Well, it basically says that we have to return some view inside the views folder, but what view does it have to return?

And that is where the naming that we have given to folder comes into picture.If we have not defined a name inside this round bracket, it will use the same name as the action name. So it will return an index view. But where will it get that view from? It will get that view from the home controller. So it will go to the home folder inside views and there it will get the index dot CSSTML.

And if we type Home Forward slash privacy, it will execute the privacy action method.

And the view that it will return will be the privacy view, because that is the name of the action method. As I said before, if nothing is defined here, it will look for the same view with the name of the action method in the folder of that controller name.

Now, rather than that, I am saying that when the index is being called, I want you to return a view with the name of privacy. In that case, it will go to the home folder and look for a view with the name of privacy and views. Have an extension of dot css html.

But what about this header that we have on the top and we also have a footer.

We have seen what is there in the home view, but we have something called as shared and we have something called as underscore view imports and underscore view start.

If I open underscore layout here you can see we have doctype html.

We have a head here with some styling. We have a body and there we have the header.

In header you can see we have the home and privacy links here and we scroll down, we have something called as render body. This render body is a built in helper in the MVC and that will display anything that we want in the body. Now what that body will be, that will be determined by what is returned from the controller.

So underscored layout will be the master page of your application or the main page of your application. And in underscored layout, we will add all the JavaScript, all the CSS that we want to use globally in the application.

After that, let's go back and we have something called as validation scripts partial. There we have basically included to JavaScript.

Now when we progress with the course, we want to add client side validation and for that we will be using this JavaScript because of that on all pages, validations are not needed.

But typically, if that is a page or component that is used throughout the application, then we typically like to add an underscore. That way, when we look at the name, we can know that, okay, this view will be used throughout the application.

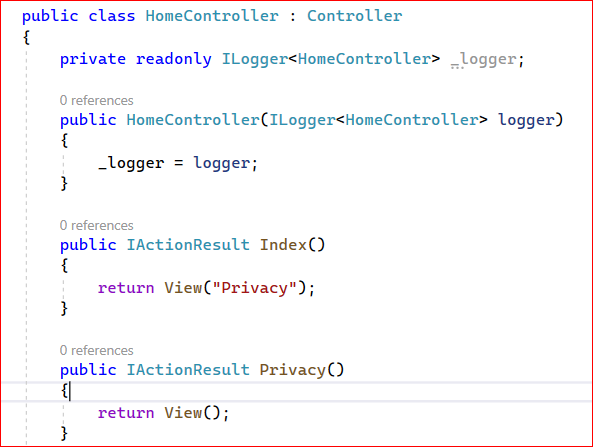
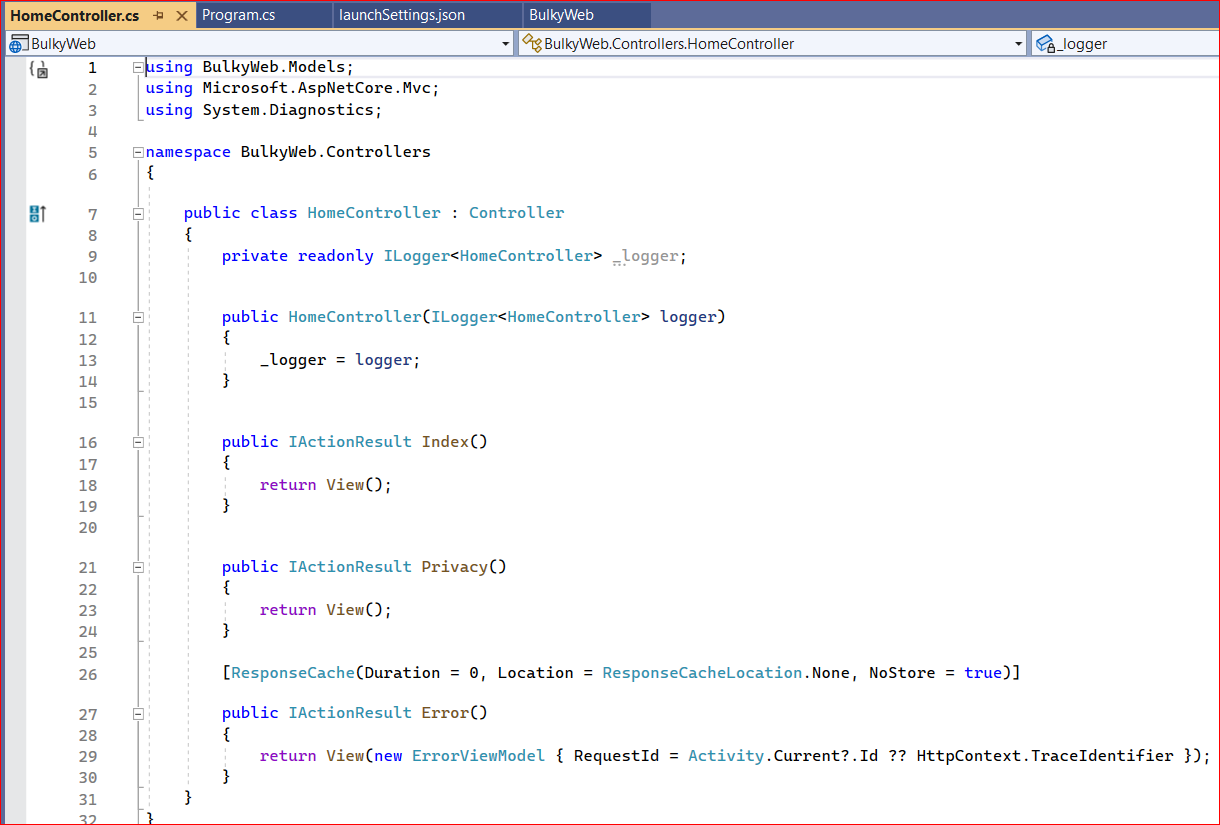
Partial view basically means that they cannot be displayed on the page by itself.

Now, one main question that I had when I was learning MVC was how does the application know that this underscored layout is the master page of the application? That is simple.

We define that inside the file which is view start dot cshtml. There we are telling that the layout of the application is underscored layout. If I change this to be layout here, let me stop that and run it. Things will not work.

But view import, you can think about that as a global import file. Rather than importing or writing the using statement in all file, you can add that here and that will automatically be available in all the views.

Now again, the view import or the using statements that we have added here will only be available in the views and not in controllers or models.



Typically we have a return type of, let's say an object or maybe a string or something like a list or a enumerable. An action result is one of the custom classes or rather interface that is implemented in the .Net framework and that basically implements all of the possible result type for an action method.